TESIS - EE185401

PERHITUNGAN KENAIKAN ATRIBUT *GAMEPLAY* UNTUK PEMAIN DAN *NON-PLAYER CHARACTER* PADA PERMAINAN *ROLE-PLAYING GAME* BERBASIS K-NN DAN NAIVE BAYES

NUR ROHMAN WIDIYANTO
07111650052005

DOSEN PEMBIMBING
Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
Dr. Supeno Mardi Susiki Nugroho, ST., MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN JARINGAN CERDAS MULTIMEDIA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2020

TESIS – EE185401

PERHITUNGAN KENAIKAN ATRIBUT *GAMEPLAY* UNTUK PEMAIN DAN *NON-PLAYER CHARACTER* PADA PERMAINAN *ROLE-PLAYING GAME* BERBASIS K-NN DAN NAIVE BAYES

NUR ROHMAN WIDIYANTO
07111650052005

DOSEN PEMBIMBING
Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
Dr. Supeno Mardi Susiki Nugroho, ST., MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN JARINGAN CERDAS MULTIMEDIA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2020

LEMBAR PENGESAHAN TESIS

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (MT)
di
Institut Teknologi Sepuluh Nopember
Oleh:
NUR ROHMAN WIDIYANTO
NRP: 07111650050205

Tanggal Ujian: 10 Juli 2020
Periode Wisuda: September 2020

Disetujui oleh:
Pembimbing:

1. Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng .....
   NIP: 195809161986011001

2. Dr. Supeno Mardi Susiki Nugroho, ST., MT .....
   NIP: 197003131995121001

Penguji:

1. Dr. Surya Sumpeno, ST., M.Sc. .....
   NIP: 196906131997021003

2. Reza Fuad Rachmadi, ST., MT., Ph.D .....
   NIP: 198504032012121001

3. Dr. Eko Mulyanto Yuniarno, ST., MT. .....
   NIP: 196806011995121009

Kepala Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas

Dodet Aditya Rrawan, ST., M.Eng, Ph.D.
NIP: 197311192000031001

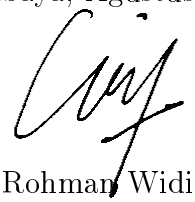
Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi buku Tesis dengan judul "**PERHITUNGAN KENAIKAN ATRIBUT *GAMEPLAY* UNTUK PEMAIN DAN *NON-PLAYER CHARACTER* PADA PERMAINAN *ROLE-PLAYING GAME* BERBASIS K-NN DAN NAIVE BAYES**" adalah benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Agustus 2020



Nur Rohman Widiyanto
07111650050205

Halaman ini sengaja dikosongkan

PERHITUNGAN KENAIKAN ATRIBUT *GAMEPLAY* UNTUK PEMAIN DAN *NON-PLAYER CHARACTER* PADA PERMAINAN *ROLE-PLAYING GAME* BERBASIS K-NN DAN NAIVE BAYES

Nama Mahasiswa	:	Nur Rohman Widiyanto
NRP	:	07111650050205
Pembimbing	:	1. Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
		2. Dr. Supeno Mardi Susiki Nugroho, ST., MT.

ABSTRAK

Permainan dengan genre *Role Playing Game* (RPG) merupakan permainan yang bersifat kompetitif, antara pemain melawan pemain ataupun melawan musuh yang berupa *Non-Player Character* (NPC). Banyak pengembang permainan dalam pembuatan permainan itu sendiri masih menggunakan cara manual dalam penentuan atribut *gameplay* untuk karakter pemain ataupun musuh. Terlebih lagi, saat sebuah permainan memiliki banyak karakter, seperti hanyalnya banyak karakter pemain (contohnya pada JRPG dan TRPG) dan juga banyak musuh. Pada penelitian ini diimplementasikan beberapa pendekatan seperti halnya $k-$NN, Distribusi Normal, dan Naive Bayes yang akan digunakan dalam program penghitung kenaikan atribut *gameplay* pada karakter pemain dan musuh secara otomatis. Pada program tersebut dibutuhkan parameter masukan yang akan menentukan atribut *gameplay* yang akan dihasilkan. Untuk karakter pemain, dibuatlah skenario atribut *gameplay* berdasarkan peran setiap karakter yang ingin dibuat seperti *knight*, *priest*, *assassin*, dan lain-lain. Sedangkan pada karakter musuh, dalam distribusi atribut *gameplay* juga dibagi menjadi beberapa tipe musuh yang dicontohkan seperti *mixed*, *hard strength*, *hard magic*, dan lain-lain. Setelah itu, hasil atribut *gameplay* diklasifikasi dengan menggunakan *Neural Network Multiclass Classification*. Hal tersebut bertujuan untuk menghitung tingkat kesesuaian atribut *gameplay* dari karakter pemain dan musuh yang dihasilkan. Operasi tersebut dilakukan secara terpisah pada atribut *gameplay* pemain dan musuh, karena tidak adanya hubungan saat pembuatan atau perhitungan. Masing-masing dibagi kedalam data *training* dan *testing*, dengan perbandingan 70% dan 30%. Proses tersebut menghasilkan keluaran berupa tipe karakter pemain dan musuh pada data *testing*, hal tersebut diperoleh dari proses *training*. Presentase kesesuaian tersebut diperoleh dengan membandingkan tipe pada karakter yang diperoleh dari hasil klasifikasi pada data *testing* dengan tipe dari karakter yang sebenarnya, maka diperolehlah sebuah presentase banyaknya karakter yang berhasil terklasifikasi.

Kata Kunci: *Role-Playing Game*, Atribut *Gameplay*, $k-$NN, Naive Bayes, *Neural Network*, Klasifikasi.

Halaman ini sengaja dikosongkan

THE CALCULATION OF PLAYER'S AND NON-PLAYER CHARACTER'S GAMEPLAY ATTRIBUTE GROWTH IN ROLE-PLAYING GAME WITH K-NN AND NAIVE BAYES

By	:	Nur Rohman Widiyanto
Student Identity Number	:	07111650050205
Supervisors	:	1. Prof. Dr. Ir. Mauridhi Hery. P, M.Eng.
		2. Dr. Supeno Mardi Susiki. N, ST., MT.

ABSTRACT

The game with the Role Playing Game (RPG) genre was a competitive game, between players against other players or enemies in the form of a Non-Player Character (NPC). Many game developers in making the game itself still use manual methods in determining gameplay attributes for player or enemy characters. Especially, when the game had many player characters (for example in JRPG and TRPG) and also the enemy. In this research, several approaches were implemented such as k-NN, Normal Distribution, and Naive Bayes which will be used in the program to automatically calculate the growth in gameplay attributes of the player and enemy characters. The program requires input parameters that will determine the result of gameplay attributes. For the player character, the gameplay attribute scenario was created based on the role of each character that user want to make such as knight, priest, assassin, and others. Whereas for enemy characters, the distribution of gameplay attributes also divided into several types of enemies that were exemplified, such as mixed, hard strength, hard magic, and others. After that, the results of the gameplay attributes classified using the Neural Network Multiclass Classification. It aims to calculate the level of suitability of the gameplay attributes of the resulting player and enemy characters. The operation is carried out separately on the player's and enemy's gameplay attributes because there is no correlation during creation or calculation. Each result divided into training and testing data, with a ratio of 70% and 30%. This process produces output in the form of player's and enemy's character types on the testing data, that obtained from the training process. The percentage of suitability obtained by comparing the types of characters from the classification results in the testing data with the types of the actual characters, so the percentage of the number of characters that were classified had obtained.

Keywords: Role-Playing Game, Gameplay Attributes, $k-$NN, Naive Bayes, Neural Network, Classification.

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji dan syukur kehadirat Allah SWT atas segala limpahan berkah, rahmat, serta hidayah-Nya, penulis dapat menyelesaikan penelitian ini dengan judul **Perhitungan Kenaikan Atribut *Gameplay* untuk Pemain dan *Non-Player Character* pada Permainan *Role-Playing Game* berbasis K-NN dan Naive Bayes**.

Penelitian ini disusun dalam rangka pemenuhan bidang riset di Jurusan Teknik Elektro ITS, Bidang Studi Teknik Komputer dan Telematika, serta digunakan sebagai persyaratan menyelesaikan pendidikan S1. Penelitian ini dapat terselesaikan tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Keluarga, Ibu, Bapak dan Saudara tercinta yang telah memberikan dorongan spiritual dan material dalam penyelesaian penelitian ini.
2. Bapak Dr. I Ketut Eddy Purnama, ST., MT. selaku Dekan Fakultas Teknologi Elektro dan Informatika Cerdas (FTEIC) ITS, Bapak Dedet Candra Riawan, ST., M.Eng, Ph.D. selaku Kepala Departemen Teknik Elektro FTEIC ITS, Bapak Ronny Mardiyanto, ST., MT., Ph.D. selaku Ketua Program Pasca Sarjana Teknik Elektro FTEIC ITS, Serta Bapak Eko Mulyanto, S.T., M.T. Selaku Ketua Program bidang keahlian Jaringan Cerdas Multimedia Teknik Elektro FTEIC ITS.
3. Secara khusus penulis mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng. dan Bapak Dr. Supeno Mardi Susiki Nugroho, ST., MT. atas bimbingan selama mengerjakan penelitian.
4. Bapak-ibu dosen pengajar Bidang Keahlian Jaringan Cerdas Multimedia, atas pengajaran, bimbingan, serta perhatian yang diberikan kepada penulis selama ini.
5. Seluruh teman-teman *B201-crew* Laboratorium Bidang Studi Teknik Komputer dan Telematika.

Kesempurnaan hanya milik Allah SWT, untuk itu penulis memohon segenap kritik dan saran yang membangun. Semoga penelitian ini dapat memberikan manfaat bagi kita semua. Amin.

Surabaya, Agustus 2020

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

LEMBAR PENGESAHAN iii

PERNYATAAN KEASLIAN v

ABSTRAK vii

ABSTRACT ix

KATA PENGANTAR xi

DAFTAR ISI xiii

DAFTAR GAMBAR xvii

DAFTAR TABEL xix

NOMENKLATUR xxi

1 PENDAHULUAN 1
- 1.1 Latar Belakang . . . 1
- 1.2 Rumusan Masalah . . . 4
- 1.3 Tujuan . . . 4
- 1.4 Batasan Masalah . . . 4
- 1.5 Kontribusi . . . 5

2 KAJIAN PUSTAKA 7
- 2.1 Kajian penelitian terkait . . . 7
- 2.2 Desain Permainan RPG dan Komponennya . . . 8
  - 2.2.1 Penempatan Peluang pada Permainan . . . 9
  - 2.2.2 Elemen Keterampilan Strategi . . . 11
  - 2.2.3 Menemukan Keseimbangan . . . 14
- 2.3 Machine Learning . . . 15
  - 2.3.1 K-Nearest Neighbor . . . 16
  - 2.3.2 Naive Bayes . . . 24
  - 2.3.3 Klasifikasi dengan Naive Bayes . . . 26
  - 2.3.4 Gaussian Naive Bayes . . . 30
- 2.4 Role-Playing Game . . . 32
  - 2.4.1 WRPG . . . 33
  - 2.4.2 JRPG . . . 34
  - 2.4.3 ARPG . . . 38
  - 2.4.4 TRPG . . . 38
  - 2.4.5 SRPG . . . 39

2.4.6 MMORPG 39
2.5 Artificial Neural Network 41
2.5.1 Activation Function 42
2.5.2 Arsitektur Neural Network 43
2.5.3 Training pada Neural Network 44

3 METODOLOGI PENELITIAN 47
3.1 Analisa Komponen Permainan RPG 48
3.1.1 Desain Skenario Pertarungan 49
3.1.2 Hubungan Sistem Pertarungan dengan Cerita 57
3.2 Pembuatan Atribut Gameplay Karakter Pemain 62
3.2.1 Distribusi Level, HP dan MP Pemain 64
3.2.2 Distribusi Elemen dan Kelemahan Pemain 65
3.2.3 Distribusi Atribut Gameplay Pemain 66
3.3 Pembuatan Atribut Gameplay Karakter Musuh 68
3.3.1 Distribusi Level Musuh 72
3.3.2 Distribusi Tipe Musuh 79
3.3.3 Distribusi Elemen dan Kelemahan Musuh 83
3.3.4 Distribusi HP, MP, dan Atribut Gameplay Musuh 85
3.4 Klasifikasi Hero permainan Dota 2 dengan Neural Network Multiclass Classification 94
3.4.1 Data Pre-processing 95
3.4.2 Korelasi Fitur 96
3.4.3 Desain Arsitektur Neural Network 98
3.4.4 Training 99
3.4.5 Pembuatan Model 101
3.5 Klasifikasi Karakter Pemain dengan Neural Network Multiclass Classification 102
3.5.1 Data Preparation 104
3.5.2 Korelasi Fitur 105
3.5.3 Desain Arsitektur Neural Network 106
3.5.4 Training dan Pembuatan Model 107
3.6 Klasifikasi Karakter Musuh dengan Neural Network Multiclass Classification 108
3.6.1 Data Preparation 109
3.6.2 Korelasi Fitur 110
3.6.3 Desain Arsitektur Neural Network 111
3.6.4 Training dan Pembuatan Model 112

4 HASIL DAN PEMBAHASAN 113
4.1 Hasil Atribut Gameplay Pemain (Single-Character) 113
4.1.1 Hasil Distibusi Level, HP, dan MP Pemain 114
4.1.2 Hasil Distibusi Elemen dan Kelemahan Pemain 116
4.1.3 Hasil Distibusi Atribut Gameplay Pemain 117
4.2 Hasil Atribut Gameplay Pemain (Multi-Character) 118
4.2.1 Hasil Distibusi Level, HP, dan MP Pemain 120

4.2.2 Hasil Distibusi Elemen dan Kelemahan Pemain 124
4.2.3 Hasil Distibusi Atribut Gameplay Pemain 125
4.3 Hasil Atribut Gameplay Musuh 128
4.3.1 Hasil Distibusi Level Musuh 128
4.3.2 Hasil Distibusi Tipe Musuh 131
4.3.3 Hasil Distribusi Element dan Kelemahan Musuh 133
4.3.4 Hasil Distribusi HP, MP, dan Atribut Gameplay pada Musuh 135
4.4 Hasil Klasifikasi Karakter pada Permainan Dota 2 142
4.5 Hasil Klasifikasi Karakter Pemain 145
4.6 Hasil Klasifikasi Karakter Musuh 147

5 PENUTUP **157**
5.1 Kesimpulan 157
5.2 Saran 158

Lampiran **159**

Daftar Pustaka **193**

Biografi Penulis **195**

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

1.1 Ilustrasi *turned-based* JRPG 1

2.1 Klasifikasi dengan $k-$NN. 18
2.2 Proses regresi dengan $k-$NN. 20
2.3 Contoh *video game* dengan genre *RPG*. 33
2.4 Contoh kustomisasi karakter pada WRPG. 34
2.5 Contoh kustomisasi atribut *gameplay* pada JRPG. 35
2.6 Contoh permainan digital dengan genre *turn-based*. 36
2.7 Mode pertarungan *turn-based* pada JRPG. 37
2.8 Mode pertarungan *real-time* pada JRPG. 37
2.9 MMORPG yang menyerupai JRPG. 41
2.10 Skruktur neuron dan deskripsi fungsinya. 41
2.11 Output fungsi linear 42
2.12 Contoh fungsi aktivasi. 43
2.13 Contoh arsitektur sederhana *Neural Network*. 44
2.14 Alur training untuk saat *forward pass* dan *backward pass*. 45

3.1 Urutan metodologi. 47
3.2 Skema pertarungan antar pemain. 49
3.3 Status dari pemain pada permainan bergenre RPG. 51
3.4 Skenario pertarungan *turn-based*. 53
3.5 Elemen pada permainan RPG. 54
3.6 Pengaruh elemen pada efektifitas serangan. 54
3.7 Pengaruh cerita terhadap tingkat kesulitan. 58
3.8 Distribusi jenis musuh sesuai dengan cerita. 60
3.9 Proses pembuatan atribut *gameplay* untuk karakter pemain. . . 62
3.10 *Class* diagram untuk atribut *gameplay* pemain. 63
3.11 Proses pembuatan atribut *gameplay* untuk karakter musuh. . . . 69
3.12 *Class* diagram untuk atribut *gameplay* musuh. 70
3.13 Atribut *gameplay* pada *hero* Centaur. 95
3.14 Proses klasifikasi *hero* pada permainan Dota 2. 95
3.15 *Gameplay* pada permainan Dota 2. 96
3.16 Arsitektur *Neural Network* untuk klasifikasi *hero* Dota 2. 98
3.17 Proses pengurangan *learning rate* saat *training*. 100
3.18 Validasi *loss* saat 2×10^4 *epoch*. 101
3.19 Validasi *loss* dengan pemberhentian otomatis. 101
3.20 Proses klasifikasi Karakter Pemain. 104
3.21 Arsitektur *Neural Network* untuk klasifikasi karakter pemain. . . 107
3.22 Proses klasifikasi Karakter Musuh. 108
3.23 Arsitektur *Neural Network* untuk klasifikasi karakter musuh. . . 111

4.1 Kenaikan HP setiap levelnya. 115
4.2 Kenaikan MP setiap levelnya. 115
4.3 Kenaikan atribut *gameplay* pemain setiap levelnya. 118
4.4 Kenaikan HP setiap levelnya pada karakter pertama. 122
4.5 Kenaikan HP setiap levelnya pada karakter kedua. 122
4.6 Kenaikan MP setiap levelnya pada karakter pertama. 123
4.7 Kenaikan MP setiap levelnya pada karakter kedua. 123
4.8 Kenaikan atribut *gameplay* karakter pertama pemain setiap levelnya. 127
4.9 Kenaikan atribut *gameplay* karakter kedua pemain setiap levelnya. 127
4.10 Distribusi Level Musuh. 129
4.11 Distribusi level musuh dalam bentuk distribusi normal. 130
4.12 Distribusi tipe musuh. 132
4.13 Distribusi kelemahan musuh. 134
4.14 Distribusi atribut *gameplay* HP musuh. 136
4.15 Distribusi atribut *gameplay* MP musuh. 137
4.16 Distribusi atribut *gameplay* musuh secara keseluruhan. 139
4.17 Distribusi *Strength* musuh. 139
4.18 Distribusi *Magic* musuh. 140
4.19 Distribusi *Endurance* musuh. 140
4.20 Distribusi *Speed* musuh. 141
4.21 Distribusi *Luck* musuh. 141
4.22 Akurasi saat *training* pada *hero* Dota 2. 142
4.23 *Loss* saat *training* pada *hero* Dota 2. 142
4.24 *Learning rate* saat *training* pada *hero* Dota 2. 143
4.25 Akurasi pada saat proses evaluasi pada *hero* Dota 2. 143
4.26 *Loss* saat proses evaluasi pada *hero* Dota 2. 143
4.27 Akurasi saat *training* pada karakter pemain. 145
4.28 *Loss* saat *training* pada karakter pemain. 145
4.29 *Learning rate* saat *training* pada karakter pemain. 146
4.30 Akurasi pada saat proses evaluasi pada karakter pemain. 146
4.31 *Loss* saat proses evaluasi pada karakter pemain. 146
4.32 Akurasi saat *training* pada karakter musuh (20k *epoch*). 148
4.33 *Loss* saat *training* pada karakter musuh (20k *epoch*). 148
4.34 *Learning rate* saat *training* pada karakter musuh (20k *epoch*). . 148
4.35 Akurasi saat proses evaluasi pada karakter (20k *epoch*). 149
4.36 *Loss* proses evaluasi pada karakter pemain (20k *epoch*). 149
4.37 Akurasi saat *training* pada karakter musuh (40k *epoch*). 150
4.38 *Loss* saat *training* pada karakter musuh (40k *epoch*). 150
4.39 *Learning rate* saat *training* pada karakter musuh (40k *epoch*). . 150
4.40 Akurasi saat evaluasi pada karakter musuh (40k *epoch*). 151
4.41 *Loss* saat evaluasi pada karakter musuh (40k *epoch*). 151

DAFTAR TABEL

3.1 Data masukan untuk pembuatan atribut *gameplay* pemain. . . . 64
3.2 Data masukan untuk pembuatan program pada musuh. 71
3.3 Hasil perhitungan korelasi matrix untuk fitur 97
3.4 Hasil perhitungan korelasi matrix pada data pemain 105
3.5 Hasil perhitungan korelasi matrix pada data musuh 110

4.1 Hasil Perhitungan HP dan MP 114
4.2 Distribusi elemen pada karakter pemain 116
4.3 Distribusi atribut *gameplay* pada karakter pemain 117
4.4 Data masukan untuk pembuatan atribut *gameplay* karakter pertama. 119
4.5 Data masukan untuk pembuatan atribut *gameplay* karakter kedua. 119
4.6 Hasil Perhitungan HP dan MP karakter pertama 120
4.7 Hasil Perhitungan HP dan MP karakter kedua 121
4.8 Distribusi elemen pada karakter pemain pertama 124
4.9 Distribusi elemen pada karakter pemain kedua 124
4.10 Distribusi atribut *gameplay* pada karakter pertama pemain . . . 125
4.11 Distribusi atribut *gameplay* pada karakter pemain 126
4.12 Hasil level yang dibuat untuk musuh. 128
4.13 Hasil level yang dibuat untuk musuh. 131
4.14 Hasil level yang dibuat untuk musuh. 133
4.15 Distribusi atribut *gameplay* HP dan MP musuh. 135
4.16 Distribusi atribut *gameplay* musuh. 137
4.17 Hasil proses *testing* dengan data *testing* pada *hero* Dota 2. . . . 144
4.18 Hasil proses *testing* dengan data *testing* pada karakter pemain. . 147
4.19 Hasil proses *testing* dengan data *testing* pada karakter musuh. . 152

5.1 Hasil keseluruhan data HP dan MP pada pemain. 159
5.2 Hasil keseluruh data atribut *gameplay* pada pemain (Bag. 1). . 160
5.3 Hasil keseluruh data atribut *gameplay* pada pemain (Bag. 2). . 161
5.4 Hasil keseluruh data atribut *gameplay* pada pemain (Bag. 3). . 162
5.5 Hasil keseluruh data atribut *gameplay* pada pemain (Bag. 4). . 163
5.6 Hasil keseluruhan data HP dan MP karakter pertama pada pemain (*multi-character*). 164
5.7 Hasil keseluruh data atribut *gameplay* karakter pertama (*multi-character*). (Bag. 1). 165
5.8 Hasil keseluruh data atribut *gameplay* karakter pertama (*multi-character*). (Bag. 2). 166
5.9 Hasil keseluruh data atribut *gameplay* karakter pertama (*multi-character*). (Bag. 3). 167

5.10 Hasil keseluruh data atribut *gameplay* karakter pertama (*multi-character*). (Bag. 4). 168
5.11 Hasil keseluruhan data HP dan MP karakter kedua pada pemain (*multi-character*). 169
5.12 Hasil keseluruh data atribut *gameplay* karakter kedua (*multi-character*). (Bag. 1). 170
5.13 Hasil keseluruh data atribut *gameplay* karakter kedua (*multi-character*). (Bag. 2). 171
5.14 Hasil keseluruh data atribut *gameplay* karakter kedua (*multi-character*). (Bag. 3). 172
5.15 Hasil keseluruh data atribut *gameplay* karakter kedua (*multi-character*). (Bag. 4). 173
5.16 Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 1). 174
5.17 Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 2). 175
5.18 Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 3). 176
5.19 Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 4). 177
5.20 Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 5). 178
5.21 Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 6). 179
5.22 Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 7). 180
5.23 Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 8). 181
5.24 Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 9). 182
5.25 Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 10). 183
5.26 Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 11). 184
5.27 Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 12). 185
5.28 Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 13). 186
5.29 Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 14). 187
5.30 Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 15). 188
5.31 Dataset atribut *gameplay hero* Dota 2 untuk *training* (Bag. 1). 189
5.32 Dataset atribut *gameplay hero* Dota 2 untuk *training* (Bag. 2). 190
5.33 Dataset atribut *gameplay hero* Dota 2 untuk *testing*. 191
5.34 Dataset karakter pemain untuk *training*. 191
5.35 Dataset karakter pemain untuk *testing*. 191

NOMENKLATUR

i	= Urutan indeks atau iterasi.
n	= Jumlah iterasi.
x	= Data x atau titik koordinat data.
x_0	= Titik koordinat data ke 0.
x_i	= Data x ke i.
x_n	= Data x ke n.
θ	= Data θ.
θ_i	= Data θ ke i.
$min\ d$	= Jarak minimum
Y	= Titik data Y.
y	= Bagian dari Y dengan banyak data.
X	= Titik data X.
y	= Bagian dari X dengan banyak data.
D	= Distance atau jarak.
p	= Data p atau titik koordinat data.
p_i	= Titik koordinat data ke i.
y_{pred}	= Hasil prediksi dari titik koordinat yang dicari.
y_i	= Titik koordinat yang dicari oleh y_{pred}.
k	= Nilai k dalam k-NN.
W	= Bobot untuk tetangga terdekat.
$P(h\|d)$	= Probabilitas hipotesis h berdasrkan data d. Hal ini disebut probabilitas posterior.
$P(d\|h)$	= Probabilitas dari data d berdasarkan hipotesis h yang bernilai benar.
$P(h)$	= Probabilitas hipotesis h yang bernilai benar terlepas dari data secara keseluruhan.
$P(d)$	= Probabilitas data secara keseluruhan terlepas dari hipotesis.
$MAP(h)$	= Maximum a Posteriori Probability.
$P(C)$	= *Class pobability*.
$C_{(0,1,..n)}$	= *Class* ke n untuk dicari probabilitasnya.
$W_{(0,1,..n)}$	= *Weather* ke n atau kondisi cuaca pada contoh kasus Naive Bayes.
$\bar{x}$	= Rata-rata nilai x.
$\sigma(x)$	= Standar deviasi.
PDF	= Probability Density Function.
y_{out}	= *Output* pada *Neural Network*.
w	= *Weight* pada *Neural Network*.
x_{in}	= *Input* pada *Neural Network*.
b	= Bias pada *Neural Network*.
$Predict$	= Hasil prediksi dari *Neural Network*.
$Target$	= Target hasil prediksi dari *Neural Network*.
$Loss$	= Selisih antara $Target$ dan $Predict$.
B	= Jumlah boss.

tB = Waktu yang dibutuhkan untuk melawan boss.
tB_{total} = Jumlah waktu melawan bos secara keseluruhan.
HP = Nilai HP yang dicari pada karakter pemain.
MP = Nilai MP yang dicari pada karakter pemain.
N_p = Level maksimum dari karakter pemain.
$P(C_{St=1})$ = Probabilitas munculnya *Attributes to Assign* dengan angka pertama.
$P(C_{St=2})$ = Probabilitas munculnya *Attributes to Assign* dengan angka kedua.
$C_{St=1}$ = Angka pertama dalam *Attributes to Assign*.
$C_{St=2}$ = Angka kedua dalam *Attributes to Assign*.
$MaxSt$ = Nilai atribut *gameplay* maximum.
$D(MaxSt, p)$ = Jarak antara koordinat data $MaxSt$ (maksimum atribut *gameplay*) dan p (jumlah atribut *gameplay* saat ini).
St_i = Atribut *gameplay* pada level ke i.
St' = Variabel vektor untuk setiap ST_i.
i_{st} = Nilai pertambahan atribut *gameplay* yang ingin ditambahkan.
LN = Range antara *Max Level* dan *Min Level* atau banyaknya musuh.
N_{LC} = Jumlah kelompok atau *cluster*.
SL_i = Jumlah sebaran level musuh pada setiap *cluster*, dengan jumlah *cluster* sebanyak N_{LC}.
SL' = Variabel vektor untuk jumlah sebaran level pada setiap *cluster* atau kumpulan level musuh yang jumlahnya sebanyak N_{LC}.
EN = Banyaknya musuh.
SE_i = Jumlah musuh pada setiap *cluster*, dengan jumlah *cluster* sebanyak N_{LC}.
SE' = Variabel vektor untuk jumlah musuh pada setiap *cluster* atau kumpulan jumlah musuh yang jumlahnya sebanyak N_{LC}.
Sc = Skala atau *scale* untuk mempersempit persebaran level dan musuh.
SSL_i = Jumlah sebaran level pada setiap *sub-cluster*, dengan jumlah *sub-cluster* atau bagian kumpulan level musuh sebanyak Sc.
SSL' = Variabel vektor untuk jumlah sebaran level pada setiap *sub-cluster*, dengan jumlah *sub-cluster* atau bagian kumpulan level musuh sebanyak Sc.
SSE_i = Jumlah musuh pada setiap *sub-cluster*, dengan jumlah *sub-cluster* atau bagian kumpulan musuh sebanyak Sc.
SSE' = Variabel vektor untuk jumlah musuh pada setiap *sub-cluster*, dengan jumlah *sub-cluster* atau bagian kumpulan jumlah musuh sebanyak Sc.

ELv_{jk} = Level dari setiap musuh dalam SSL dan SSE.
$P(ELv_{jk})$ = Probabilitas munculnya level pada setiap musuh dalam SSL dan SSE.
$\bar{E}lv$ = Rata-rata dari level setiap musuh.
$\sigma(ELv)$ = Varian dari level setiap musuh.
$PDF(ELv, \bar{E}lv, \sigma)$ = Gaussian PDF dari level musuh.
DP = Variabel vektor untuk presentase distribusi persebaran tipe musuh.
N_{DP} = Jumlah set atau indeks presentse distribusi persebaran tipe musuh.
DP_i = Indeks presentse distribusi persebaran tipe musuh.
DN' = Variabel vektor untuk target distribusi jumlah musuh setiap tipenya.
DN_i = Indeks target distribusi jumlah musuh setiap tipenya.
DL' = Variabel vektor untuk distribusi jumlah musuh setiap tipenya.
DL_i = Indeks distribusi jumlah musuh setiap tipenya.
rDL' = Variabel vektor untuk sisa musuh yang belum terdistribusi setiap tipenya.
rDL_i = Indeks sisa musuh yang belum terdistribusi setiap tipenya.
M_{DL} = Jumlah set atau indeks presentse distribusi persebaran tipe musuh.
N_{EN} = Jumlah musuh.
ET_j = Tipe pada tiap satu musuh.
$P(ET_j)$ = Probabilitas tipe pada tiap satu musuh.
M_{EN} = Jumlah musuh.
N_{ET} = Jumlah musuh yang memiliki tipe.
rDL_i = Indeks sisa musuh yang belum terdistribusi setiap tipenya.
M_{rDL} = Jumlah set untuk sisa musuh yang belum memiliki tipe pada rDL.
N_{rET} = Jumlah sisa musuh yang belum terdistribusi pada rDL sebagai batas.
$P(rET_j)$ = Probabilitas untuk tipe pada setiap musuh yang belum memiliki tipe.
ElN = Nama-nama elemen yang dapat digunakan oleh karakter musuh.
$DmgNa$ = Nama-nama respon atau efek serangan untuk setiap elemen atau ElN pada musuh.
$DmgNu'$ = Varaibel vektor berisi penomoran setiap $DmgNa$ atau efek serangan.
N_{Na} = Jumlah $DmgNa$ pada satu karakter musuh.
$DmgNu_0$ = Respon atau efek serangan dari satu elemen yang terpilih.

$P(DmgNu_0)$ = Probabilitas diperolehnya $DmgNu_0$ dari $DmgNa$ dari ElN.
M_{Nu} = Jumlah respon atau efek serangan pada musuh.
$DmgNu_i$ = Probabilitas respon atau efek serangan pada satu karakter musuh.
$P(DmgNu_i)$ = Probabilitas respon atau efek serangan pada satu karakter musuh.
$DmgNu_{ij}$ = Respon atau efek serangan pada setiap karakter musuh.
N_{Nu} = Dimensi vektor $DmgNu'$ pada satu karakter musuh.
$DmgEN'$ = Variabel vektor untuk $DmgNu'$ pada semua karakter musuh.
N = Jumlah tipe pada karakter musuh.
ET_i = Jumlah setiap tipe musuh pada karakter musuh.
ET' = Variabel vektor untuk jumlah setiap tipe pada karakter musuh.
MX = Variabel vektor untuk musuh bertipe *Mixed* pada ET_0.
HM = Variabel vektor untuk musuh bertipe *Hard Magic* pada ET_1.
SM = Variabel vektor untuk musuh bertipe *Soft Magic* pada ET_2.
HS = Variabel vektor untuk musuh bertipe *Hard Strength* pada ET_3.
SS = Variabel vektor untuk musuh bertipe *Soft Strength* pada ET_4.
ET_N = Tipe musuh ke N.
ST_N = Nama variabel Tipe musuh pada ET_N.
N_{st} = Jumlah atribut *gameplay* dalam satu tipe musuh.
$minHP$ = Nilai minimum HP untuk mencari HP musuh.
bHP = Nilai batas bawah HP untuk mencari HP musuh.
bHP_i = Nilai batas bawah HP untuk mencari HP pada setiap musuh.
$maxtHP$ = Nilai maximum HP untuk mencari atribut *gameplay* musuh.
tHP = Nilai batas atas HP untuk mencari HP musuh.
tHP_i = Nilai batas atas HP untuk mencari HP pada setiap musuh.
$P(HP_i)$ = Probabilitas munculnya HP pada setiap karakter.
$minMP$ = Nilai minimum MP untuk mencari MP musuh.
bMP = Nilai batas bawah MP untuk mencari MP musuh.
bMP_i = Nilai batas bawah HP untuk mencari HP pada setiap musuh.
$maxtMP$ = Nilai maximum MP untuk mencari atribut *gameplay* musuh.
tMP = Nilai batas atas MP untuk mencari MP musuh.
tMP_i = Nilai batas atas MP untuk mencari MP pada setiap musuh.
$P(MP_i)$ = Probabilitas munculnya MP pada setiap karakter.

MX_1 = Variabel untuk musuh bertipe *Mixed* yang fokus dengan HP.
MX_2 = Variabel untuk musuh bertipe *Mixed* yang fokus dengan MP.
NM_{hp} = Jumlah musuh bertipe *Mixed* yang fokus dengan HP.
$minSt$ = Nilai minimum atribut *gameplay* untuk mencari atribut *gameplay* musuh.
bSt = Set variabel untuk nilai batas bawah atribut *gameplay* untuk mencari atribut *gameplay* musuh.
bSt_{ij} = Set variabel untuk nilai batas bawah atribut *gameplay* pada setiap karakter musuh.
$minSt_{ij}$ = Nilai minimum atribut *gameplay* pada setiap karakter musuh.
$maxSt_{ij}$ = Nilai maximum atribut *gameplay* pada setiap karakter musuh.
tSt = Set variabel untuk nilai batas atas atribut *gameplay* untuk mencari atribut *gameplay* musuh.
tSt_{ij} = Set variabel untuk nilai batas atas pada setiap karakter musuh.

Halaman ini sengaja dikosongkan

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Permainan dengan basis *Role-Playing Game* (RPG) adalah permainan yang mana pemain memerankan sebuah karakter khusus dalam sebuah cerita. Pada permainan tersebut, pemain memiliki tujuan untuk menjalankan misi dan mengikuti alur cerita dari permainan tersebut sampai selesai. Terdapat berbagai macam variasi jenis dari RPG, diantaranya adalah WRPG, JRPG, ARPG, SRPG dan MMORPG (Stenström, 2012).

Pada Gambar 1.1 adalah representasi dari JRPG, yang mana pemain dapat memainkan satu karakter atau lebih. Karakter tersebut melakukan serangan secara bergantian atau *turn-based* antara pemain atau musuh, biasanya musuh yang dilawan berupa *Non-Player Character* (NPC). Bila ditarik garis besar dari jenis permainan RPG tersebut maka dapat digolongkan menjadi 2 yaitu *turn-based* dan *real-time*. Baru saja dijelaskan mengenai *turn-based* sedang untuk *real-time* adalah pertarungan antara pemain dengan musuh dilakukan secara langsung tanpa bergantian. Tipe RPG yang tergolong *turn-based* adalah TRPG dan sebagian JRPG, sedangkan yang tergolong ke dalam *real-time* adalah WRPG, ARPG, SRPG, MMORPG, dan sebagian JRPG.

Gambar 1.1: Ilustrasi *turned-based* JRPG

Dalam sebuah karakter permainan terdapat dua atribut yang berpengaruh yaitu atribut *gameplay* dan atribut estetika. Pada atribut *gameplay* biasanya bersifat non-visual yang menyajikan informasi tentang sebuah karakter dalam sebuah permainan. Sehingga setiap permainan memiliki tipe karakteristik dapat dideskripsikan. Contohnya pada salah satu papan permainan bergenre RPG yang terkenal yaitu *Dungeons and Dragons* (Heinsoo et al., 2008), yang memiliki enam atribut *gameplay* seperti *Strength*, *Constitution*, *Dexterity*, *Intelligence*, *Wisdom*, dan *Charisma*. Sedangkan atribut estetika adalah berupa tampilan dari karakter permainan yang meliputi penampilan, warna kulit, pakaian, dan lain sebagainya (Camelo et al., 2014). Atribut *gameplay* atau *stats* adalah istilah yang merujuk pada statistik dari fitur yang sesuai dengan karakter seperti ketepatan dalam menggunakan senjata atau keterampilan *magic*. Atribut *gameplay* tersebut berpengaruh pada efisiensi dan kekuatan karakter (Wenz, 2013).

Terdapat juga penjelasan yang menjelaskan tentang atribut *gameplay* atau *stats* dalam permainan adalah angka yang menjelaskan aspek entitas dalam dunia permainan. Entitas pada permainan bisa jadi berupa monster, karakter, senjata, atau mantra. Dalam RPG, biasanya atribut *gameplay* karakter pemain dapat ditingkatkan untuk menguntungkan pemain. Atribut *gameplay* juga digunakan untuk mensimulasikan pertempuran. Jika pemain menyerang karakter musuh, komputer membutuhkan cara untuk memutuskan apakah pemain memukulnya dan jika pemain melakukannya, seberapa banyak kerusakan yang ditimbulkan, dan terakhir perlu memeriksa apakah pukulan itu dapat membunuh musuh atau tidak. Dalam pembuatan keputusan ini, komputer menjalankan kode simulasi dengan menggunakan atribut *gameplay* (Schuller, 2017).

Pada permainan dengan genre RPG yang terdiri dari banyak karakter baik itu karakter pemain dan musuh. Di mana atribut *gameplay* tersebutlah yang akan menentukan jalannya pertarungan antara pemain dan musuh. Kemudian atribut *gameplay* juga menunjukan tingkat kesulitan dari musuh,

kelemahan dari musuh, daya tahan musuh terhadap serangan dan lain-lain. Hal ini juga merupakan bagian utama dari desain permainan, yang mana nominal pada atribut *gameplay*, jumlah pemain dan musuh dapat menentukan beberapa faktor penting lain seperti durasi permainan dan alur cerita.

Maka dari itu jenis permainan RPG dapat digolongkan juga menjadi 2 golongan lagi berdasar jumlah karakter yang dapat dimainkan oleh pemain yaitu *single-character* dan *multi-character*. Hal ini sangat berkaitan erat dengan penelitian ini saat dilakukannya pengujian, dalam pembuatan atribut *gameplay* untuk pemain harus dibedakan. Tipe RPG yang tergolong *multi-character* adalah TRPG dan JRPG, sedangkan yang tergolong dalam *single-character* adalah WRPG, ARPG, SRPG dan MMORPG.

Banyak penelitian yang mengaplikasikan berbagai bentuk algoritma dan metode untuk mempercepat pembuatan *video games* sepeti halnya pengukuran tingkat kesulitan dengan menggunakan *challenging rate* (CR) pada permainan 2D *Real Time Strategy* (RTS) (Christyowidiasmoro et al., 2016), pembuatan *game engine* yang berorientasi pada *multi-agent systems* (Marín-Lora et al., 2020), pembuatan level secara otomatis pada permainan secara prosedular dengan tingkat kesulitan tertentu (Wu et al., 2018), pemetaan atribut *gameplay* untuk menentukan estetika karakter secara otomatis dengan menggunakan algoritma fuzzy (Camelo et al., 2014), pencapaian keseimbangan dalam permainan *Real Time Strategy* (RTS) dengan pembuatan sebuah *framework* yang dinamakan dengan *attribute space* untuk evaluasi yang direpresentasikan dalam pertarungan *single unit* atau satu lawan satu pada permainan RTS (Bangay and Makin, 2014). Berdasarkan beberapa penelitian tersebutlah maka penelitian ini dibuat, dengan tujuan memudahkan desainer permainan dalam mendesain sebuah permainan dengan genre RPG.

Penggunaan berbagai metode seperti $k-$NN, Distribusi Normal dan Naive bayes yang kemudian dilanjutkan dengan diperolehnya atribut *gameplay* untuk pemain dan musuh yang siap digunakan. Kemudian dilanjutkan dengan proses klasifikasi dari atribut *gameplay* tersebut apakah sudah sesuai dengan

apa yang direncanakan. Pada proses tersebut digunakanlah *Neural Network Multiclass Classification*.

1.2 Rumusan Masalah

Pembuatan atribut *gameplay* untuk karakter yang akan dimainkan oleh pemain dan karakter musuh pada permainan dengan genre RPG yang terkadang masih dilakukan secara manual, terlebih lagi jika karakter dari pemain dan musuh pada permainan tersebut sangat banyak maka diperlukannya sebuah program yang mampu menyusun hal tersebut secara otomatis.

1.3 Tujuan

Dari permasalahan yang dirumuskan sebelumnya,maka penelitian ini memiliki tujuan sebagai berikut:

1. Mempercepat proses desain permainan dengan menggunakan program yang dapat menghasilkan atribut *gameplay* yang berupa data statistik untuk karakter dari pemain dan musuh.

2. Diperolehnya atribut *gameplay* untuk karakter pemain dan musuh yang dihasilkan secara otomatis oleh program dan terklasifikasi bedasarkan tipe yang ingin dibuat.

1.4 Batasan Masalah

Berdasrkan fokus permasalahan pada bagian sebelumnyaa, kemudian diambilah beberapa pembatasa masalah. Berikut adalah batasan-batasan masalah tersebut.

1. Atribut *gameplay* yang dihasilkan oleh program hanya dapat dipakai oleh permainan dengan genre RPG.

2. Program yang dibuat untuk menghasilkan atribut *gameplay* dapat dipakai oleh lebih sama dengan satu karakter pemain dan musuh.

3. Metode yang digunakan dalam proses perhitungan atribut *gameplay* pemain dan musuh diantaranya adalah $k-$NN, Distribusi Normal dan Naive Bayes.

4. Perhitungan tingkat kesesuaian atribut *gameplay* dari karakter pemain dan musuh yang telah dihasilkan, dengan cara klasifikasi menggunakan *Neural Network Multiclass Classification*.

1.5 Kontribusi

Di harapkan penelitian ini mampu mejadi rujukan dalam proses desain permainan, khususnya saat pembuatan atribut *gameplay* untuk karakter pemain dan musuh. Selain itu menjadikan proses pengembangan permainan dengan genre RPG di masa mendatang menjadi lebih cepat.

Halaman ini sengaja dikosongkan

BAB 2

KAJIAN PUSTAKA

Demi mendukung penelitian ini, dibutuhkan beberapa teori penunjang sebagai bahan acuan dan referensi. Berikut adalah paparan teori dasar meliputi komponen desain permainan, *machine learning*, penjelasan permainan genre RPG, dan Artificial Neural Network.

2.1 Kajian penelitian terkait

Beberapa penelitan sebelumnya yang berkaitan, menjadi acuan dan pendukung dalam penelitian ini diantaranya adalah sebagai berikut:

1. Publikasi dengan judul "*A game engine to make games as multi-agent systems*" yang ditulis oleh Carlos Marín-Lora dan rekan-rekan tahun 2020. Pada tulisan ini meneliti tentang pengembangan *game engine* yang berfokus pada multi-agen. Di mana pada setiap aktor atau agen-agen memiliki sejumlah properti dan aturan kebiasaan untuk berinteraksi dengan lingkungan dalam permainan. Tujuan dari *engine* tersebut adalah memenuhi kebutuhan dasar dari sistem multi-agen dengan merubah karakteristik dari sistem tanpa adanya pengaruh terhadap potensi karakter.

2. Publikasi dengan judul "*Procedurally Generating Game Level with Specified Difficulty*" yang ditulis oleh Zong-Han Wu dan rekan-rekan tahun 2018. Penelitian ini mencakup ranah pembuatan konten dalam sebuah permainan secara prosedural, hal tersebut bertujuan agar para pemain memperoleh pengalaman yang berbeda dalam bermmain. Pada paper ini yang paling banyak dibahas adalah permainan PAC-MAN, dengan labirin yang dapat berubah secara prosedural.

3. Publikasi dengan judul "*Measuring Level of Difficulty in Game Using*

Challenging Rate (CR) on 2D Real Time Strategy Line Defense Game" yang ditulis oleh Christyowidiasmoro dan rekan-rekan tahun 2015. Meneliti tentang formula *Challenging Rate* (CR) adalah formula yang menggunakan elemen dasar dalam permainan RTS (*Real Time Strategy*) sebagai parameternya. Dalam tulisan ini dibuat sebuah *prototipe* permainan dengan tingkat kesulitan yang diajukan untuk mencoba kemampuan formula *Challenging Rate* untuk mengukur nilai tingkat kesulitan.

4. Publikasi dengan judul "*Automatic Mapping Between Gameplay And Aesthetics RPG Character Attributes Through Fuzzy System*" yang ditulis oleh Jefferson Henrique Camelo dan rekan-rekan tahun 2014. Meneliti tentang pemetaan atribut *gameplay* untuk menentukan estetika karakter secara otomatis dengan menggunakan algoritma fuzzy, dalam pemilihan atribut *gameplay* yang dilakukan oleh pemain.

5. Publikasi dengan judul "*Generating an Attribute Space for analyzing Balance in Single Unit RTS Game Combat*" yang ditulis oleh Shaun Bangay dan Owen Makin tahun 2014. Fokus pada tulisan ini adalah tentang pencapaian keseimbangan dalam permainan *Real Time Strategy* (RTS), dengan dibuatnya sebuah *framework* yang dinamakan dengan *attribute space*. Pendekatan ini dievaluasi dan direpresentasikan untuk pertarungan *single unit* atau satu lawan satu pada permainan RTS. Melalui proses tersebut kemudian dibuatlah model prediksi terhadap simulasi pertarungan tersebut, dengan menggunakan atribut atau data statistik pada permainan tersebut seperti halnya *speed*, *range*, *health* dan *damage*.

2.2 Desain Permainan RPG dan Komponennya

Desain permainan adalah proses menciptakan konten dan aturan permainan. Desain permainan yang baik adalah proses menciptakan tujuan agar seorang pemain merasa termotivasi untuk mencapainya dan aturan yang dibuat dapat diikuti oleh seorang pemain saat membuat keputusan untuk mengejar

tujuan-tujuan tersebut (Brathwaite and Schreiber, 2009). Berikut adalah sebagian elemen dari desain permainan yang akan digunakan pada permainan dengan genre RPG pada penelitian ini.

2.2.1 Penempatan Peluang pada Permainan

Sebagian besar permainan setidaknya mengandung beberapa faktor acak. Misalnya, banyak permainan kartu melibatkan proses acak. Adanya variabel *damage* dipertarungan pada sebagian besar permainan dengan genre RPG. Permainan seperti *Rock-Paper-Scissors* memiliki pola yang tampak acak, meskipun terlihat tanpa mekanisme acak. Pada bagian ini akan banyak membahas mengenai aspek acak pada permainan dan bagaimana *developer* akan menggunakannya.

Pada dasarnya permainan yang memasukan unsur keberuntungan dapat dimainkan dan dimenangkan oleh khalayak luas. Bila dicari alasan penggunaannya, hasilnya sangatlah banyak. Pada akhirnya harus diakui bahwa hal-hal tersebut adalah keputusan dari desainer permaninan untuk menciptakan dinamika yang dikehendakinya. Berikut adalah contoh-contoh penerapan penempatan peluang pada permainan yang biasanya diterapkan.

a). Menunda atau Mencegah Penyelesaian Masalah

Dalam permainan yang memiliki sedikit cara penyelesaian saat dipecahkan oleh komputer dan terutama manusia. Maka sesuatu harus dilakukan untuk menjaga permainan agar tetap *fresh* dan menyenangkan. Maka ditambahkan elemen acak dengan tujuan agar pemain tidak terlalu menguasai permainan, karena saat pemain membuat keputusan yang sama persis dapat memberi hasil yang berbeda. Hal tersebut juga berpengaruh pada tingkat rasa bosan pemain dalam memainkan sebuah permainan, jika pola yang sama terus muncul.

b). Membuat Permainan Lebih Kompetitif

Elemen acak yang sesekali memungkinkan pemain yang kurang berpe-

ngalaman untuk menang atau setidaknya menawarkan keuntungan yang membuat permainan ini bertahan lebih lama dengan menggunakan dua cara. Pertama adalah selalu ada peluang kemenangan bagi setiap pemain. Kemudian yang kedua adalah efek dari kekalahan berkurang ketika seorang pemain bisa menyalahkan nasib buruknya sendiri atas kekalahannya. Hal tersebut tentunya juga dapat memunculkan rasa tidak terima akan sebuah kekalahan dan dimulainya permainan baru lagi.

c). Meningkatkan Keberagaman

Permainan tanpa elemen acak selalu dimulai dengan cara yang sama dan pola-pola tertentu seperti langkah awal pada permainan Catur. Akibatnya, pemain dapat memiliki pengalaman yang sama dari satu pertandingan ke pertandingan yang lain, dan para pemain dapat mejatuhkan pilihannya untuk selalu memakai strategi yang sama.

d). Menciptakan Momen Dramatis

Ketika seorang pemain dengan hati-hati menyusun strategi dan kemudian harus bergantung pada lemparan dadu atau apa pun yang bersifat acak untuk melihat apakah rencana tersebut berhasil, momen tersebut bisa menjadi sangat menegangkan. *Role-Playing Games* (RPG), *Real-Time Strategy* (RTS) dan banyak *board game* yang mengandalkan kondisi ini saat bermain. Akankah *healing spell* (mantra penyembuhan) yang digunakan akan sampai ke rekan bermainmu yang terluka sebelum monster menyerang? Apakah AI akan memberi pemain dua monster untuk bertempur ataukah hanya satu?

e). Meningkatkan Pengambilan Keputusan

Inti dari sebagian besar permainan adalah keputusan yang dibuat para pemain. Dalam permainan strategi, pemain memiliki informasi lengkap dan tahu hasil pasti dari setiap gerakan yang mereka lakukan. Karena semua variabel diketahui, beberapa keputusan menjadi tidak terlalu

menarik, jika ada kesempatan untuk membunuh ratu lawan pada permainan catur secara cuma-cuma tanpa ada yang perlu dikorbankan, hal tersebut bukan merupakan keputusan yang menarik karena adanya jawaban "benar" yang jelas. Ketika elemen acak ada dalam permainan, tidak ada lagi strategi yang selalu benar. Beberapa gerakan mungkin memiliki peluang kegagalan yang tinggi tetapi juga potensi hasil yang besar, hal tersebut menjadikannya sebuah pilihan berisiko, gerakan lainnya mungkin aman tetapi menghasilkan sedikit keuntungan. Hal tersebut menjadikan pemain akan melakukan analisis gerakan dengan cara yang berbeda berikut risiko dan keutungan yang akan didapat, kemudian tidak lupa mempertimbangkan posisi pemain tersebut dalam permainan atau langkah selanjutnya.

2.2.2 Elemen Keterampilan Strategi

Strategi adalah salah satu kekuatan yang dapat dilakukan oleh para pemain. Karena hal tersebutlah yang membuat para pemain tetap bermain. Pemain bermain *game* dan menikmati setiap prosesnya karena mereka berusaha untuk menguasai pola dalamnya (Koster, 2013). Saat pola berhasil dikuasai, maka selanjutnya pemain akan terus bersenang-senang. Mereka juga membentuk strategi berdasarkan pemahaman mereka tentang dinamika permainan. Penguasaan, pengembangan strategi, hal tersebut terjadi bukan karena ketidak sengajaan.

Hal ini adalah tujuan dari desainer permainan atas penggunaan mekanika untuk menciptakan strategi dan taktik dalam permainan yang mereka buat (Brathwaite and Schreiber, 2009). Berikut beberapa poin penting pada penelitian ini yang mengacu tentang pentingnya hal tersebut.

a). Peran dari Keterampilan pada Permainan

Permainan yang bagus dibangun dari serangkaian keputusan menarik, membangun unit untuk menyerang atau bertahan, mencari tahu

apa yang harus dilakukan oleh unit tersebut selanjutnya, apa saja kelemahannya dan lain sebagainya. Keberhasilan keputusan tersebut juga didasari oleh reaksi dari mental atau fisik pemain, karena hal tersebut adalah ukuran keterampilan dari seorang pemain.

Permainan yang bagus menyebabkan pemain sering melatih kemampuannya dan memberi *reward* atau bonus dengan pola timbal balik yang jelas. Kemudian sepanjang permainan pemain bertanya-tanya tentang apa yang harus dilakukan selanjutnya. Masuklah mereka ke dalam istilah yang disebut dengan “lingkaran sihir" sebuah *video game*, melalui monitor atau TV sebagai medianya terseraplah mereke ke dalam dunia game. Efek tersebut sama seperti ketika menonton film atau membaca buku yang sangat menarik, tetapi permainan yang baik memiliki daya tarik yang lebih kuat karena terjadi interaksi antara pemain dan keputusan yang dibuat yang terkonversi menjadi sebuah pengalaman.

Ketika pemain terus-menerus membuat keputusan, secara tidak sadar mereka telah memasuki kondisi yang oleh psikolog dan peneliti terkenal Mihaly Csikszentmihalyi disebut “*flow*". Hal tersebut adalah keadaan permainan yang optimal dan merupakan hasil kerja keras dari seorang desainer pemainan. Csikszentmihalyi menulis seluruh buku tentang topik, *Flow: The Psychology of Optimal Experience.* Buku ini tidak terbatas pada permainan saja, tetapi mencakup keadaan semacam ini dari perspektif yang lebih luas.

b). Strategi dan Taktik

Secara teknis strategi utama adalah keseluruhan cara untuk mencapai tujuan akhir jangka panjang (misalnya, kemenangan dalam permainan). Strategi utama terdiri dari beberapa strategi pendukung dengan tujuan jangka pendek atau menengah yang harus dilakukan untuk mencapai strategi besar (misalkan dalam peperangan besar, pilihan untuk bertarung dalam pertempuran tertentu adalah sebuah pilihan strategis).

Taktik adalah keputusan mikro tingkat terendah yang dibuat ketika menjalankan strategi misalkan pada pergerakan pasukan, apakah akan diperintahkan untuk melakukan serangan udara, kapan pasukan haarus mulai bergerak, dimana posis yang tepat untuk menempatkannya, kapan mulai menembak dan lain-lain adalah contoh keputusan taktis yang dibuat selama pertempuran militer. Secara informal pemain dalam permainan membuat keputusan strategis ketika membuat rencana jangka panjang (panjangnya strategi tersebut bersifat relatif terhadap panjangnya permainan), dan keputusan taktis ketika pemain ingin mencapai tujuan jangka pendek.

Pengorbanan mejadikan pengambilan keputusan atau pembuatan taktik menjadi lebih menarik. Keputusan yang diambil secara cepat atau *twitch mechanics*, bisa disebut juga dengan ketangkasan memiliki keterbatasan pada taktik. Hal ini menunjukkan bahwa permainan yang lebih fokus pada strategi, umumnya bersifat giliran atau *turn-based* seperti Catur dan *Go*, yang mana lebih terfokus pada keputusan yang melibatkan pengorbanan.

c). Game Berbasis Keterampilan Sepenuhnya

Permainan yang fokus pada strategi dan pengorbanan cenderung memiliki setidaknya beberapa elemen peluang. Ketika permainan-permainan ini murni berbasis keterampilan, seperti *Tic-Tac-Toe* atau sebagian besar *video game* petualangan, mereka dapat diselesaikan, dan keputusan-keputusan yang tadinya bervariasi kemudian menjadi keputusan-keputusan yang jelas. Misalkan dalam membuat keputusan pada akhirnya hanya ada satu gerakan atau pilihan benar.

Sebagian besar game yang sepenuhnya berupa keterampilan adalah game aksi berbasis fisik. Mungkin hal inilah sebabnya, tidak seperti pada pengorbanan yang bukan tentang mendapatkan jawaban yang benar atau terbaik melainkan mendapatkannya dengan cepat. Waktu reaksi

atau tindakan manusia dapat terus meningkat dari waktu ke waktu selamanya, terutama pada permainan dimana manusia saling berlawanan satu dengan yang lain atau *multiplayer*.

2.2.3 Menemukan Keseimbangan

Beberapa permainan sangat terlihat jelas bahwa semua membutuhkan keterampilan. Hal ini sangat bervariasi dari yang sederhana (*Tic-Tac-Toe*) menuju yang lebih kompleks (Catur dan *Go*). Ada juga perbedaan antara keterampilan berbasis strategis (seperti pada kebanyakan permainan berbasis strategi dan giliran) dan kemampuan pengambilan keputusan secara cepat atau *twitch* (biasa ditemukan dalam permainan berbasis keterampilan dan olahraga) atau bisa disebut juga dengan ketangkasan. Permainan ini bisa menjadi sangat menyenangkan karena terdapat keputusan berarti yang dibuat oleh pemain, baik itu keputusan yang dibuat dengan cepat (*twitch*) dan keputusan yang dibuat berdasarkan strategi (Brathwaite and Schreiber, 2009).

Banyak permaian yang menggabungkan kedua basis di atas. *Settlers of Catan* memiliki banyak elemen *gameplay* berbasis keterampilan, termasuk perdagangan, pembangunan, dan manajemen sumber daya. Namun permainan tersebut juga memiliki dadu, setumpuk kartu yang dikocok secara acak, dan pengaturan papan yang disusun dengan acak. *Backgammon* dan *Poker* keduanya memiliki elemen keterampilan dan keberuntungan yang kuat. Bayangkan sebuah permainan *Backgammon* tanpa dadu, atau *Poker* tanpa kemampuan untuk bertaruh atau menggertak lawan, dan menjadi jelas bahwa beberapa permainan mampu menjadi lebih baik dengan adanya campuran keterampilan dan peluang. Jika salah satunya dihapus maka permainan tersebut tidak akan menjadi menarik.

Bagaimana seorang desainer permainan memasukkan campuran keterampilan dan peluang yang benar dalam sebuah permaian? Kapan saat yang tepat untuk mengarahkan permainan ke arah itu? Jawabannya adalah kembali kepada siapakah pemainnya?

Hal tersebut adalah pertanyaan penting, seringkali pertanyaan pertama yang diajukan seorang desainer. Ketika merancang permainan untuk anak berusia enam tahun hasil yang diperoleh akan sangat berbeda jika dibandingkan dengan merancang permainan yang diperuntukkan bagi mahasiswa. Pemain yang berbeda memiliki tingkat toleransi yang berbeda untuk setiap peluang dan keterampilannya. Permainan yang mungkin menjadi sangat populer bagi para gadis muda bisa jadi menjadi membosankan bagi orang dewasa, karena sebagai orang tua yang telah memainkan sejumlah permainan anak-anak dan dapat mengujinya. Beberapa contoh target pemain seperti anak-anak, pemain permainan kompetitif, permainan sosial, pemain profesional dan keluarga. Dalam penelitian ini akan difokuskan dengan permainan kompetitif, karena fokus utamanya adalah *fun games*.

Pemain permainan kompetitif cenderung lebih menyukai permainan dengan lebih banyak elemen keterampilan. Hal ini memberi mereka kesempatan untuk bermain satu lawan satu dengan pemain lain, mencocokan setiap pemikiran menjadi sebuah strategi, refleks melawan refleks, strategi melawan strategi. Hal terakhir yang diinginkan oleh pemain yang benar-benar kompetitif adalah proses acak seperti halnya dengan penggelindingan dadu yang merusak permainan yang dieksekusi dengan sempurna dan terampil.

Mengapa desainer menambah keberuntungan pada permainan berbasis keterampilan? Banyak alasan yang membuat hal-hal tidak dapat diprediksi, meningkatkan kemampuan pemain dengan memainkan permainan itu lagi, dan memungkinkan pemain dengan keterampilan yang sedikit berbeda untuk tetap dapat bersaing. Jika pemain yang lebih baik selalu menang, maka satu-satunya pertarungan yang menarik adalah antara dua pemain dengan kemampaun yang sama rata.

2.3 Machine Learning

Machine Learning (ML) adalah studi ilmiah tentang algoritma dan model statistik yang digunakan oleh komputer untuk melakukan tugas tertentu seca-

ra efektif tanpa menggunakan instruksi secara eksplisit, dengan mengandalkan pola dan inferensi sebagai gantinya. *Machine learning* juga dilihat sebagai bagian dari kecerdasan buatan. Algoritma dari *machine learning* membangun model matematika berdasarkan data sampel, yang dikenal sebagai "data training" dalam membuat prediksi atau keputusan tanpa diprogram secara eksplisit untuk melakukan tugasnya (Koza et al., 1996). Kemudian setelah dilakukannya *training* maka dapat digunakan untuk menyelesaikan permasalahan dari tujuan dibutnya *machine learning* itu sendiri, atau yang biasa disebut dengan proses *testing*.

Algoritma *machine learning* dapat digunakan pada berbagai aplikasi, seperti penyaringan email, dan visi komputer, yang mana tidak mungkin untuk mengembangkan algoritma instruksi khusus untuk melakukan tugas dengan sendirinya. *Machine learning* terkait erat dengan komputasi statistik yang berfokus pada pembuatan prediksi dengan menggunakan komputer. Studi tentang optimasi matematika memberikan metode, teori dan domain aplikasi ke bidang *machine learning*. Penambangan data atau *data mining* adalah bidang studi yang termasuk ke dalam cakupan *machine learning*, dan berfokus pada analisis dan eksplorasi data melalui pembelajaran yang mampu berjalan secara otomatis (Friedman, 1997).

Pada penelitian ini akan digunakan beberapa algoritma *machine learning* untuk menyelesaikan permasalahan dalam mendesain permainan. Penggunaan algoritma tersebut secara umum adalah membuat data baru yang terstruktur dengan referensi data awal yang disesuaikan dengan kebutuhan untuk mendesain permaian. Berikut algoritma-algoritma tersebut dan penjelasannya secara umum nantinya, di mana natinya akan dipakai dan dijelaskan secara detail pada BAB 3.

2.3.1 K-Nearest Neighbor

Di asumsikan terdapat sampel (x_i, θ_i) yang didistribusikan secara sesuai dengan distribusi (x, θ) argumen heuristik tertentu yang memungkinkan da-

lam prosedur pengambilan keputusan. Contohnya saat pengasumsian tentang jarak dari beberapa data, metrik atau koordinat yang sesuai akan diklasifikasi berdasarkan kesamaannya atau setidaknya memiliki kesamaan distribusi pada setiap datanya. Maka dalam klasifikasi sampel x, dipertimbangkan berdasarkan x_i terdekat yang diasumsikan sebagai kemungkinan terbesar. Sehingga prosedur pengambilan keputusan paling sederhana adalah aturan *nearest neighbor* (NN) dengan mengklasifikasikan x sebagai tetangga terdekat.

Di buatlah satu set n pasangan $(x_1, \theta_2), ..., (x_n, \theta_n)$, yang mana x_i's mengambil nilai dalam metrik X yang didefinisikan sebagai metrik d, dan O_i nilai yang diambil di set $\{1, 2, ..., M\}$ seperti pada persamaan 2.1 dan 2.2. Setiap θ_i dianggap sebagai indeks dari kategori yang dimiliki oleh data ke-i, dan setiap x_i adalah hasil dari pengukuran setiap data. Lebih singkatnya pada "x_i bagian dari θ_i" adalah saat data ke-i yang menjadi dasar pengukuran x_i yang telah diamati dan diukur, termasuk juga pada kategori θ_i.

Di lakukan beberapa penambahan pasangan baru (x, θ), yang mana pengukuran x yang akan dilakukan, dan digunakan hasilnya digunakan untuk memperkirakan θ dengan memanfaatkan hasil pengukuran atau klasifikasi sebelumnya. Pada persamaan 2.1 dinyatakan bahwa seluruh data x atau x_n yang akan diguanakan untuk mecari nilai θ.

$$x'_n \in x_1, x_2, \ ..., \ x_n \tag{2.1}$$

Akan termasuk ke dalam *nearest neighbor* terhadap x jika dilanjutkan dengan persammaan berikut.

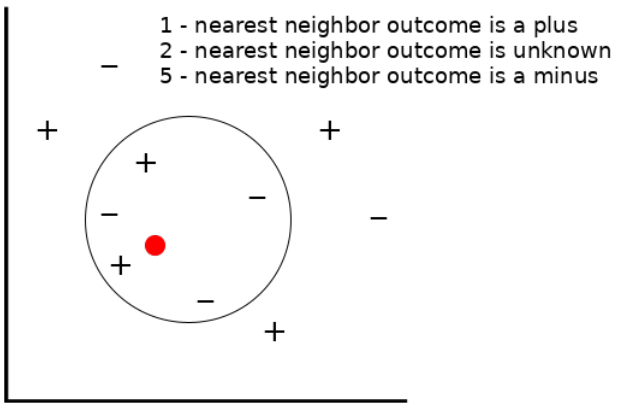
$$min \ d(x_i, x) = d(x'_n, x), \quad i = 1, 2, \ ..., \ n \tag{2.2}$$

Aturan *nearest neighbor* menentukan, apakah x termasuk dalam kategori θ'_n dari tetangga terdekatnya x'_n seperti pada persamaan 2.2 di mana metrik d adalah *distance* atau jarak. Kesalahan dibuat jika $\theta'_n \neq \theta$. Perhatikan bahwa aturan *nearest neighbor* hanya menggunakan klasifikasi berdasarkan tetangga

terdekat saja. Sedangkan $n-1$ yang merupakan sisa hasil klasifikasi θ_i akan diabaikan (Cover and Hart, 1967). Dengan demikian, metode klasifikasi berbasis $k-$NN dapat dengan mudah diterapkan dalam banyak tugas klasifikasi. Secara umum, sebagian besar varian $k-$NN menentukan label kelas dari sampel data dengan menggunakan satu parameter k yang merupakan jarak dari keseluruhan data yang di anggap sebagai *neighbour* dan juga keputusan di ambil dengan mempertimbangkan jarak data secara mayoritas. Tetapi keputusan klasifikasi tersebut dapat dengan mudah dipengaruhi oleh sensitivitas k itu sendiri dan juga dapat memperburuk sensitivitasnya, sehingga sangat menurunkan kinerja klasifikasi dari $k-$NN terutama dalam kasus ukuran sampel yang kecil.

a). Klasifikasi

Dalam mendemonstrasikan analisis $k-$*nearest neighbor*, dipertimbangkan juga proses klasifikasi objek baru (koordinat data) diantara sejumlah contoh yang diketahui. Hal ini ditunjukkan pada Gambar 2.1, yang menggambarkan representasi dari data atau *instance* dengan tanda plus dan minus dan titik dengan lingkaran merah. Dan permasalahan yg harus diselesaikan adalah memperkirakan (klasifikasi) hasil dari titik tersebut berdasarkan sejumlah tetangga terdekat atau *nearest neighbor* yang dipilih. Dengan kata lain, apakah titik tersebut dapat diklasifikasikan sebagai tanda plus atau minus.



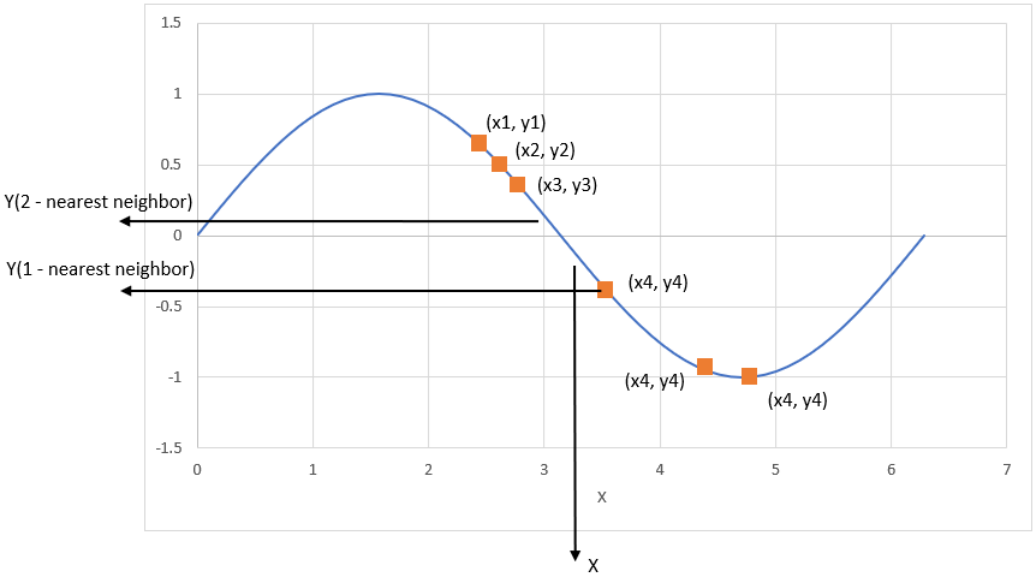
Gambar 2.1: Klasifikasi dengan $k-$NN.

Kemudian dipertimbangkannya hasil $k-$NN berdasarkan tetangga terdekat pertama (1-*nearest neighbor*). Jelas bahwa dalam hal ini $k-$NN akan memprediksi hasil dari titik atau representsi data dengan nilai tambah (karena titik terdekat adalah tanda plus). Dilanjutkan dengan penambahan jumlah tetangga terdekat menjadi 2 (2-*nearest neighbor*). Kali ini $k-$NN tidak akan dapat mengklasifikasikan hasil dari titik dari data yang dicari, karena titik terdekat keduanya adalah minus, sehingga tanda plus dan minus mencapai jarak yang sama. Untuk langkah selanjutnya, ditambahkan lagi jumlah tetangga terdekat menjadi 5 (5-*nearest neighbor*). Hal ini akan menentukan daerah tetangga terdekat, yang ditunjukkan oleh lingkaran yang seperti pada Gambar 2.1. Karena ada 2 tanda plus dan 3 tanda minus. Pada lingkaran tersebut $k-$NN akan memberikan tanda minus pada hasil dari titik koordinat yang dicari.

b). Regresi

Pada bagian ini akan digeneralisasi konsep $k-$*nearest neighbor* untuk menyelesaikan permasalahan regresi. Permasalahan regresi sangat berkaitan dengan prediksi hasil dari variabel dependen berdasarkan beberapa variabel independen. Di mulai dengan mempertimbangkan skema yang ditunjukkan pada Gambar 2.2, di mana satu data *training* atau data sampel (kotak jingga) yang diambil dari hubungan antara variabel independen x dan variabel dependen y (kurva biru).

Kemudian ditandai dengan kotak jingga yang dlianjutkan dengan penerapan $k-$*nearest neighbor* untuk memprediksi hasil keluaran yang ditandai dengan X. Kemudian direpresentasikan juga dengan kotak jingga. Di mulai dengan mempertimbangkan metode 1-*nearest neighbor* sebagai data sampel. Dalam kasus ini akan dicari data sample (kotak jingga) yang paling dekat dengan titik X. Pada Gambar 2.2 yang dicari dinyatakan dalam koordinat (X, Y) sedangkan koordinat data yang paling mendekati adalah (x_4, y_4). Maka dapat disimpulkan bahawa tetangga terdekatnya adalah (x_4, y_4) untuk 1-*nearest neighbor*.



Gambar 2.2: Proses regresi dengan $k-$NN.

Langkah selanjutnya adalah mencari 2-*nearest neighbor* seperti pada Gambar 2.2. Pada kasus ini ditemukan dua titik terdekat dengan X atau 2-*nearest neighbor*, yang kebetulan ditunjukan pada Gambar 2.2 adalah $y3$ dan $y4$. Maka diambil rata-rata dari masing-masing nilai pada data tersebut yang dinyatkan pada persamaan 2.3.

$$Y = \frac{y_3 + y_4}{2} \tag{2.3}$$

Pada metode $k-$*nearest neighbor* hasil Y dari titik X dianggap sebagai hasil rata-rata dari nilai $k-$*nearest neighbor* dari tetangga terdekatnya.

c). Cross-Validation

Cross-validation adalah teknik yang dapat digunakan untuk mendapatkan estimasi model parameter yang tidak diketahui. Pembahasan ini bertujuan untuk menerapan teknik dalam memperkirakan nilai k.

Gagasan umum dari metode ini adalah untuk membagi sampel data ke dalam sejumlah kelompok atau *cluster* (diambil secara acak, *sub-*

samples atau bagian-bagian yang terpisah). Untuk nilai tetap pada k, digunakanlah $k-$NN untuk membuat prediksi pada bagian ke v (mislahnya dengan penggunaan bagian $v-1$ sebagai contoh) dan kemudian digunakan untuk evaluasi kesalahan. Metode paling umum untuk analsia kesalahan pada regresi adalah dengan jumlah kuadrat atau *sum of square* dan pada klasifikasi hal tersebut didefinisikan sebagai akurasi atau tingkat kebenaran klasifikasi.

Proses ini kemudian diterapkan secara berurutan pada semua bagian atau v yang mungkin. Pada bagian terakhir, kesalahan yang dijumlahkan kemudian dirata-rata untuk menghasilkan model yang stabil (seberapa baik model memprediksi titik koordinat data). Langkah-langkah di atas kemudian diulang untuk berbagai nilai k yang lain, nilai yang mencapai kesalahan terendah atau akurasi klasifikasi tertinggi dijadikan sebagai nilai optimal untuk k.

Usaha dalam mencapai nilai optimal inilah yang kemudian disebut dengan *cross-validation*. Perlu diperhatikan bahwa *cross-validation* membutuhkan proses komputasi tinggi dan algoritma harus dibiarkan berjalan untuk beberapa waktu terutama saat ukuran sampel berjumlah besar. Atau nilai k dapat ditentukan sendiri. Ini mungkin tindakan yang wajar jika sudah megetahui apa yang akan diambil sebagai nilai k, dari analisis $k-$NN sebelumnya yang mungkin dilakukan pada data yang sama atau pengamatan secara manual.

d). Jarak Metrik

Seperti disebutkan sebelumnya, pada titik data yang dicari, $k-$NN membuat prediksi berdasarkan hasil k dari tetangga terdekat dengan titik itu. Oleh karena itu, untuk membuat prediksi dengan $k-$NN, sangat diperlukan pendefinisian metrik untuk mengukur jarak antara titik koordinat data dan kasus dari data sampel.

Salah satu metode paling populer untuk mengukur jarak ini dikenal dengan istilah *Euclidean*. Langkah-langkah lain diantaranya adalah *Euclidean squared*, *City-block*, dan *Chebyshev* seperti yang ditunjukan pada persamaan 2.4.

$$D(x,\ p) = \left\{ \begin{array}{cc} \sqrt{(x-p)^2} & Eclidean \\ (x-p)^2 & Eclideansquared \\ abs(x-p) & Cityblock \\ Max(|x-p|) & Chebyshev \end{array} \right\} \tag{2.4}$$

Di mana pada persamaan 2.4 variabel x dan p masing-masing adalah titik koordinat data yang dicari dan data sampel. Kemudian yand dimasud dengan D adalah *distance* atau jarak antara data sampel dengan koordinat data yang dicari.

e). Prediksi

Setelah memilih nilai k, maka prediksi berdasarkan data sampel dengan menggunakan $k-$NN dapat dilakukan. Sedangkan pada regresi sepeti yang dijelaskan pada bagian sebelumnya, $k-$NN memprediksi hasil rata-rata dari $k-$*nearest neighbor*.

$$y_{pred} = \frac{1}{k} \sum_{i=1}^{k} y_i \tag{2.5}$$

Di mana y_i adalah kejadian ke-i pada data sampel dan y_{pred} adalah hasil prediksi dari titik koordinat data yang dicari. Berbeda dengan regresi, dalam masalah klasifikasi, prediksi $k-$NN didasarkan pada skema *voting* di mana pemenanglah yang akan digunakan untuk melabeli titik koordinat data yang dicari.

Untuk klasifikasi biasanya nilai ganjil seperti $y_{pred} = 1, 3, 5, dan$ *seterusnya* sering digunakan untuk menghindari terjadinya *ties condi-*

tion, yaitu kondisi dimana terdapat dua label kelas yang mencapai skor yang sama.

f). Pembobotan Jarak

Karena prediksi $k-$NN didasarkan pada asumsi intuitif bahwa objek yang jaraknya dekat berpotensi dianggap sama. Hal tersebut sangat masuk akal dalam hal membedakan antara k pada tetangga-tetangga terdekat ketika membuat prediksi.

Dengan membiarkan titik terdekat antara k pada setiap tetangga terdekat maka semakin banyak jarak yang diperoleh dan akan mempengaruhi hasil dari titik data yang ingin diolah. Hal ini dapat dicapai dengan menerapkan bobot atau W pada setiap tetangga terdekat, yang ditentukan oleh jarak masing-masing tetangga dengan titik data yang ingin diolah seperti pada persamaan 2.6.

$$W(x, p_i) = \frac{exp(-D(x, p_i))}{\sum_{i=1}^{k} exp(-D(x, p_i))} \tag{2.6}$$

Di mana $D(x, pi)$ adalah jarak antara titik data yang akan diolah, sementara x dan pi adalah data sampel ke-i. Hal tersebut sudah mejelaskan bahwa bobot yang ditentukan dengan persamaan 2.6 akan dapat dipenuhi atau dibuktikan dengan persamaan 2.7.

$$\sum_{i=1}^{k} W(x_0, x_i) = 1 \tag{2.7}$$

Sedangkan untuk permasalahan regresi dapat dipenuhi atau dibuktikan dengan persamaan 2.8.

$$y = \sum_{i=1}^{k} W(x_0, x_i)\ y_i \tag{2.8}$$

Kemudian untuk permasalah klasifikasi, persamaan 2.7 dapat diambil untuk klasifikasi setiap variabelnya. Sudah jelas bahwa dari pemm-

bahasan ini, ketika $k > 1$, maka seseorang dapat secara langsung menentukan standar deviasi untuk prediksi pada regresi dengan menggunakan persamaan 2.9.

$$error\ bar = \mp \sqrt{\frac{1}{k-1}} \sum_{i=1}^{k} (y - y_1)^2 \tag{2.9}$$

2.3.2 Naive Bayes

Pada *machine learning* sering digunakannya hipotesis terbaik (h) dari data yang akan diproses (d). Dalam klasifikasi dengan menggunakan Naive Bayes, hipotesis (h) mungkin dapat dijadikan sebagai kelas untuk mengklasifikasi data baru (d).

Salah satu cara termudah untuk memilih hipotesis yang memungkinkan adalah dengan menggunkan data yang sudah ada. Kemudian digunakan sebagai referensi untuk menyelesaikan masalah tersebut. Teorema Bayes memberikan cara untuk menghitung probabilitas berdasarkan hipotesis yang diberikan berdasarkan pemahaman terhadap data yang ingin diolah. Maka teorema bayes dapaat dinyatakan seperti pada persamaan 2.10

$$P(h|d) = \frac{(P(d|h) \times P(h))}{P(d)} \tag{2.10}$$

Di mana pada persamaan 2.10 akan dijelaskan menjadi beberapa poin, berikut poin-poin tersebut.

1. $P(h|d)$ adalah probabilitas hipotesis h berdasrkan data d. Hal ini disebut probabilitas posterior.

2. $P(d|h)$ adalah probabilitas dari data d berdasarkan hipotesis h yang bernilai benar.

3. $P(h)$ adalah probabilitas hipotesis h yang bernilai benar terlepas dari data secara keseluruhan.

4. $P(d)$ adalah probabilitas data secara keseluruhan terlepas dari hipotesis.

Dapat dilihat bahwa dalam menghitung probabilitas posterior $P(h|d)$ dari probabilitas sebelumnya $P(h)$ dengan $P(d)$ dan $P(d|h)$. Setelah menghitung probabilitas posterior dengan beberapa hipotesis, maka dapat dipilih hipotesis dengan probabilitas tertinggi. Ini adalah hipotesis dengan probabilitas paling tinggi biasanya disebut juga dengan MAP (*Maximum a Posteriori Probability*). Pada persamaan 2.11, 2.12, dan 2.13 dinyatakan bagamana persamaan untuk mencari MAP dari seluruh probabilitas hasil Naive Bayes.

$$MAP(h) = max(P(h|d)) \tag{2.11}$$

atau

$$MAP(h) = max\left(\frac{P(d|h) \times P(h)}{P(d)}\right) \tag{2.12}$$

atau

$$MAP(h) = max(P(d|h) \times P(h)) \tag{2.13}$$

$P(d)$ adalah aturan normalisasi yang biasanya digunakan dalam perhitungan probabilitas. Hal tersebut dapat tidak berlaku saat sudah ditemukannya hipotesa yang paling mungkin selama hal tersebut hanya berupa aturan yang konstan dan digunakan saat normalisasi saja maka dari itu pada persamaan 2.13 dapat ditinggal atau tidak diperhitungkan. Sedangkan h dalam $MAP(h)$ maksudnya adalah hipotesa dari seluruh hasil perhitungan dengan Naive Bayes yang memiliki probabilitas yang paling besar.

Kembali lagi ke klasifikasi, jika telah memiliki sejumlah *instance* atau sekumpulan data dalam setiap kelas pada data *training*, maka probabilitas masing-masing kelas $P(h)$ akan sama karena sudah terklasifikasi sebelumnya. Hal tersebut $P(h)$ menjadi sebuah aturan yang bersifat konstan, sehingga dapat dihapus dari persamaan 2.13 menjadi persamaan 2.14

$$MAP(h) = max(P(d|h)) \tag{2.14}$$

2.3.3 Klasifikasi dengan Naive Bayes

Naive Bayes adalah algoritma yang dapat digunakan untuk menyelesaikan masalah klasifikasi biner (dua kelas) dan multi kelas. Teknik ini paling mudah dipahami ketika dijelaskan dengan menggunakan nilai masukan biner atau terkategorisasi.

Metode ini dinamakan *naive* karena perhitungan probabilitas untuk setiap hipotesis disederhanakan agar kalkulasi dapat dilakukan. Jika dibandingkan dengan menghitung nilai dari masing-masing atribut $P(d1,\ d2,\ d3)$ maka diasumsikan masing-masing atribut saling independen dengan target nilai dan dihitung sebagai $P(d1|h) \times P(d2|h)$ dan seterusnya.

1). Representasi dari Naive Bayes adalah Probabilitas

Representasi dari Naive Bayes adalah probabilitas. Daftar probabilitas yang akan disimpan ke kesebuah variabel untuk dilakukan proses *learning* dengan model Naive Bayes. Berikut dua langkah yang biasanya dipakai dalam pengklasifikasian dengan Naive Bayes.

a. ***Class probabilities***: Di carinya probabilitas untuk setiap kelas dalam *dataset training*, jadi tujuan dari langkah ini adalah diperolehnya probabilitas pada setiap kelas, sekelompok data atau data *clusster*.

b. ***Conditional probabilities***: Sering disebut dengan probabilitas bersyarat, dicarinya probabilitas dari setiap nilai dalam *dataset training* terhadap kelas yang ada.

2). Proses Learning Data dengan Naive Bayes

Proses learning pada *data training* dengan menggunakan model Naive Bayes sangatlah cepat. Proses tersebut menjadi cepat karena hanya probabilitas pada masing-masing kelas dan probabilitas pada setiap kelas diberi nilai input (x) yang berbeda kemudian dihitung. Selain itu tidak

ada koefisien yang perlu dicocokkan dengan menggunakan prosedur optimasi. Di karenakan konsep dasar dari Naive Bayes adalah probabiltas yang mengandalkan prosees pembagian, maka dari itu tidak memakan terlalu banyak sumber daya komputasi.

3). Perhitungan pada *Class Probabilities*

Class probabilities sebenarnya adalah frekuensi dari *instance* atau sekumpulan data pada masing-masing kelas dibagi dengan jumlah total *instance*. Sebagai contoh dalam klasifikasi biner atau dua kelas, probabilitas *instance* yang termasuk dalam kelas ke 1 dapat dinyatakan dengan persamaan 2.15.

$$P(C_1) = \frac{C_1}{(C_0 \ + \ C_1)} \tag{2.15}$$

Pada persamaan 2.15 terdapat dua buah kelas yang mewakili kombinasi angka binier, angka 1 yang diwakili dengan C_1 dan angka 0 yang diwakili dengan C_0. Kemudian dicarilah peluang munculnya angka 1 atau $P(C_1)$. Dalam kasus paling sederhana setiap kelas akan memiliki probabilitas 0.5 atau 50% untuk masalah klasifikasi biner dengan jumlah *instance* yang sama di setiap kelas. Lebih jelasnya lagi bahwa terdapat sekumpulan data yang terkelompokkan menjadi dua yang diumpamakan dengan 0 dan 1.

4). Perhitungan *Conditional Probability*

Probabilitas bersyarat atau *Conditional probability* adalah frekuensi dari setiap atribut nilai terhadap setiap nilai pada sebuah kelas dibagi dengan frekuensi dari *instances* atau nilai-nilai pada kelas itu. Misalkan terdapat sebuah atribut "cuaca" atau "*weather*" memiliki nilai "cerah" dan "hujan" kemudian akan diklasifikasikan berdasarkan atribut dari kelas yang memiliki nilai "jalan-jalan" dan "tetap di rumah" yang akan digunakan untuk mengklasifikasikan atribut dari "cuaca", maka probabi-

litas bersyarat dari setiap atribut pada "cuaca" dapat dirumuskan dengan persamaan 2.16, 2.17, 2.18, dan persamaan 2.19.

$$P(W_1|C_1) = \frac{W_1 \times C_1}{C_1} \tag{2.16}$$

$$P(W_1|C_2) = \frac{W_1 \times C_2}{C_2} \tag{2.17}$$

$$P(W_2|C_1) = \frac{W_2 \times C_1}{C_1} \tag{2.18}$$

$$P(W_2|C_2) = \frac{W_2 \times C_2}{C_2} \tag{2.19}$$

Berdasarkan beberapa persamaan diatas atribut "cuaca" atau "*weather*" dinyatakan dengan W sedangkan atribut untuk mewakili kelas yang akan digunakan untuk mengklasifikasikan dinyatakan dengan C, kemudian hasil dari perhitungan probabilitas tersebut dinyatakan dengan P. Atribut dari "cuaca" terdiri dari "cerah" (W_1) dan "hujan" (W_2), sedangkan atribut dari kelas yang terdiri dari "jalan-jalan" (C_1) dan "tetap di rumah" (C_2). Kemudian $P(W_n|C_n)$ adalah probabilitas yang mungkin terjadi untuk "jalan-jalan" atau "tetap di rumah" saat kondisi "cerah" atau "hujan", dengan n yang dapat diganti dengan angka 1 dan 2.

5). Membuat Prediksi dengan Naive Bayes

Dengan didapatkannya model Naive Bayes, maka dapat dilakukan prediksi data baru menggunakan teorema bayes. Dengan menggunkan persamaan 2.13, di mana fokus dari persamaan tersebut adalah dipilihnya kelas dengan perhitungan probabilitas terbesar. Karena diperolehnya hipotesa "cuaca" yang mungkin untuk "jalan-jalan" adalah "cuaca" berada pada atribut "cerah".

Di gunakannya contoh kasus yang sama seperti pada Sub-Bab 2.3.3 poin ke 4 dan dipilihnya atribut "cerah" dimana cuaca adalah *instance*.

Kemudian dilakukan perhitungan untuk mencari nilai probabilitas maksimal dari "jalan-jalan" atau "tetap di rumah" yang merupakan bagian dari atribut kelas.

$$C_1 = P(W_1|C_1) \times P(C_1) \tag{2.20}$$

$$C_2 = P(W_1|C_2) \times P(C_2) \tag{2.21}$$

Seperti pada persamaan 2.20 dan 2.23 dimana C_1 adalah nilai maksimal probabilitas untuk atribut "jalan-jalan" dan C_2 adalah nilai maksimal probabillitas untuk "tetap di rumah" yang diperoleh dari kalkulasi probabilitas bersyarat dari kondisi "cerah" terhadap "jalan-jalan" dinyatakan dengan $P(W_1|C_1)$ dikalikan dengan probabilitas dari "jalan-jalan" $P(C_1)$ dan kondisi "cerah" terhadap "tetap di rumah" dinyatakan dengan $P(W_1|C_2)$ dikalikan dengan probabilitas dari "tetap di rumah". Kemudian dapat dipilihlah dari kedua atribut C_1 dan C_2 yang memiliki nilai tertinggi, maka itulah hasil prediksinya. Nilai-nilai tersebut yang kemudian dapat diubah kedalam bentuk probabilitas dengan menormalkannya seperti pada persamaan berikut.

$$P(C_1|W_1) = \frac{C_1}{C_1 + C_2} \tag{2.22}$$

$$P(C_2|W_1) = \frac{C_2}{C_1 + C_2} \tag{2.23}$$

Di mana pada persamaan 2.22 dan 2.23, $P(C_1|W_1)$ adalah probabilitas untuk "jalan-jalan" dan $P(C_2|W_1)$ adalah probabilitas untuk "tetap di rumah". Jika terdapat lebih banyak variabel input, maka contoh diatas dapat lebih diperluas lagi. Misalkan dengan ditambahkan atribut "mobil" dengan nilai "berjalan" dan "rusak", maka probabilitasnya dikalikan menjadi sebuah persamaan. Sebagai contoh perhitungan pada kelas dengan

atribut "jalan-jalan" ditambahkan *instance* baru yaitu "mobil" yang beri hipotesa "berjalan" maka persamaan 2.20 berubah menjadi persamaan 2.24. Dengan adanya penambahan variabel baru tersebut maka persamaan dari Naive Bayes juga ikut berubah mengikuti kondisi.

$$C_1 = P(W_1|C_1) \times P(V_1|C_1) \times P(C_1) \tag{2.24}$$

Di mana pada persamaan 2.24 terdapat variabel V yang menyatakan "*vehicle*" atau "mobil", kemudian terdapat dua kondisi atau atribut pada variabel tersebut yaitu "berjalan" dan "rusak" yang masing-masing dapat dinyatakan dalam variabel V_1 dan V_2.

2.3.4 Gaussian Naive Bayes

Penggunaan Naive Bayes dapat diperluas ke atribut yang bernilai bilangan real, hal yang paling umum dilakukan adalah dengan membuat asumsi distribusi .

Gaussian Naive Bayes adalah salah satu pengembangan dari Naive Bayes. Salah satu kegunaanya adalah dapat digunakan untuk memperkirakan distribusi data, tetapi istilah (distribusi normal) adalah salah satu cara termudah untuk digunakan, karena hanya perlu memperkirakan rata-rata dan standar deviasi dari data *training* yang ingin digunakan.

1). Representasi Gaussian Naive Bayes

Pada bagian sebelumnya penghitungan probabilitas untuk nilai masukan data pada setiap kelas menggunakan frekuensi. Dengan masukan yang berupa angka-angka, kemudian dihitung rata-rata dan standar deviasi dari masukan (x) pada setiap kelasnya untuk dilakukan distribusi data baru.

Hal tersebut menunjukan bahwa selain terjadinya penambahan probabilitas pada setiap kelas, ditambahkan juga rata-rata dan standar deviasi pada setiap masukan variabel dari masing-masing kelas.

2). Proses Learning Data dengan Gaussian Naive Bayes

Hal ini sesederhana menghitung nilai rata-rata dan standar deviasi dari setiap nilai dari variabel masukan (x) pada setiap kelas.

$$\bar{x} = \frac{1}{n} \sum_{i=0}^{k} x \tag{2.25}$$

Di mana pada persamaan 2.25, variabel n adalah jumlah nilai dari sekumpulan data dan x adalah nilai-nilai untuk variabel masukan dalam data training. Kemudian k adalah banyaknya data *training*, dan $\bar{x}_k$ adalah rata-rata nilai dari data *training*. Kemudian untuk standar deviasi dapat dihitung dengan menggunakan persamaan berikut ini.

$$\sigma(x) = \sqrt{\frac{\sum_{i=0}^{k} (x_i - \bar{x})^2}{n}} \tag{2.26}$$

Pada dasarnya standar deviasi pada persamaan 2.26 adalah akar kuadrat dari perbedaan rata-rata yang dikuadratkan dari setiap nilai x yang dinyatakan dengan x_i dari hasil rata-rata atau $\bar{x}$. Kemudian n adalah banyaknya data dan i adalah urutan dari setiap data tersebut. Hasil dari perhitungan standar deviasi tersebut dinyatakan dengan $\sigma(x)$ atau standar deviasi dari x.

3). Prediksi dengan Model Gaussian Naive Bayes

Pada bagian ini akan dijelaskan prediksi nilai x selanjutnya dengan menggunakan PDF (*Probability Density Function*). Saat membuat prediksi, parameter ini dimasukan ke PDF dengan masukan baru untuk variabel, dan sebagai gantinya PDF akan memberikan perkiraan probabilitas nilai masukan baru untuk kelas tersebut.

$$PDF(x, \bar{x}, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \; exp\left(-\frac{(x - \bar{x})^2}{2\sigma^2}\right) \tag{2.27}$$

Di mana maksud dari $PDF(x, \bar{x}, \sigma)$ pada persamaan 2.27 adalah PDF dari x yaitu nilai masukan untuk variabel masukan untuk prediksi,

$\bar{x}$ atau rata-rata dan σ adalah standar deviasi yang sudah dihitung pada bagian atau berdasarkan data-data sebelumnya, π adalah konstanta numerik pada umumnya, kemudian fungsi $exp()$ atau e adalah konstanta numerik yang betujuan untuk membentuk hasil prediksi dengan pendekatan exponensial.

Kemudian pada persamaan 2.27 dapat dimasukan ke dalam persamaan 2.24 untuk membuat prediksi dengan masukan data baru. Sebagai contoh, dengan mengadaptasi salah satu perhitungan pada persamaan 2.24 untuk "cuaca" dan "car".

$$C_1 = P(PDF(W)|C_1) \times P(PDF(V)|C_1) \times P(C_1) \tag{2.28}$$

Pada persamaan 2.28 penjelasan masih sama persis dengan 2.28, tetapi data hipotesa diganti dengan fungsi Gauss PDF. Di mana $PDF(W)$ adalah data hasil perdiksi data baru untuk "cuaca", kemudian digunakan hipotesa dalam memprediksi hasil prediksi. Hal tersebut berlaku juga untuk data hasil prediksi "mobil", dimana akan digunakan hipotesa dalam memprediksi hasil prediksi.

2.4 Role-Playing Game

Role-playing Game (RPG) (Panumate et al., 2015) adalah jenis permainan yang mana pemain memainkan peran karakter dalam latar fiktif. Selain itu pemain juga bertanggung jawab untuk memainkan peran sesuai narasi cerita, baik melalui adegan secara literal atau melalui proses pembuatan keputusan dalam setiap adegannya yang berujung pada pengembangan karakter. Salah satu contoh permainan digital bagian dari genre RPG yang populer adalah *Western* RPG yang mana pemain memerankan sebuah karakter dan menjalankan cerita dari permainan tersebut seperti pada Gambar 2.3.

Pada dasarnya permainan dengan genre RPG (*Role-Playing Game*) khususnya video game terbagi menjadi banyak kategori diantaranya adalah WR-

Gambar 2.3: Contoh *video game* dengan genre *RPG*.

PG, JRPG, ARPG, TRPG, SRPG, dan MMORPG (Stenström, 2012). Penjelasan selengkapnya akan dijelaskan pada Sub-bab selanjutnya dari Sub-bab 2.4.1 sampai dengan Sub-bab 2.4.6 berikut ini.

2.4.1 WRPG

Deskripsi tentang genre WRPG (*Western Role-Playing Game*) ini diambil dari *Dungeons and Desktops: The History of Computer-Role-playing Games* oleh Barton. Namun seperti yang direpresenatasikan oleh Barton menggunakan istilah CRPG, sedangkan dalam penelitian ini istilah WRPG yang akan digunakan (Barton and Stacks, 2019).

Pada WRPG pemain dapat bebas mengkostumisasi karakternya dengan leluasa seperti jenis kelamin, wajah dan lain sebagainya. Kemudian pemain juga dapat memilih kemampuannya yang direalisasikan dalam bentuk atribut *gameplay* dari karakter pemain tersebut (Barton and Stacks, 2019). Pada umumnya hal yang dipilih oleh pemain biasanya berupa suku atau fraksi seperti pada *video game* berjudul *The Elder Scroll V: Skyrim*, Bethesda (2011) seperti pada Gambar 2.4.

Sekumpulan musuh akan muncul sebelum permainan dimulai, dan pemain dapat menyiapkan perangkap atau membuat langkah kongkrit untuk mengalahkan sekumpulan musuh tersebut. Sekumpulan musuh tersebut bia-

Gambar 2.4: Contoh kustomisasi karakter pada WRPG.

sanya tersebar jumlahnya lebih banyak dari pemain. Atau mungkin pemain hanya berjumlah 1 dengan musuh yang berjumlah lebih dari itu, kemudian tugas dari pemain adalah mengalahkan sekumpulan musuh yang tersebar tersebut dengan melakukan pertarungan secara langsung atau *real-time*, hal ini sangat identik dengan ARPG (akan dibahas pada poin selanjutnya).

Biasanya juga pada WRPG terapat banyak *side quest* atau cerita sampingan yang berbeda dengan cerita utama. Hal ini memberikan pilihan jalan cerita untuk pemain dalam menyelesaikan permainan, selain itu karakter musuh dan juga boss dari musuh berbeda dengan misi atau cerita utama. Pada *video game* berjudul *The Witcher 3*, CD Pojekt (2015) memiliki banyak sekali cerita sampingan bahkan bisa dibilang cerita sampingan dari permainan tersebut tidak kalah bagusnya dengan cerita utama. Sekilas mengenai *video game* berjudul *The Witcher 3* dapat dilihat pada Gambar 2.3 pada bagian sebelumnya, saat dilakukan pertarungan secara *real-time* melawan monster.

2.4.2 JRPG

Pada mulanya permaian dengan genre JRPG (*Japanese Role-Playing Game*) tidak begitu banyak mengandung kustomisasi karakter layaknya WRPG. Tetapi pada permainan RPG dengan genre ini berfokus pada beberapa karakter atau biasa disebut dengan istilah *party member* yang dapat disusun oleh

pemain. Sama seperti pada WRPG dimana pemain dapat mengkustomisasi kemampuan dari setiap *party member* sesuai dengan peran masing-masing dengan tujuan untuk mengalahkan musuh. Dalam kostumisasi karakter pemain, biasanya diwujudkan dalam penambahan atribut *gameplay* dan jurus seperti pada *video game* berjudul *Shin Megami Tensei: Digital Devil Saga 1* dan *2*, Atlus (2004, 2005) seperti yang ditunjukan pada Gambar 2.5.

Gambar 2.5: Contoh kustomisasi atribut *gameplay* pada JRPG.

Pada mulanya pada permainan JRPG, pemain tidak mengetahui akan kedatangan musuh. Kedatangan musuh muncul secara tiba-tiba saat pemain melakukan pergerakan dalam peta permainan, hal ini biasa disebut denga istilah *random encounter*. Contoh pada kondisi tersebut terdapat pada *video game Shin Megami Tensei: Digital Devi Saga 1* dan *2*. Jumlah musuh yang harus dilawan oleh *party member* juga berjumlah lebih dari satu, sama-sama membentuk *party member*.

Namun pada beberapa *video game* JRPG, *random encounter* juga direpresentasikan seperti monster atau bayangan yang mengejar representasi dari karakter pemain yang mewakili *party member* dalam menjelajah dunia (eksplorasi). Setelah mereka bertabrakan maka terjadilah pertarungan antara *party*

member pemain dan musuh, seperti pada *video game* berjudul *Tales of Zestiria*, Bandai Namco (2015) dan juga *Persona 4*, Atlus (2008).

Penataan strategi pada *genre* JRPG bersifat sangat krusial, jika pada genre WRPG, ARPG dan SRPG pemain dapat mengganti perlalatan perangnya tetapi tidak pada genre JRPG. Pemain harus mempersiapkan segala seuatu dengan benar, apa saja yang dibutuhkan untuk mengalahkan musuh. Dalam mode pertarungan pada genre JRPG ada yang bersifat *turn-based* seperti TRPG (akan dibahas pada poin selanjutnya) dan ada pula yang bersifat *real-time* seperti WRPG.

Permainan berbasis *turn-based* (Panumate et al., 2015) adalah permainan yang memanfaatkan waktu secara diskrit, yang mana alur dari permainan khususnya pertarungan terbagi menjadi beberapa bagian yang disebut dengan giliran. Pada setiap giliran, pemain akan memiliki waktu yang terbatas atau tidak terbatas dalam berpikir dalam membuat keputusan setiap langkahnya. Kemudian sistem dari permainan akan memproses tindakan dari pemain. Kemudian permainan dilanjutkan oleh pemain berikutnya atau musuh yang berupa AI akan melakukan giliran selanjutnya seperti pada Gambar 2.6.

Gambar 2.6: Contoh permainan digital dengan genre *turn-based*.

Pada Gambar 2.7 adalah mode pertarungan JRPG secara *turn-based* yang dicontohkan oleh *Persona 4*, Atlus (2008). Di mana para karakter yang

dimainkan oleh pemain harus melakukan serangan secara bergatian yang kemudian baru dilanjutkan dengan serangan dari musuh.

Gambar 2.7: Mode pertarungan *turn-based* pada JRPG.

Pada Gambar 2.8 adalah mode pertarungan JRPG secara *real-time* yang dicontohkan oleh *Tales of Zestiria*, Bandai Namco (2015). Di mana pemain menentukan formasi dalam party member, siapa saja karakter yang akan bertarung dan *skill* apa saja yang dibutuhkan untuk mengalahkan musuh. Saat berlangsungnya pertarungan hanya satu karakter yang dapat dijalankan oleh pemain, tetapi pemain diberikan kelebihan untuk berganti karakter dalam pertarungan atau *switch*.

Gambar 2.8: Mode pertarungan *real-time* pada JRPG.

Sama seperti dengan WRPG pada JRPG juga memiliki *side-quest* tetapi biasanya hanya berupa permainan kecil saja, yang bersifat memainkan permainan lain diluar pertarungan antar *party member*. Selain itu juga terdapat *secret boss* yang bersifat sangat kuat dan sulit dikalahkan. Kemudian adanya *new game+*, yang memungkinkan pemain untuk tetap memainkan karakter dengan kemampuan yang sama seperti permainan sebelumnya dan adanya penambahan konten untuk dimainkan (biasanya *boss* yang lebih banyak).

2.4.3 ARPG

Biasanya genre ini disebut juga dengan "*Hack and Slash*" atau "*Dungeon crawl RPG's*" (Moore, 2016). Kenapa sampai diberi sebutan "*Hack and Slash*", hal ini mengacu dari kesederhanaan dari beberapa permainan ARPG. Pada permainan RPG jenis ini (khususnya *video game*) berfokus pada aksi atau ketangkasan seperti memukul, menghindar, mengeluarkan jurus dan lain sebagainya dibandingkan dengan pengaturan strategi atau penyusunan atribut *gameplay*. Jika melihat cara bertarung pada permainan jenis ini, maka akan sangat mirip sekali dengan WRPG yang mana pemain fokus dalam bertarung dan berekplorasi. Contoh genre ini adalah *Diablo*, Blizzard (1997) dan *Final Fantasy 12*, Square Enix (2006).

2.4.4 TRPG

Pada RPG jenis ini tidak memiliki dunia atau maps yang digunakan untuk bereksploarsi sama sekali. Jadi pemain tidak perlu berkeliling sama sekali, sedangakan untuk sisi cerita pemain hanya melihat interaksi antar karakter atau interaksi antara karakter pemain dengan karakter musuh yang membentuk sebuah cerita.

Jumlah karakter yang dimainkan oleh pemain biasanya berjumlah sangat banyak dan terpapar dalam sebuah panel isometrik atau kotak, dan disanalah dilakukannya pertarungan antara karakter-karakter dari pemain melawan musuh-musuh. Pada RPG genre ini sangat berbanding terbalik dengan ARPG, yang mana pada TRPG sama sekali tidak membutuhkan aksi. Dalam ar-

tian tidak diperlukannya ketangkasan dalam melakukan serangan atau dalam menghindari serangan musuh, yang sangat diperlukan adalah 100% perencanaan dan strategi. Pertarungan benar-benar dilakukan secara bergantian antara pemain dan musuh dalam melakukan serangan (Moore, 2016). Contoh dari permainan dengan genre ini adalah *Advance Wars*, Nintendo (2001). Penjelasan dari *turn-based* atau bergantian sudah dijelaskan juga pada Sub-Bab 2.4.2 dan juga dicontohkan dengan *video game* yang berjudul *Duelyst*, Counterplay Games (2016).

Hampir sebagian besar permainan dengan genre ini berasal dari pengembang *video game* asal jepang selayaknya JRPG. Terdapat juga versi lain TRPG, dengan adanya penambahan atribut *speed* atau kecepatan yang menentukan karakter mana yang akan melakukan aksi terlebih dahulu. Contoh dari permainan tersebut terdapat pada *Final Fantasy Tactics*, Square (1997).

2.4.5 SRPG

Genre RPG ini adalah termasuk yang paling modern, pada penelitian ini *shooter* RPG dipersingkat menjadi SRPG. Biasanya istilah SRPG rancu dengan TRPG, yang mana S tersebut diartikan menjadi *strategy*. SRPG sangat mirip sekali dengan ARPG hanya saja senjata utama dari permain ini menggunakan senapan atau panahan. Mengingat RPG jenis ini adalah *shooter*, bukan berarti prespektif dari permainan ini adalah FPS (*First Person Shooter*). Bisa jadi juga prespektifnya adalah *third person* atau bahkan pemain bisa memilih mode yang mereka inginkan seperti pada *video game* berjudul *Fallout 4*, Bethesda (2015).

2.4.6 MMORPG

Kebanyakan *gameplay* dari MMORPG (*Masive Multiplayer Online Role-Playing Game*) terbagi menjadi dua jenis. Pertama adalah *gameplay* yang mengedepankan kenaikan level dan selanjutnya adalah *gameplay* yang mengedepankan penguatan karakter berdasarkan item yang dimiliki dalam memperankan perannya. Genre ini sangat mirip dengan WRPG dan JRPG, namun

dijalankan secara *online* atau daring yang mempertemukan banyak pemain dan terciptalah interaksi antar pemain yang diwakilkan dengan karakter yang dimainkan dalam permainan tersebut.

Permainan ini difokuskan pada interaksi sosial, dan perdagangan antar pemain, menciptakan perekonomian virtual, dan perusahaan virtual dari para pemain, yang biasa disebut persekutuan, grup, atau *guilds*. *Guilds* akan memiliki tujuan yang berbeda dalam permainan (*Guilds* besar sering aktif dalam beberapa misi atau pertarungan), biasanya mereka akan fokus pada membersihkan *dungeon* yang jauh lebih sulit yang mengharuskan banyak pemain atau anggota *guilds* untuk ikut serta. Ini adalah alasan normal dari pemain untuk bergabung dengan *guilds*, dengan menemukan para pemain pada waktu yang tepat, dan yang dapat memainkan peran dan permainan dengan baik. Hal itu akan menjadi sulit, jika tidak ada semacam jaringan atau penghubung antar pemain (Nick, 2009). Contoh dari permainan MMORPG yang populer adalah *World of Warcraft*, Blizzard Entertainment (2004) dan *Ragnarok Online*, Gravity, Gravity Interactive (2002).

Sistem pertarungannya adalah gabungan dari WRPG, JRPG, ARPG, dan TRPG. Namun pada umumnya MMORPG mirip dengan WRPG, terdapat pada bagian pembuatan karakter, dan pertempuran secara *real-time* meskipun tidak benar-benar *real-time*. Selain itu, terdapat beberapa permainan MMORPG yang memiliki pola permainan khas JRPG (Moore, 2016), dengan banyaknya karakter yang dikendalikan oleh seorang pemain dan melakukan serangan sercara bergantian seperti *Summoners War: Sky Arena*, Com2uS (2014) pada Gambar 2.9. Kemudian dengan adanya *boss* dan banyak musuh dengan susunan dan kemampuan yang sudah terpola. Terdapat juga MMORPG dengan pertempuran yang sering kali mengharuskan para pemain untuk bergerak dan menghindari kemampuan musuh dan melakukan serangan disaat yang tepat seperti pada ARPG.

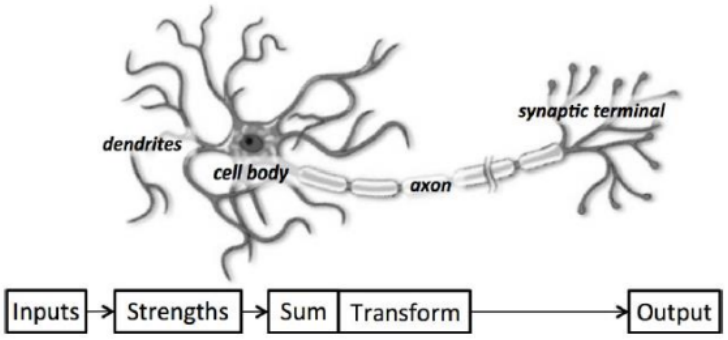
Terakhir, permainan menyediakan fitur berupa pengembangan karakter dan perencanaan dari karakter yang dimainkan oleh para pemain, dan terle-

Gambar 2.9: MMORPG yang menyerupai JRPG.

lih lagi mengharuskan pemain untuk melakukan perencanaan untuk bertarung bersama atau *party* tetapi setiap pengguna hanya dapat memainkan satu karakter dalam *party* tersebut, *gameplay* tersebut sangat mirip dengan apa yang ditawarkan pada genre TRPG.

2.5 Artificial Neural Network

Terinspirasi dari cara kerja otak manusia dalam memproses informasi seperti yang ditunjukan pada Gambar 2.10 (Buduma and Locascio, 2017), maka dibuatlah sebuah paradigma pemprosesan informasi yang biasanya digunakan untuk mempelajari tingkah laku sistem yang kompleks pada komputer, istilah ini biasa disebut dengan *Artificial Neural Network*.



Gambar 2.10: Skruktur neuron dan deskripsi fungsinya.

Dengan mengiplementasikan cara kerja dari satu neuron dengan neuron yang lain yang diterapkan dalam program untuk komputer. Persamaan yang

menyatakan perhitungan dari satu unit neuron adalah seperti yang ditujukan pada persamaan 2.29.

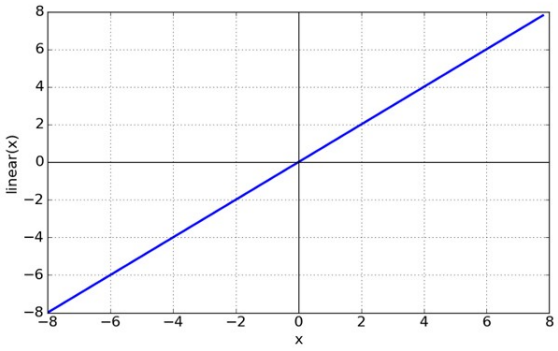
$$y_{out} = wx_{in} + b \tag{2.29}$$

Masukan skalar x ditransmisikan melalui koneksi yang nilainya dikalikan oleh skalar beban atau *weight* dengan w. Hasil perkalian tersebut ditambahkan dengan skalar bias yang dinyatakan dengan b. Proses tersebut menghasilkan luaran skalar yang dinyatakan dengan y, yang mana nilainya akan diteruskan ke neuron selanjutnya.

2.5.1 Activation Function

Activation function pada bertugas untuk menentukan apakah neuron tersebut harus aktif atau tidak, seperti neuron dalam otak manusia. Aktifasi neuron tersebut dipengaruhi oleh jumlah w dari masukan. Secara umum terdapat 2 jenis *activation function* yaitu linear dan non-linear.

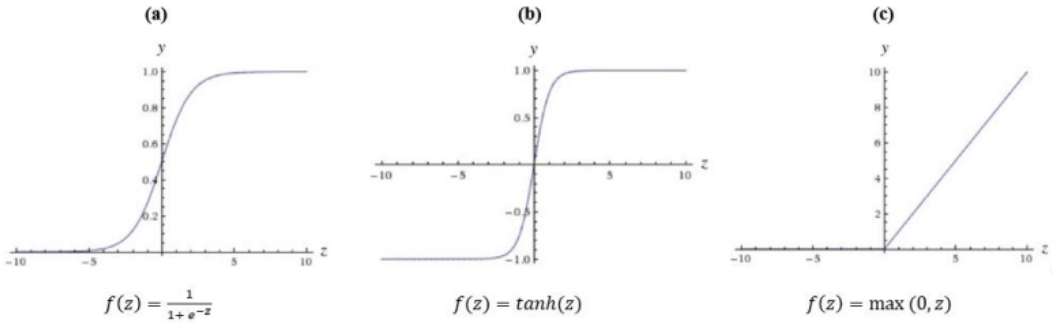
Pada dasarnya *activation function* dari sebuah neuron adalah Linear. Jika sebuah neuron menggunakan fungsi linear, maka keluaran dari neuron tersebut adalah w dari $x + b$. Pada Gambar 2.11 adalah contoh keluaran dari sebuah fungsi linear.



Gambar 2.11: Output fungsi linear

Fungsi sigmoid mempunyai rentang antara 0 hingga 1 sedangkan rentang dari Tanh adalah -1 hingga 1. Kedua fungsi ini biasanya digunakan untuk

klasifikasi dua kelompok data. Kemudian pada ReLU biasanya dilakukan *treshold* dari 0 sampai dengan tidak terhingga. Pada Gambar 2.12 (Buduma and Locascio, 2017) adalah contoh keluaran dari sebuah fungsi non-linear, yang masing-masing secara berurutan dengan (a) adalah Sigmoid, (b) adalah Tanh, dan (c) adalah ReLU.



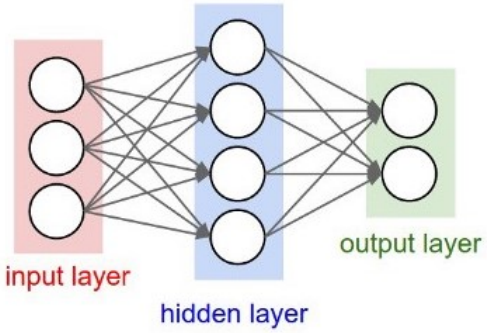
Gambar 2.12: Contoh fungsi aktivasi.

Kemudian terdapat sebuah activation function lagi yang akan digunakan pada penelitian ini, yaitu Softmax. Pada suatu kondisi akan dibutuhkan hasil klasifikasi yang berupa vektor dengan label atau kategori yang masih saling berhubungan. Contohnya saat penggunaan Neural Network dalam mengenali tulisan tangan yang berupa angka pada dataset MNIST. Hasil prediksi yang berupa angka atau label (0 sampai 9) yang masih saling terhubung, seperti kecilnya kemungkinan sebuah tulisan tangan angka dikenali dengan tingkat keyakinan 100%. Maka digunakanlah probabilitas distribusi sehingga setiap label masih memiliki tingkat keyakinan masing-masing (Buduma and Locascio, 2017).

2.5.2 Arsitektur Neural Network

Arsitektur pada Gambar 2.13 biasa disebut sebagai *Multi Layer Perceptron* (MLP) atau *Fully-Connected Layer*. Arsitektur tersebut mempunyai 3 buah neuron pada *input* Layer dan 2 buah *node* pada *output Layer*. Di antara *input* dan *output*, terdapat 1 *hidden layer* dengan 4 buah neuron seperti yang ditunjukan pada Gambar 2.13. Setiap neuron pada MLP saling berhubungan yang ditandai dengan tanda panah pada gambar diatas. Tiap koneksi memili-

ki *weight* yang nantinya nilai pada setiap *weight* akan berbeda-beda. *Hidden layer* dan *output layer* memiliki tambahan *input* yang biasa disebut dengan bias. Sehingga pada arsitektur pada Gambar 2.13 terdapat 3×4 *weight* + 4 *bias* dan 4×2 *weight* + 2 *bias*. Total keseluruhan terdapat 26 parameter yang pada proses *training* akan mengalami perubahan demi memperoleh hasil yang terbaik.



Gambar 2.13: Contoh arsitektur sederhana *Neural Network*.

Kemudian neuron pada *input* layer tidak memiliki *activation function*, sedangkan neuron pada *hidden layer* dan *output layer* memiliki *activation function* yang kadang berbeda tergantung daripada data atau problem yang ingin diselesaikan.

2.5.3 Training pada Neural Network

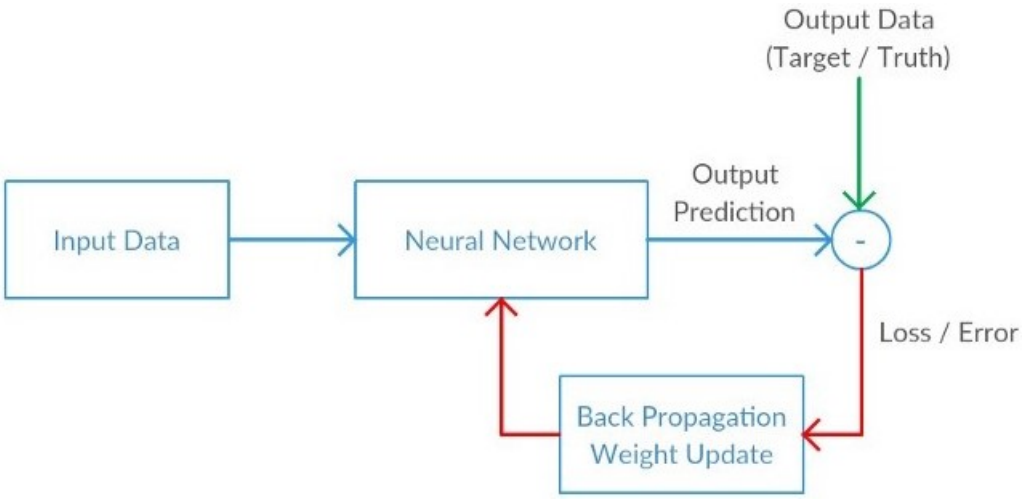
Pada *Supervised Learning* menggunakan *Neural Network*, pada umumnya proses *learning* terdiri dari 2 tahap, yaitu *training* dan *evaluation*. Namun kadang terdapat tahap tambahan yaitu *testing*, namun sifatnya tidak wajib. Dalam tahapan *training* setiap *weight* dan *bias* pada tiap neuron akan diupdate terus menerus hingga *output* yang dihasilkan sesuai dengan harapan. Pada tiap iterasi akan dilakukan proses *evaluation* yang biasanya digunakan untuk menentukan kapan harus menghentikan proses *training* atau stopping point. Secara garis besar proses training terdiri dari dua tahap sebagai berikut.

a. ***Forward Pass***: *Forward pass* atau biasa disebut *forward propagation* adalah proses dimana data dibawa dari input melewati setiap neuron

pada *hidden layer* sampai kepada *output layer* yang nanti akan dilakukan perhitungan *error*.

b. ***Backward Pass***: *Error* yang didapat pada *forward pass* akan digunakan untuk mengupdate setiap *weight* dan *bias* dengan *learning rate* tertentu.

Kedua proses tersebut dilakukan secara berulang-ulang sampai didapatkan nilai *weight* dan *bias* yang dapat memberikan nilai *error* sekecil mungkin pada *output layer* (saat *forward pass*). Bila digambarkan maka akan menjadi seperti pada Gambar 2.14 dengan panah biru adalah *forward pass* dan panah merah adalah *backward pass*.



Gambar 2.14: Alur training untuk saat *forward pass* dan *backward pass*.

Dalam *Supervised Learning*, *training data* terdiri dari *input* dan *output* atau target. Pada saat *forward pass*, input akan dipropagasikan menuju *output* layer dan hasil prediksi *output* akan dibandingakan dengan target dengan menggunakan sebuah fungsi yang biasa disebut dengan *Loss Function*. Maka untuk apa *loss function* tersebut? Sederhananya loss function digunakan untuk mengukur seberapa baik performa dari *Neural Network* yang dibuat saat melakukan prediksi terhadap target.

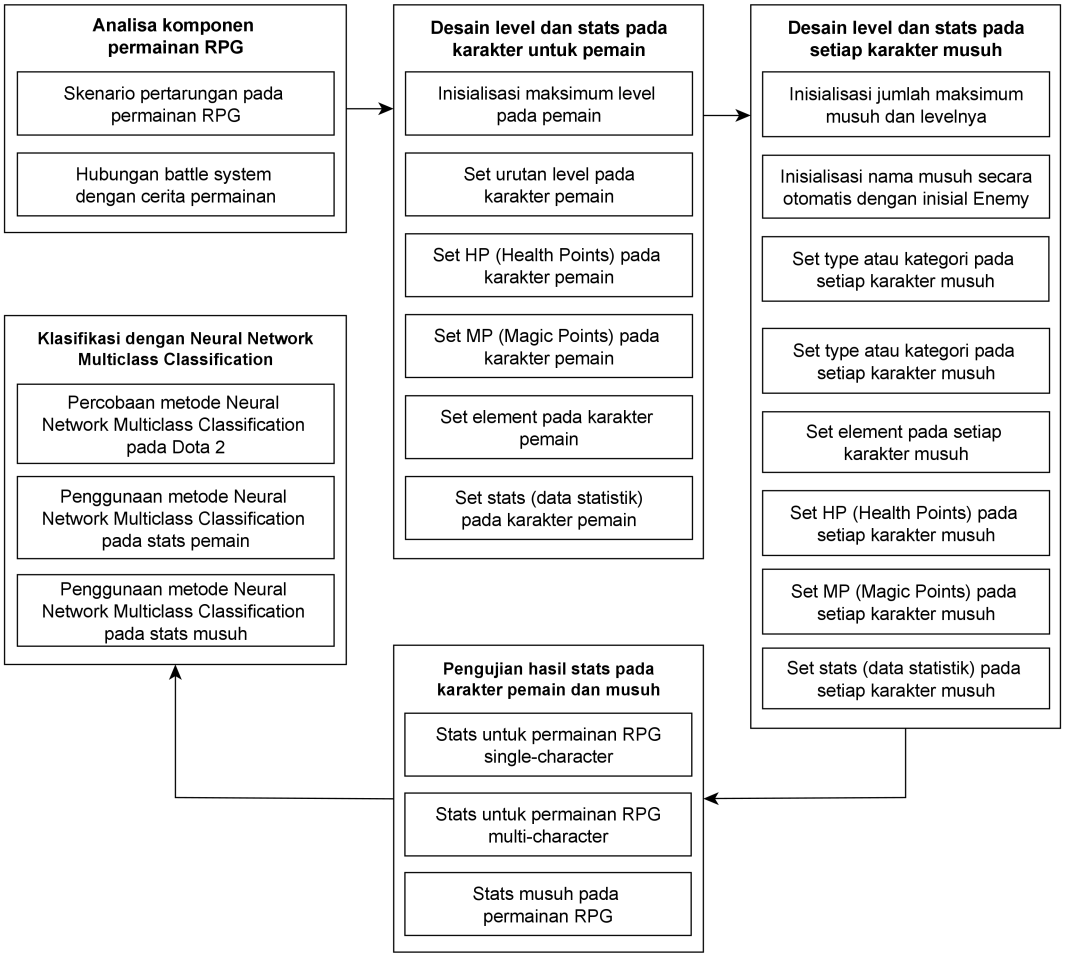
$$Loss = (Target - Predict)^2 \tag{2.30}$$

Pada persamaan 2.30 terdapat variabel *Target* yang merupakan hasil yang diinginkan kemudian dikurangin dengan hasil prediksi yang dinyatakan denga *Predict*, maka diperolehlah *Loss* atau tingkat kegagalan saat prediksi. Terdapat berbagai macam *loss function*, namun yang paling sering digunakan adalah *Squared Error* (L2 Loss) untuk regresi. Sedangkan untuk klasifikasi biasanya digunakan adalah *Cross Entropy*.

BAB 3

METODOLOGI PENELITIAN

Pada penelitian ini nantinya akan terdiri dari lima langkah utama yaitu analisa komponen permainan RPG, kenaikan level dan atribut *gameplay* pada karakter pemain, distribusi level dan atribut *gameplay* pada setiap karakter musuh, dilanjutkan pengujian atribut *gameplay* pada karakter pemain dan musuh, dan dilakukan klasifikasi dengan *Neural Network Mullticlass Classification*. Setiap langkah yang dilakukan dibawahnya merupakan detail dalam pengerjaan seperti yang direpresentasikan pada Gambar 3.1.



Gambar 3.1: Urutan metodologi.

Tujuan umum dari penelitian ini adalah membuat atribut *gameplay* untuk pemain dan musuh, sehingga dalam desain permaian menjadi lebih cepat. Tujuan khusus dan fokus pada penelitian ini adalah untuk membuat atribut *gameplay* untuk karakter pemain dan musuh dengan menggunakan metode $k-$NN dan *Naive Bayes* yang kemudian dilakukan klasifikasi dengan menggunakan metode *Neural Network Multiclass Classification*.

3.1 Analisa Komponen Permainan RPG

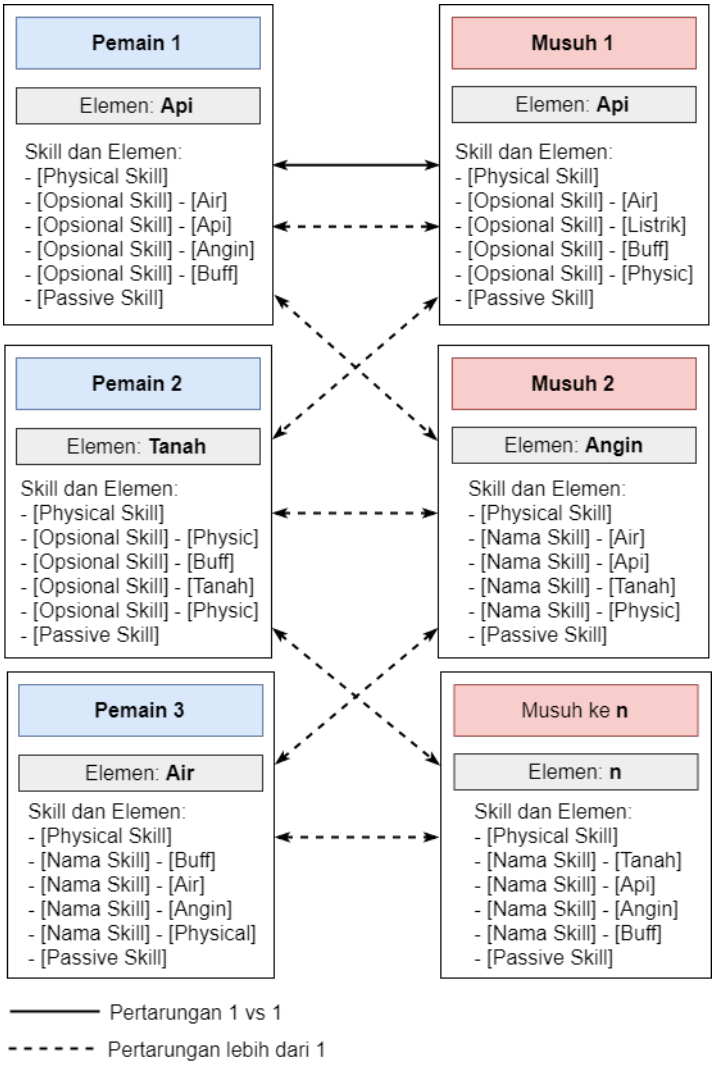
Pada bagian ini terdapat beberapa langkah yang akan menjelaskan penyusunan skenario pertarungan pada permainan RPG *turn-based*. Pertama yang akan dilakukan adalah pembuatan skenario pertarungan pada permainan *turn-based* RPG. Dari skenario tersebut tentunya harus ada hubungan dengan cerita dari permainan, apa saja parameter yang berpengaruh dari cerita terhadap skenario. Dalam proses analisa ini cukup dilakukan pada RPG yang tergolong *turn-based*, karena hal tersebut dianggap paling ideal. Saat karakter pemain dan musuh saling serang secara bergantian.

Di lanjutkan dengan desain level dan atribut *gameplay* dari pemain, selain itu digunakannya algoritma $k-$NN (*Nearest Neighbor*) untuk mempercepat proses pembuatannya. Kemudian dilanjutkan dengan desain level dan atribut *gameplay* musuh yang juga dibuat secara otomatis dengan algoritma yang sama dengan pemain, namun tetap dipolakan oleh *gaussian naive bayes* atau distribusi normal.

Bagian selanjutnya adalah penambahan elemen pada pemain dan musuh yang membentuk kelebihan atau kekurangan pada masing-masing karakter. Pembagian elemen pada karakter yang dapat dimainkan oleh pemain dilakukan sesuai dengan cerita, sedangkan pembagian elemen pada musuh disebar secara acak berdasarkan atribut *gameplay* yang telah dibuat. Hal ini berkaitan erat dengan penjelasan pada Sub-bab 2.2.1 tentang meningkatnya keberagaman, yang dibuktikan dengan banyaknya variasi musuh berikut dengan kelemahan dan kelebihannya.

3.1.1 Desain Skenario Pertarungan

Di umpamakan jumlah karakter yang rencananya akan digunakan berjumlah tunggal atau *single-character* (WRPG, ARPG, SRPG, dan MMORPG) sampai tiga karakter atau jamak atau *multi-character* (TRPG dan JRPG) bergantung dengan alur cerita dari permainan dan jumlah musuh juga dimisalkan berjumlah antara satu sampai dengan enam, bergantung kepada tingkat kesulitannya seperti pada Gambar 3.2.



Gambar 3.2: Skema pertarungan antar pemain.

Sebelumnya pernah dijelaskan pada Sub-bab 2.2.1 tentang meningkatnya pengambilan keputusan, pada kondisi semakin banyak musuh makan semakin

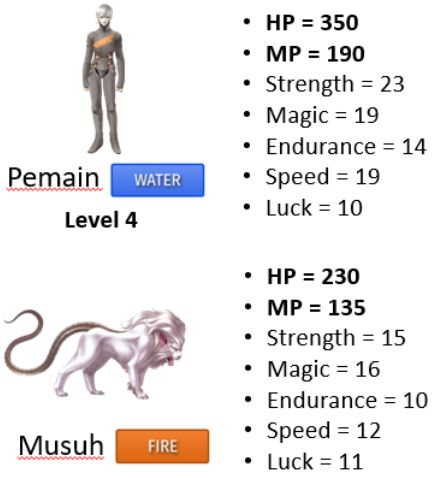
banyak keputusan yang harus diambil oleh pemain. Selain itu setiap pemain memiliki elemen dan *skill*, setiap elemen dapat menjadi kelemahan dan kelebihan dari setiap karakter. Hal tersebut akan membangun sebuah momen dramatis berdasarkan kompleksitas kombinasi dari musuh, hal tersebut dibahas pada Sub-bab 2.2.1 tentang momen dramatis pada desain permainan.

Pada bagian ini dijelaskan contoh perancangan dari skenario pertarungan dengan jumlah karakter tunggal dan karakter pemain yang berjumlah lebih dari satu dengan dibandingkan secara langsung. Pada Gambar 3.2 jika dipecah dan dijelaskan lebih lanjut maka setiap karakter yang dimainkan oleh pemain atau musuh dalam bentuk NPC (*Non Playable Character*) maka bagian-bagian tersebut akan menjadi seperti beberapa poin dibawah ini.

1). Atribut *Gameplay*

Atribut *gameplay* memiliki ciri non-visual dan menyajikan informasi tentang sebuah karakter dalam permainan. Setiap permainan memiliki tipe karakteristik dapat dideskripsikan. Contohnya pada salah satu papan permainan bergenre RPG yang terkenal yaitu Dungeons and Dragons (Heinsoo et al., 2008), yang memiliki enam atribut *gameplay* seperti *Strength* (kemampuan fisik), *Constitution* (kebugaran dan stamina), *Dexterity* (Koordinasi tangan dan mata, kelincahan, reflek, dan keseimbangan), *Intelligence* (Kemampuan untuk belajar dan bernalar), *Wisdom* (kemampuan umum, presepsi, kedisiplinan individu), dan *Charisma* (Kepribadian, kemampuan persuasif, kepemimpinan).

Atribut *gameplay* dalam RPG sangatlah diperhitungkan dan berpengaruh dalam proses pertarungan. Pada Gambar 1.1 ditunjukan tidak hanya karakter pemain saja, melainkan status dari pemain saat melakukan pertarungan. Pada Gambar 3.3 adalah penggambaran dari atribut *gameplay* pemain yang lebih detail seperti *Health Point* (HP), *Attack* atau serangan, *Defense* atau pertahanan, *Speed* atau kecepatan. Hal tersebut membuat komputer dapat membuat keputusan (Schuller, 2017).



Gambar 3.3: Status dari pemain pada permainan bergenre RPG.

Berikut adalah penjabaran lebih dari beberapa komponen seperti HP, *Attack*, *Defense*, *Speed* yang terdapat pada Gambar 1.1.

a). **Health Point (HP)** adalah indikator nyawa atau kehidupan dari pemain, jika HP bernilai 0 maka karakter tersebut akan mati atu kalah.

b). **Magic Point (MP)** adalah indikator jumlah dari jurus yang dapat dikeluarkan oleh pemain, jika MP bernilai 0 maka karakter tersebut tidak bisa mengeluarkan jurus atau kemampuan khusus.

c). **Strength** adalah jumlah atau nilai serangan yang akan dilakukan untuk mengalahkan pemain lawan. Angka tersebut akan berlawanan atau dibandingkan dengan jumlah *endurance* musuh. Hal tersebut bertujuan untuk mengurangi HP dari musuh.

d). **Magic** atau *Special Attack* biasanya tidak dimiliki oleh semua karakter pada permainan berbasis *turn-based*. *Special Attack* biasanya menjadi pembeda dalam setiap karakter berdasarkan jenis serangannya. Misalkan pada *strength* biasanya berupa serangan fisik sedangkan pada *special attack* berupa serangan *magic* atau sihir.

e). **Endurance** adalah jumlah atau nilai ketahanan yang digunakan oleh pemain atau musuh dalam menerima serangan. Hal ini bertu-

juan agar mencegah penurunan HP secara segnifikan, dengan membandingkan nilai serangan dengan nilai pertahanan.

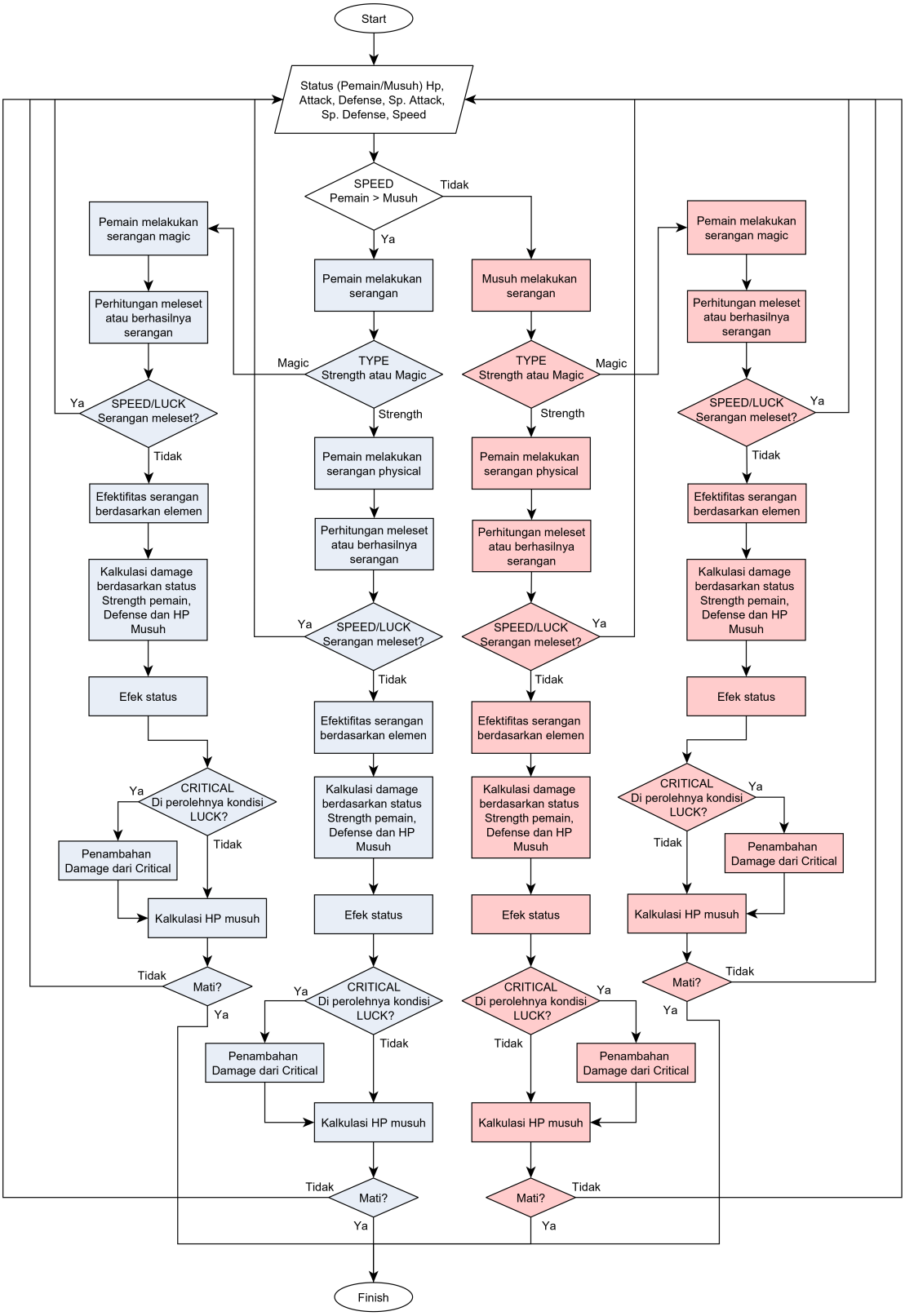
f). **Speed** atau kecepatan ada juga yang menyebutnya dengan *agility* bertujuan dalam menentukan giliran dan keberhasilan dalam melakukan serangan. Semakin tinggi nilai *speed*, biasanya semakin cepat melakukan serangan atau dapat mulai melakukan serangan lebih awal dibandingkan pemain atau musuh dengan nilai *speed* yang lebih kecil.

g). **Luck** atau keberuntungan adalah sebuah variabel yang digunakan menentukan hal yang bersifat acak, seperti bonus serangan atau *critical attack*, kesempatan saat melakukan serangan balik atau *counter attack* setelah diserang.

Sementara itu skenario pertarungan dengan melibatkan atribut *gameplay* dijelaskan pada Gambar 3.4 yang mana pemain atau musuh melakukan serangan berdasarkan *speed*. Semakin tinggi *speed* maka akan memperoleh giliran pertama untuk menyerang.

Di lanjutkan lagi dengan perhitungan dengan membandingkan *Speed*, dan *Luck* antara karakter pemain dengan musuh, pada proses tersebutlah yang menentukan apakah serangan dari pemain dapat diterima atau meleset terhadap musuh dan sebaliknya. Tingginya *Speed* pada masing-masing karakter dapat diartikan perbandingn antara kecepatan untuk menyerang dan menghindar, sedangkan *Luck* adalah faktor keberuntungan yang mempengaruhi serangan tersebut. Kemudian dilanjutkan lagi dengan kalkulasi *attack* dan *defense* antara karakter yang menyerang dan target. Dari hasil kalkulasi tersebut akan berpengaruh pada jumlah HP dari karakter yang menjadi target.

Baik pada permainan RPG yang dimainkan secara *real-time* atau *turn-based*, keduanya melakukan mekanisme sama seperti yang dijelaskan dengan kondisi di atas. Hanya saja permainan RPG secara *real-time*



Gambar 3.4: Skenario pertarungan *turn-based*.

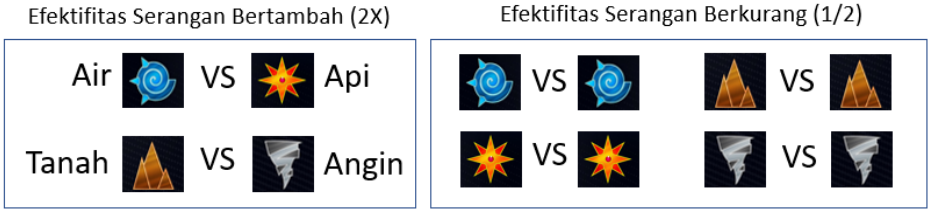
lebih mengandalkan ketangkasan dari pemain dalam memainkan karakter yang ingin dimainkan sepeti yang dijelaskan pada Sub-bab 2.2.2 pada bagian Peran dari keterampilan pada permainan. Maka terjadilah momen saling serang secara langsung antara pemain dan musuh.

2). Elemen dan Efektifitas Serangan

Pada Gambar 3.5 adalah contoh elemen yang biasa digunakan dalam permainan RPG. Jumlah dari elemen dapat ditambah atau dikurangi sesuai dengan kebutuhan. Biasanya jumlah dari elemen ditentukan berdasarkan cerita. Mengapa terdapat elemen tersebut, bagaimana asal-usulnya dan sebagainya seperti yang dijelaskan pada Sub-bab 3.1.2.

Gambar 3.5: Elemen pada permainan RPG.

Pembahasan ini mengacu kepada Gambar 3.4 pada bagian "efektifitas serangan berdasarkan elemen". Sedangkan pada Gambar 3.6 adalah perbandingan pengaruh atau keterkaitan sebuah elemen dengan elemen yang lain. Di mana setiap elemen memiliki kelemahan yang berupa elemen lain dan sebaliknya. Elemen-elemen tersebut saling berlawanan satu dengan yang lainnya sehingga mampu membentuk sebuah permainan yang membutuhkan strategi khusus. Kemudian dilanjutkan dengan perhitungan atribut *gameplay* seperti bagian sebelumnya.



Gambar 3.6: Pengaruh elemen pada efektifitas serangan.

Jika melihat pada Gambar 3.6 dapat disimpukan bahwa beberapa element yang saling berlawanan kemudian efektifitas menjadi 2 kali lipat, contohnya pada elemen air terhadap api dan tanah terhadap angin begitu juga sebaliknya. Jika elemen yang sama saling bertarung maka efektifitasnya berkurang 1/2 dari yang seharunya. Kemudian pada elemen lain yang belum disebutkan berlaku efektifitas normal atau 1 kali. Pembagian elemen pada pemain dan musuh akan dibahas lebih detail pada Sub-bab 3.2.

Kebanyakan elemen dan efektifitas serangan berlaku pada permainan RPG yang tergolong *turn-based*, berbeda halnya dengan *real-time* yang lebih mengandalkan keterampilan dari pemain dalam memaikan karakternya seperti yang dijelaskan pada Sub-bab 2.2.2 pada bagian pemain berbasis keterampilan sepenuhnya.

3). Efek Status

Pada Gambar 3.4 terdapat bagian "efek status", maksud dari proses tersebut adalah penambahan efek yang merugikan terhadap pemain setelah diserang. Efek kerusakan dapat membawa kesan lebih taktis pada pertempuran. Berikut adalah efek status yang akan diterapkan pada sistem pertarungan.

a). **Infected:** Karakter kehilangan 10% dari total HP setiap giliran.

b). **Confused:** Karakter tidak dapat dikendalikan dan mungkin akan bertahan, menyerang dan tidak melakukan apa-apa. Ada juga kemungkinan mereka akan berbalik menyerang *party member* mereka sendiri.

c). **Silence:** Karakter tidak dapat mengubah Personae atau menggunakan keterampilan Persona.

d). **Tired:** Karakter kehilangan SP untuk setiap tindakan yang diambil, dan menerima lebih banyak *damage* dari musuh ketika diserang.

e). **Hacked:** Efek yang muncul setelah target diserang sesuai dengan kelemahannya. Target kemudian akan menerima lebih banyak *damage* dan tidak dapat menghindar. Jika target diserang lagi, maka statusnya akan berubah menjadi disabled.

f). **Disabled:** Target akan hilang 1 giliran untuk mengambil tindakan, kemudian mendapat lebih banyak kerusakan dan serangan tidak dapat dihindari.

Tidak semua kemampuan pemain dapat memberi efek status, hal tersebut mengacu pada desain permainan yang mengatur keseluruhan *skill*, tidak hanya pada karakter utama melainkan juga pada musuh. Tetapi dalam penelitian ini, hal tersebut masih belum terpakai dikarenakan masih menghitung atribut *gameplay* dari pemain dan musuh. Hal ini baru semacam perkiraan saja saat mendesain sebuah permainan.

Pada bagian efek status juga berlaku pada permainan RPG dengan jumlah karakter tunggal, setelah berlangsungnya pertarungan antara pemain dan musuh. Di mana pada sisi pemain dapat memhangun karakternya sedemikian hingga demi memberi efek status ke pada musuh saat berlangsunnya pertarungan seperti yang dijelaskan pada Sub-bab 2.2.2 tentang peran dan keterampilan pemain serta strategi dan taktik.

4). Kondisi Kritis pada Serangan

Selain *attack*, elemen dan efek status, masih terdapat *damage* yang dapat ditimbulkan oleh penyerang kepada target yaitu kondisi kritis pada serangan. Dengan jumlah *attack* ditambah dengan jumlah *attack* yang dikalikan dengan presentase tingkat kondisi kritis (*critical rate*) pada *skill* yang dipilih untuk menyerang lawan. Kondisi tersebut didapat dengan membandingkan nilai *Strength* atau *Magic* dan *luck* milik penyerang dan target, pada proses tersebutlah yang menentukan apakah serangan tersebut diperoleh kondisi kritis atau tidak. Hal tersebut juga membangun sebuah momen dramatis berdasarkan peluang terjadinya kondisi kritis

saat terjadinya serangan dari pemain terlebih lagi dari musuh seperti yang dibahas pada Sub-bab 2.2.1 tentang momen dramatis.

5). Kalkulasi HP dan MP

Pada akhirnya jumlah kerusakan yang ditimbulkan oleh penyerang dan HP dari target akan dikalkulasikan. Jika HP target habis atau sama dengan 0, maka terget tersebut dinyatakan mati. Dan jika HP target masih bersisa, maka pertarunggan akan dilanjutkan oleh giliran karakter dari pemain atau musuh selanjutnya. Kemudian pada sisi penyerang juga ada yang dikorbankan dalam upaya melakukan serangan. Saat penyerang memilih serangan fisik makan pemain akan mengorbanakan sebagian HP yang dimiliki, jika yang dipilih adalah serangan *magic* maka yang dikorbankan adalah MP.

3.1.2 Hubungan Sistem Pertarungan dengan Cerita

Pada setiap permainan RPG khususnya *real-time* dan *turn-based* RPG pasti memiliki cerita yang menjadi latar belakang permainan seperti yang dijelaskan pada Sub-bab 2.4. Tentunya cerita tersebut juga memiliki pengaruh penting terhadap jumlah karakter yang dapat dimainkan oleh pemain, jumlah musuh, elemen apa saja yang akan dipakai, total waktu permainan, jumlah musuh yang harus dilawan dan lain sebagainya. Pada penelitian ini hal semacam itu akan dibuat menjadi sebuah estimasi yang kemudian disimulasikan ke dalam mode pertarungan dengan berbagai macam estimasi sebagaimana penjelasan berikut.

1). Tingkat Kesulitan Musuh

Berikut adalah beberapa pertanyaan yang harus ditanyakan kepada setiap pembuat desain permainan:

a). Haruskah kebanyakan pemain nantinya dapat menyelesaikan permainan tanpa melakukan *side quest* (misi sampingan) atau me-

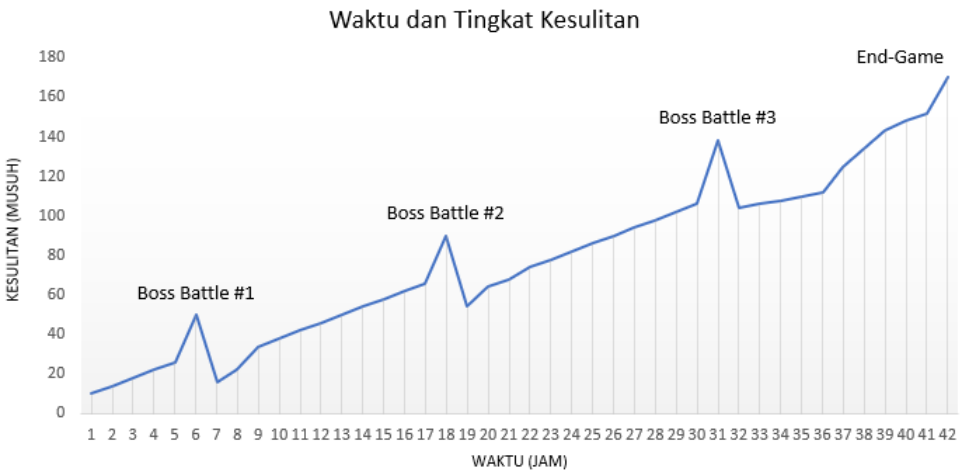
lakukan *grinding* (menaikan kemampuan karakter) diluar standar perkembangannya?

b). Berapa banyak bos yang akan dilawan oleh pemain, dan seberapa jauh jaraknya? Dan bagaimana dengan penambahan bos?

c). Berapa banyak *dungeon* (tempat muncul dan bertemu dengan musuh) yang akan disajikan, dan seberapa besar *dungeon* tersebut?

d). Akankah pemain bisa menyimpan progres permainan kapan saja, atau hanya di titik penyimpanan tertentu yang sudah ditentukan?

Biasanya pada permaianan RPG terdapat sebuah peta besar tentang lokasi yang merupakan latar dari cerita, tersebarlah berbagai jenis musuh yang relatif mudah dikalahkan. Kemudian ditampilkan juga beberapa *dungeon* yang akan terbuka satu demi satu, yang mana *dungeon* tersebut memiliki tingkat kesulitan dan kerumitan yang terus meningkat sampai dengan bertemu dengan Bos. Secara keseluruhan tingkat kesulitan juga akan terus meningkat sampai dengan akhir permaian seperti yang dicontohkan oleh Gambar 3.7.



Gambar 3.7: Pengaruh cerita terhadap tingkat kesulitan.

Adapun beberapa cara untuk meminimalisir *grinding* dan lamanya waktu permainan, dengan diberikan Exp (*Experience* adalah sebuah variabel untuk pemain agar naik level) kepada pemain untuk menyelesaikan

misi dan mengalahkan musuh yang lebih sulit dari pada mengalahkan musuh yang relatif mudah ditaklukan yang tersebar pada peta. Tentu saja musuh yang tersebar di peta juga memberikan Exp bagi pemain, namun seiring bertambahnya level pemain maka Exp yang diperoleh saat melawan musuh dengan level rendah akan semakin kecil.

Pada penelitian ini tingkat kesulitan langsung disimulasikan dengan pertarungan antara karakter-karakter yang dimainkan oleh pemain melawan musuh, dengan kondisi tingkat kesulitan musuh yang terus naik lalu turun kemudian naik lagi dan turun lagi, naik lagi dan seterusnya sampai dengan kondisi puncak. Hal ini mensimulasikan kondisi yang dilalui oleh pemain saat melawan *trash mobs*, memasuki *dungeon*, saat bertarung melawan bos dan kemudian pada akhirnya bertarung melawan bos terakhir. Lebih detailnya akan dijelaskan pada poin selanjutnya.

2). Waktu yang Diperlukan untuk Kalahkan Musuh

Musuh pada permainan RPG umumnya terbagi menjadi empat kategori diantaranya adalah:

A). *Trash Mobs* adalah musuh yang tersebar pada seluruh area atau map.

B). *Dungeon Mobs* dapat dibagi menjadi dua sub-kategori:

   a). *Dungeon Trash* atau sama seperti *Trash Mobs* yang sebagian besar ditemukan awal sampai tengah *dungeon*.

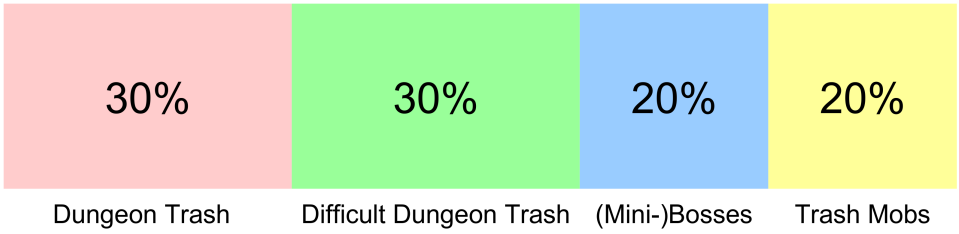
   b). *Difficult Dungeon Trash* atau yang lebih sulit terletak lebih dekat ke bos atau dari tengah ke akhir *dungeon*.

C). *Mini-Boss/Boss Mobs*.

D). *End-Game Boss/Secret Boss* (Bos yang bersifat opsional).

Pada penelitian ini keseimbangan permainan dirancang terus meningkat seperti yang dibahas pada bagian sebelumnya, dalam permainan

ini sebagian besar waktu (asumsikan saja 80%) akan dihabiskan bertarung di dalam *dungeon*. Kemudian 20% dari waktu pertempuran akan digunakan untuk bertarung melawan bos. Sedangkan sisanya 60% dari waktu bertarung akan dibagi antara pertarungan melawan musuh yang lemah dan juga kuat, bila digambarkan pembagian tersebut akan seperti pada Gambar 3.8.



Gambar 3.8: Distribusi jenis musuh sesuai dengan cerita.

Perhatikan bahwa dalam distribusi yang dibuat, jumlah waktu yang dihabiskan untuk melawan bos sama seperti waktu yang dibutuhkan untuk memerangi *Thrash Mobs*. Dalam proses replikasi distribusi dalam permainan, pertama tentukan jumlah total waktu yang dibutuhkan oleh pemain untuk mengalahkan naga, raksasa atau apa pun yang biasanya disebut dengan bos.

Misalnya, dalam permaian RPG, bos pertama idealnya akan membutuhkan 3 menit bagi pemain yang kompeten untuk mengalahkan, dan bos terakhir menghabiskan waktu 20 menit. Kemudian terjadi peningkatan kompleksitas pada bos di level menengah, untuk terdapat dua bos yang masing-masing membutuhkan waktu sekitar 10 menit untuk dikalahkan, dan bos kedua membutuhkan waktu 7 menit.

$$tB_{total} = \sum_{n=0}^{B} tB_n \tag{3.1}$$

Merujuk ke persamaan 3.1 bila dijabarkan maka tB_{total} adalah jumlah waktu melawan bos secara keseluruhan, jumlah bos dinyatakan deng-

an B dan waktu yang dihabiskan untuk melawan satu bos dinyatakan dengan tB kemudian diiterasi oleh n sejumlah B.

Di perlukannya penyesuaian terhadap model distribusi, salah satunya waktu yang habis untuk melawan bos setara untuk melawan musuh yang mudah atau *trash mobs*. Untung saja terdapat banyak cara untuk memodifikasi waktu yang akan dihabiskan saat bertarung melawan musuh yang mudah. Hal seperti menjelajah seluruh peta atau berpetualang menuju tempat-tempat sebelumnya juga tidak perlu dilakukan. Berikut adalah langkah-langkah yang dapat dilakukan:

a). Mengurangi atau meningkatkan tingkat pertemuan dengan musuh yang mucul secara acak atau yang biasa disebut dengan *random encounter rate*. Bisa juga dengan tingkat *spawn* (muncul lagi setelah mati).

b). Menambah atau mengurangi jumlah musuh saat pertempuran.

c). Membatasi kemampuan musuh yang menimbulkan efek status yang menyulitkan dan menghabiskan waktu seperti *confused*, *silence*, *tired* dan lain-lain.

d). Menambah atau mengurangi kekuatan musuh.

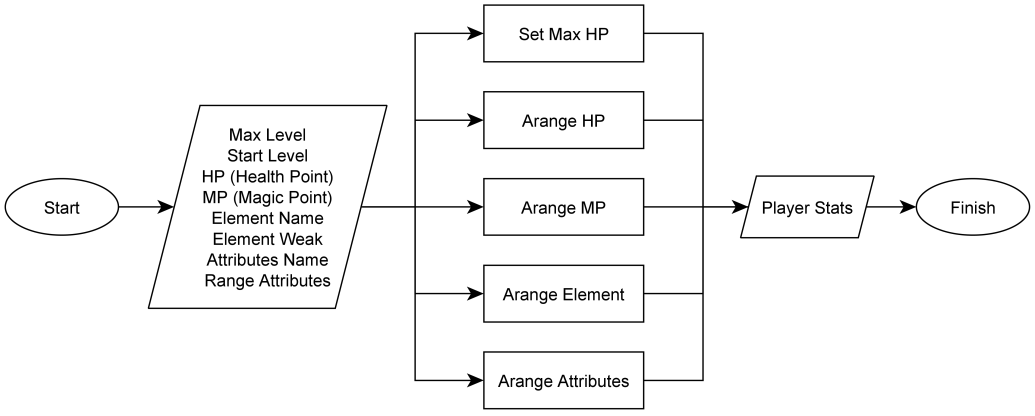
e). Menambah atau mengurangi kekuatan dari *party member*.

Terdapat banyak fleksibilitas di sini, developer dapat mengisi daftar musuh yang akan muncul secara acak dengan musuh yang sulit dikalahkan dengan tingkat probabilitas kemunculan yang kecil, atau lebih sering memunculkan musuh yang mudah dikalahkan. Alternatif lain adalah dengan meningkatkan kekuatan *party member* atau jumlah rata-rata musuh yang bertarung dalam satu kali pertarungan. Kebebasan desain semacam ini yang nantinya akan memudahkan penyeimbangan permainan. Sedangkan jumlah musuh dan panjangnya level dari pemain atau musuh sendiri menggambarkan akan lamanya permainan tersebut. Pada dasarnya semua proses diatas mengacu pada pokok

pembahasan dari referensi yang dibahas pada Sub-bab 2.2.3 tentang menemukan keseimbangan.

3.2 Pembuatan Atribut Gameplay Karakter Pemain

Di buatlah sebuah program yang secara otomatis dapat membuat atribut *gameplay* untuk karakter pemain dengan masukan sesuai dengan kebutuhan desainer permainan atau pengembang. Program tersebut terdiri dari beberapa fungsi yang pada awalnya adalah obyek yang memiliki masukan parameter yang nantinya akan menghasilkan sebuah data yang berupa atribut *gameplay* dari karakter pemain seperti proses yang ditunjukan oleh diagram alur pada Gambar 3.9.

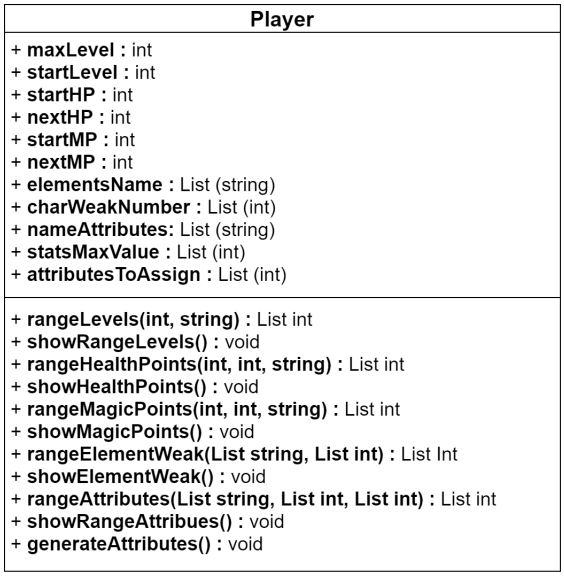


Gambar 3.9: Proses pembuatan atribut *gameplay* untuk karakter pemain.

Pada permainan dengan genre RPG dengan jumlah karakter yang dapat dimainkan berjumlah lebih dari satu, maka program yang ditunjukan melalui proses pada Gambar 3.9 akan dijalankan lebih dari satu kali. Hal tersebut dilakukan dengan tujuan agar karakter utama atau yang dapat dimainkan oleh pemain berjumlah lebih dari satu. Lain halnya dengan *action* RPG, cukup hanya dengan satu kali menjalankan program maka sudah diperolehnya atribut *gameplay* dari karakter yang dibuat. Hal tersebut dikarenakan biasanya permainan dengan genre action RPG, jumlah karakternya hanya satu.

Pada Gambar 3.9 disisi masukan program terdapat banyak sekali masukan variabel seperti *Max Level* yang berupa maksimum level yang di inginkan,

kemudian *Start Level* yang berupa level awal dari pemain, kemudian HP yang berupa *range* atau jarak antara HP terendah dengan HP terendah setelah itu. Sama halnya dengan HP, dalam perhitungan *range* atau jarak pada MP juga menggunakan cara tersebut. Kemudian *Element Name* berisi daftar elemen apa saja yang ingin digunakan, begitu juga dengan *Name Attributes*. Kemudian untuk *Range Attributes* berisikan nilai atribut *gameplay* awal atau inisialisasi dan maksimum atribut *gameplay*. Lebih detail tentang program yang dibuat dapat dilihat pada Gambar 3.10, yang merupakan *class* diagram untuk membuat atribut *gameplay* pemain.



Gambar 3.10: *Class* diagram untuk atribut *gameplay* pemain.

Karena program ini dibangun menggunakan OOP (*Object Oriented Programming*) maka program ini dapat dijabarkan menggunakan Gambar 3.10. Selain itu program ini juga dapat secara mudah dimodifikasi untuk keperluan pengembangan kedepannya, dengan fungsi-fungsi yang ada sangat memungkinkan dilakukan *override* atau pembuatan fungsi yang sama dengan isi atau proses yang berbeda.

Dalam menjelaskan proses pada BAB ini maka diambilah sebuah kasus dalam desain permainan, khususnya dalam penyusunan atribut *gameplay* untuk karakter dari pemain pada permainan RPG, baik itu permainan RPG

yang tergolong *real-time* atau *turn-based*. Pada Tabel 3.1 adalah masukan untuk menguji program yang akan dijelaskan pada bagian selanjutnya, yang mana masukan pada Tabel 3.1 akan menghasilkan atribut *gameplay* pada sebuah karakter untuk pemain. Hal ini merupakan perwujudan dari Sub-bab 3.1.1 bila diwujudkan dalam bentuk yang lebih teknis.

Tabel 3.1: Data masukan untuk pembuatan atribut *gameplay* pemain.

Variabel	Input
Start Level	1
Max Level	100
Start HP	159
Next HP	163
Start MP	89
Next MP	93
List Element	['Phys', 'Water', 'Wind', 'Earth', 'Fire']
List Weaknesess	[0, 0, 1, 2, 0]
List Attributes Name	['Strength', 'Magic', 'Endurance', 'Speed', 'Luck']
Max Attributes Value	[74, 38, 63, 65, 60]
Attributes to Assign	[2, 1]

3.2.1 Distribusi Level, HP dan MP Pemain

Berdasarkan Tabel 3.1 level untuk karakter pemain dimulai dari 1 dan level maksimalnya adalah 100, pada program ini level pemain akan terus naik satu demi satu level sampai ke tingkat maksimal. Selanjutnya adalah HP atau *Health Point* yang diberi masukan berupa *start HP*, bisa dinyatakan juga sebagai HP saat level satu.

Kemudian variabel lanjutannya adalah *next* HP atau HP pada level selanjutnya, misalkan level dua. Muncul sebuah pertanyaan berapa HP selanjutnya sampai dengan level ke 100. Cara semacam itu juga berlaku untuk perhitungan MP pada program ini, dengan pola masukan yang sama dengan HP yaitu

Start MP dan *Next* MP. Hal tersebut diwujudkan pada persamaan 3.2 dan 3.3 yang digunakan untuk mencari nilai HP dan MP selanjutnya.

$$HP' = \sum_{i=0}^{N_p} HP_{(i+1)} + \left(HP_{(i+1)} - HP_i\right) \tag{3.2}$$

$$MP' = \sum_{i=0}^{N_p} MP_{(i+1)} + \left(MP_{(i+1)} - MP_i\right) \tag{3.3}$$

Pada persamaan 3.2 dan 3.3, HP' adalah nilai HP yang dicari pada setiap levelnya yang dinyatakan dengan vektor menjadi $HP' = (HP_0, HP_1, ..., HP_i)$, yang mana N_p adalah level maksimum, i adalah level mulai dan $i + 1$ adalah level selanjutnya. Jadi pada tahap ini seperti yang dijelaskan pada Tabel 3.1 yang mana nilai i dan $i + 1$ sudah diketahui sebagai inisialisasi, masing-masing adalah HP_i dan $HP_{(i+1)}$. Penjelasan ini juga berlaku untuk mencari nilai MP', yaitu nilai MP yang dicari yang dinyatakan dengan $MP' = (MP_0, MP_1, ..., MP_i)$. Selanjutnya melalui persamaan 3.2 dan 3.3 dihasilkan data dan grafik seperti yang ditunjukan pada Sub-bab 4.1.1 dan 4.2.1.

3.2.2 Distribusi Elemen dan Kelemahan Pemain

Kemudian untuk variabel *List Element* berisi elemen apa saja yang akan diterapkan pada permainan tersebut, seperti yang ditunjukan oleh Tabel 3.1. Maksud dari variabel ini adalah memberi penjelasan dari kelemahan dan keunggulan dari pemain berdasarkan elemen, seperti yang sudah dijelasakan pada Sub-bab 3.1.1 tentang elemen dan efektifitas serangan. Di lanjutkan dengan variabel *List Weaknesses* yang memuat angka-angka yang bertujuan menggambarkan saat pemain menerima serangan dari lawan, berikut adalah penjelasan dari angka-angka tersebut.

a). **Angka 0** adalah *Normal* atau efek serangan bersifat normal tanpa tambahan bonus serangan. jadi antara serangan yang dilakukan setara dengan kerusakan yang diterima akibat serangan tersebut.

b). **Angka 1** adalah *Repel* atau memiliki sifat menghindari serangan atau bahkan menahan serangan tersebut. Sehingga tidak terjadinya kerusahaan bagi penerima serangan.

c). **Angka 2** adalah *Weaknesess* atau serangan tepat menyerang terhadap kelemahan dari pemain sehingga efek kerusakan atau *damage* menjadi lebih terasa, biasanya dua kali serangan normal.

Pembagian elemen pada pemain bersifat pada penelitian ini dibuat statis. Maksudnya elemen yang dari awal didefinisikan tidak akan berubah sampai akhir level. Kedepannya hal seperti inilah yang akan menjadi konsetrasi pengembangan program ini berikut juga desainer permainan. Cukup dengan melakukan *override*, maka diperolehlah fungsi baru yang bisa menghasilkan perubahan elemen di level tertentu.

Pada bagian selanjutnya adalah pembahasan mengenai pembagian atribut *gameplay* jika merujuk pada Tabel 3.1 dengan variabel *List Attributes Name* yang berisi nama atau info atribut *gameplay* dari pemain yang akan digunakan. Kemudian diikuti dengan variabel *Max Attributes Value* yang berisi nilai maksimum atribut *gameplay* yang akan dihasilkan. Lebih detailnya untuk bagian ini, akan dibahas secara khusus pada bagian tersendiri yaitu pada Sub-bab 3.2.3.

3.2.3 Distribusi Atribut Gameplay Pemain

Pada bagian ini akan dibahas tentang pembagian atribut *gameplay* dengan beracuan pada Tabel 3.1 dengan variabel *List Attributes Name* yang berisi nama atau info atribut *gameplay* dari pemain yang akan digunakan. Kemudian diikuti dengan variabel *Max Attributes Value* yang berisi nilai maksimum atribut *gameplay* yang akan dihasilkan. Pada tahap ini digunakannya metode $k-$Nearest Neighbor atau $k-$NN seperti yang sudah dijelaskan pada Sub-bab 2.3.1 dan Naive Bayes yang juga sudah dijelaskan pada Sub-bab 2.3.2.

Pada persamaan 2.4 yang kemudian disesuaikan dengan pertambahan nilai variabel *Attributes to Assign* dari Tabel 3.1, yang mana nilai tersebut

ditambahkan secara acak antara 2 dan 1 dengan perhitungan *class probability* pada persamaan 2.15 yang disesuaikan menjadi persamaan 3.4 dan 3.5. Hasil proses acak tersebut juga harus dibatasi jumlahnya dengan persamaan 3.6 yang akan sangat menentukan perhitungan pada persamaan 3.7.

$$P(C_{St=1}) = \frac{C_{St=1}}{(C_{St=1} \ + \ C_{St=2})} \tag{3.4}$$

$$P(C_{St=2}) = \frac{C_{St=2}}{(C_{St=1} \ + \ C_{St=2})} \tag{3.5}$$

$$D(MaxSt, \ St_i) = \sqrt{(MaxSt - St_i)^2} \tag{3.6}$$

$$St' = \begin{cases} \sum_{i=0}^{N_p} \ St_i, & saat \ D(MaxSt, \ St_i) \geqslant 2, \ St_i = 2 \\ \sum_{i=0}^{N_p} \ St_i, & saat \ D(MaxSt, \ St_i) \geqslant 1, \ St_i = 1 \\ 0, & lainnya \end{cases} \tag{3.7}$$

Pada persamaan 3.4 dan 3.5, variabel $P(C_{St=1})$ adalah probabilitas munculnya nilai dari variabel *Attributes to Assign* ke 1 dan $P(C_{St=2})$ adalah probabilitas munculnya nilai dari variabel *Attributes to Assign* ke 2, kemudian $C_{St=1}$ adalah jumlah angka satu dan $C_{St=2}$ adalah jumlah angka dua dari variabel *Attributes to Assign*. Pada dasarnya konsep munculnya nilai dari variabel *Attributes to Assign* adalah probabilitas yang merupakan dasar Naive Bayes seperti pada Sub-bab 2.3.3, kondisi tersebut menyatakan dengan probabilitas munculnya angka 0, 1, dan 2.

Setiap atribut *gameplay* hasil pengacakan tentunya juga akan dibatasi jumlahnya dengan melakukan *checking* pada maksimum atribut *gameplay*, hal tersebut dijelaskan pada persamaan 3.6, yang mana D adalah jarak antara x atau maksimum atribut *gameplay* dan p adalah atribut *gameplay* saat ini. Kemudian nilai $MaxSt$ pada persamaan 3.6 dan 3.7 adalah nilai maksimum atribut *gameplay* yang ingin dicapai. Setiap kenaikan level dari pemain dinyatakan dengan St_i yang nilainya akan terus naik sampai dengan level maksimum yang dinyatakan N_p dan untuk setiap levelnya dinyatakan dengan i. Pada St_i

yang berupa nilai atribut *gameplay* yang bertambah setiap levelnya diperoleh dari persamaan 3.4, 3.5 dan pertambahannya dibatasi 3.6. Hasil dari proses tersebut disimpan dalam bentuk vektor pada setiap levelnya yang dinyatakan dengan $St' = (St_0, St_1, ..., St_i)$.

$$P(C_{St=i_{st}}) = \frac{C_{St=i_{st}}}{(C_{St=0} \ + \ C_{St=1} \ + \ ... \ + \ C_{St=i_{st}})} \tag{3.8}$$

$$D(MaxSt, \ St_i) = \sqrt{(MaxSt - St_i)^2} \tag{3.9}$$

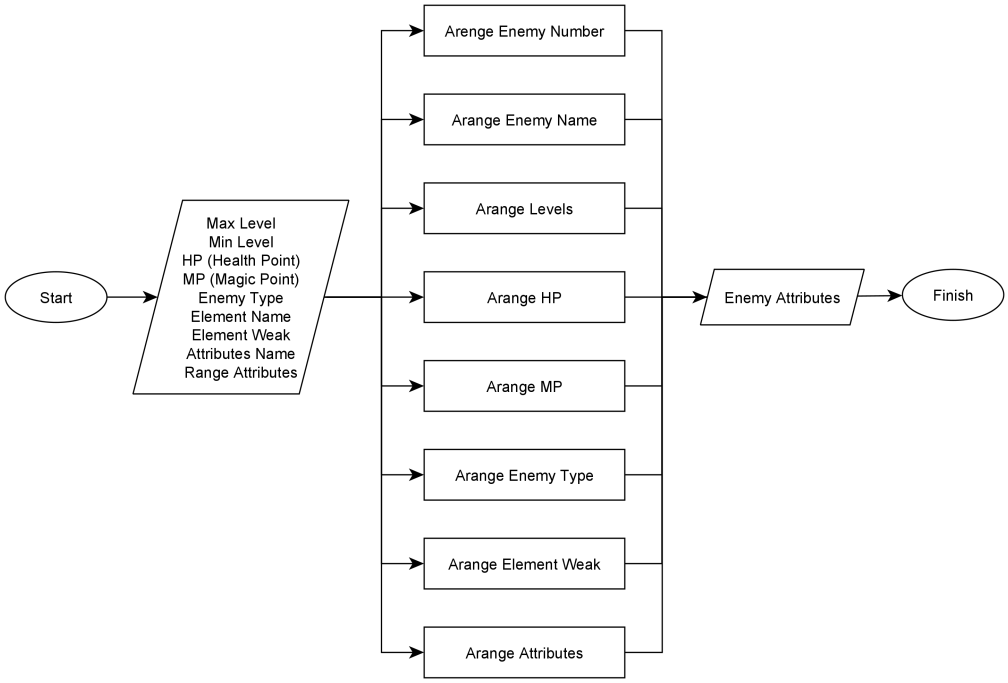
$$St' = \begin{cases} \sum_{i=0}^{N_p} \ St_i, \ saat \ D(MaxSt, \ St_i) \geqslant i_{st}, \ St_i = i_{st} \\ \sum_{i=0}^{N_p} \ St_i, \ saat \ D(MaxSt, \ St_i) \geqslant 2, \ St_i = 2 \\ \sum_{i=0}^{N_p} \ St_i, \ saat \ D(MaxSt, \ St_i) \geqslant 1, \ St_i = 1 \\ \qquad 0, \ lainnya \end{cases} \tag{3.10}$$

Sedangakan pada persamaan 3.8, 3.9 dan persamaan 3.10 digunakan saat jumlah atau dimensi *Attributes to Assign* seperti pada Tabel 3.1 yang berjumlah lebih dari dua, dengan ditambahkannya variabel untuk atribut *gameplay* yang ingin ditambahkan dinyatakan dengan i_{st}. Melalui persamaan 3.7 dan beberapa persamaan yang digunakan sebelumnya pada Sub-bab 3.2.3 dihasilkan data seperti yang ditunjukan pada Sub-bab 4.1.3 dan Sub-bab 4.2.3. Pada persamaan 3.6 dan 3.9 adalah penggunaan perbandingan *distance* atau jarak yang merupakan dasar dari k-NN, antara setiap kenaikan nilai atribut *gameplay* dan maksimum atribut *gameplay* yang kemudian menghasilkan grafik data berupa jumlah kenaikan atribut *gameplay* setiap levelnya seperti yang dijelaskan pada Sub-bab 4.1.3.

3.3 Pembuatan Atribut Gameplay Karakter Musuh

Sama halnya dengan karakter pemain maka dibuatlah sebuah program yang secara otomatis dapat membuat atribut *gameplay* musuh dengan masukan sesuai dengan kebutuhan desainer permainan atau pengembang. Program tersebut terdiri dari beberapa fungsi yang pada awalnya adalah objek yang

memiliki masukan parameter-parameter yang nantinya akan menghasilkan sebuah data yang berupa atribut *gameplay* dari karakter pemain seperti proses yang ditunjukan oleh diagram alur sederhana pada Gambar 3.11.



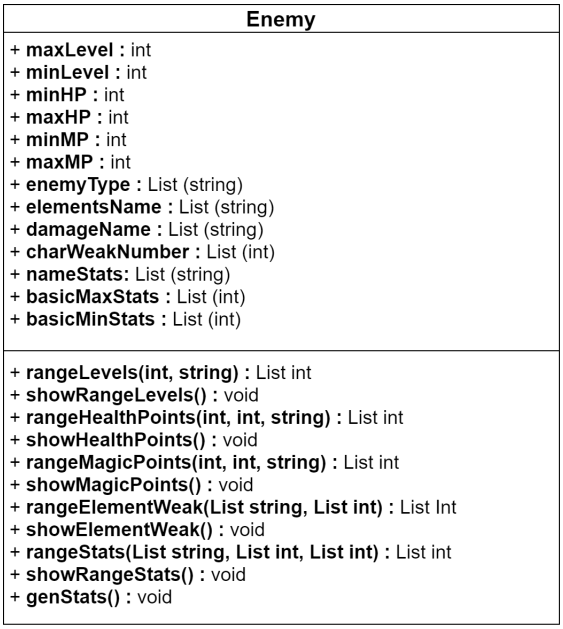
Gambar 3.11: Proses pembuatan atribut *gameplay* untuk karakter musuh.

Pada permainan bergenre RPG yang memiliki jumlah yang sangat banyak dan beragam, maka program yang ditunjukan melalui proses pada Gambar 3.11 dapat dijalankan satu kali saja, dan menghasilkan banyak musuh. Kecuali ingin menghasilkan kombinasi musuh yang berbeda, misalnya pada proses *generate* yang pertama menghasilkan musuh yang memiliki kemampuan *magic* dan kelemahan sedangkan pada kombinasi musuh selanjutnya tidak memiliki kemampuan *magic*, hanya mengandalkan kemampuan fisik saja.

Pada Gambar 3.11 di sisi masukan program terdapat banyak sekali masukan variabel seperti *Max Level* yang berupa maksimum level yang diinginkan, kemudian *Min Level* yang berupa minimum level dari musuh, kemudian HP yang berupa *range* atau jarak antara HP minimum dengan HP tertinggi. Sama halnya dengan HP, dalam perhitungan *range* atau jarak, pada *MP* juga menggunakan perhitungan dengan cara tersebut.

Kemudian *Enemy Type* yang berisi klasifikasi jenis atribut *gameplay* musuh, tergolong musuh seperti apakah atribut *gameplay* yang dihasilkan tersebut. Selanjutnya adalah *Element Name* yang berisi daftar elemen apa saja yang dapat dipilih saat pembuatan karakter musuh, selanjutnya *Element Weak* berisi tetang kelemahan dan keunggulan dari musuh tersebut ketika diserang, apakah saat diserang dengan menggunakan elemen tersebut akan mengalami kerusakan atau *damage* yang parah, normal atau tidak mempan sama sekali.

Sedangkan untuk *Attributes Name* berisikan nama atribut *gameplay* yang dipakai dalam membuat karakter musuh. Selanjutnya adalah isi atau data dari atribut *gameplay* itu sendiri yang menentukan karakter dari musuh itu sendiri, seberapa kuat musuh tersebut dalam menyerang atau bertahan dan lain sebagainya. Lebih detail tentang program yang dibuat dapat dilihat pada Gambar 3.12, yang merupakan *class* diagram pembuatan atribut *gameplay* musuh.



Gambar 3.12: *Class* diagram untuk atribut *gameplay* musuh.

Karena program ini dibangun menggunakan OOP (*Object Oriented Programming*) maka program ini dapat dijabarkan menggunakan Gambar 3.12. Selain itu program ini juga dapat secara mudah dimodifikasi untuk keperluan

pengembangan kedepannya, dengan fungsi-fungsi yang ada sangat memungkinkan dilakukan *override* atau pembuatan fungsi yang sama dengan isi atau proses yang berbeda seperti pada penjelasan pada pembuatan atribut *gameplay* pemain pada Sub-bab 3.2.

Seperti pada bagian sebelumnya pada Sub-bab 3.2 dalam pembuatan atribut *gameplay* karakter pemain, maka dibuatlah Tabel 3.2 yang berupa masukan untuk menguji program yang akan dijelaskan pada bagian-bagian selanjutnya, yang mana masukan pada tabel tersebut akan menghasilkan atribut *gameplay* pada sebuah karakter untuk pemain. Sama seperti pada Sub-bab 3.2 sebelumnya, yang mana pada bagian ini juga merupakan perwujudan teknis dari Sub-bab 3.1.1 dalam bentuk yang lebih teknis.

Tabel 3.2: Data masukan untuk pembuatan program pada musuh.

Variabel	Input
Enemy Numbers	400
Max Level	80
Min Level	1
Level Class	['*Easy*', '*Medium*', '*Hard*']
Min HP	159
Max HP	163
Min MP	89
Max MP	93
Enemy Type	['*Mixed*', '*Hard Magic*', '*Soft Magic*', '*Hard Strength*', '*Soft Magic*']
Distribution Percentage	[40, 10, 20, 10, 20]
List Element	['*Phys*', '*Water*', '*Wind*', '*Earth*', '*Fire*']
List Damage	['*Normal*', '*Repel*', '*Weak*']
List Attributes Name	['*Strength*', '*Magic*', '*Endurance*', '*Speed*', '*Luck*']
Max Attributes Value	[50, 60, 40, 55, 45]

Min Attributes Value	[2, 2, 2, 2, 2]

3.3.1 Distribusi Level Musuh

Pada bagian ini akan dijelaskan tentang pembagian level pada musuh, dengan masukan seperti yang disebutkan pada Tabel 3.2. Beracuan pada tabel tersebut beberapa variabel utama yang akan digunakan diantaranya adalah "*Enemy Numbers*" yang menentukan jumlah musuh yang akan dibuat, "*Max Level*" dan "*Min Level*" adalah nilai maksimum dan minimum level musuh. Selanjutnya tingkat kesulitan dari musuh ditentukan dengan variabel "*Level Class*" yang isinya dibagi menjadi "*Easy*", "*Medium*" dan "*Hard*". Musuh dengan tingkat kesulitan *"Easy"* akan menjadi yang paling mudah dikalahkan, diikuti dengan *"Medium"* dan "*Hard*" secara berurutan.

Kemudian terdapat variabel pendukung yang akan menentukan data atau level yang ingin dihasilkan, yaitu *scale* yang disebutkan pada Tabel 3.2. Selanjutnya dilakukan beberapa proses seperti pada persamaan 3.11, 3.12, 3.13, dan persamaan 3.14 yang kemudian diperoleh hasil berupa level untuk banyak musuh sekaligus seperti yang ditunjukan pada persamaan 3.15.

$$SL' = \begin{cases} \sum_{i=0}^{N_{LC}} \frac{LN}{N_{LC}} & Saat\ 0 \equiv LN\ (mod\ LC_N) \\ \sum_{i=0}^{N_{LC}} \frac{LN}{N_{LC}} & Saat\ 0 \not\equiv LN\ (mod\ LC_N), \\ & 0 \not\equiv N_{LC}\ (mod\ 2) \quad \Rightarrow\ SL_{(\lceil NL_C/2 \rceil)} = \left\lceil \frac{LN}{N_{LC}} \right\rceil \\ & \quad \Rightarrow\ SL_i = \left\lfloor \frac{LN}{N_{LC}} \right\rfloor \\ \sum_{i=0}^{N_{LC}} \frac{LN}{N_{LC}} & Saat\ 0 \not\equiv LN\ (mod\ N_{LC}), \\ & 0 \equiv N_{LC}\ (mod\ 2) \quad \Rightarrow\ SL_{NL_C/2},\ SL_{(NL_C/2)+1} = \left\lceil \frac{LN}{N_{LC}} \right\rceil \\ & \quad \Rightarrow\ SL_i = \left\lfloor \frac{LN}{N_{LC}} \right\rfloor \end{cases} \tag{3.11}$$

Pada persamaan 3.11 adalah proses pembagian *range* atau banyaknya level yang dijelaskan pada Tabel 3.2 pada variabel *Max Level* dan *Min Level* sebagai *range* atau banyaknya level, dinyatakan dengan LN yang kemudian dibagi dengan N_{LC} yang merupakan jumlah kelompok atau *cluster* dari jumlah data yang ada pada variabel *Level Class*. Jika beracuan pada Tabel 3.2,

maka jumlah *cluster* adalah data yang ada pada varaibel "*Level Class*" seperti "*Easy*", "*Medium*" dan "*High*", jadi jumlahnya adalah 3 *cluster*. Maka pada kasus yang dicontohkan ini LC atau *level cluster* menjadi $LC = (0, 1, 2)$, saat jumlah *cluster* yang ingin dibuat lebih dari contoh, maka akan menjadi $LC = (0, 1, ..., N_{LC})$ dengan N_{LC} adalah jumlah *cluster* yang ingin dibuat.

Dalam proses pencarian SL atau variabel vektor yang terdiri dari nilai pada setiap *cluster* level atau banyaknya level dari setiap *cluster* menjadi $SL' = (SL_0, SL_1, ..., SL_i)$ yang dicontohkan terbagi menjadi tiga seperti $SL' = (SL_0, SL_1, SL_3)$. Kemudian terdapat beberapa keputusan yang harus dijalankan, seperti hasil bagi antara LN dan N_{LC} saat bernilai bilangan tidak bulat, maka harus dilakukan proses pembulatan dalam pembagian banyaknya level pada setiap cluster atau SL_i. Sepeti pada persamaan 3.11, perlunya dilakukan pengecekan, apakah hasil bagi antara LN dan N_{LC} berupa bilangan bulat atau tidak dengan menggunakan operasi modulus atau *mod*. Jika LN habis terbagi dengan N_{LC}, maka setiap SL_i akan diisi secara merata oleh hasil bagi antara LN dan N_{LC}. Banyaknya level akan dibagi ke setaip *cluster* adalah sebesar LN dibagi dengan LC_N yang kemudian hasil setiap *cluster* disimpan dalam variabel SL' yang berbentuk vektor.

Namun jika tidak dapat dibagi secara merata, maka dilakukan terlebih dahulu proses pengecekan apakah jumlah N_{LC} berjumlah ganjil atau genap, hal tersebut dilakukan dengan cara melakukan modulus dari N_{LC} dengan angka 2. Jika ternyata N_{LC} bernilai genap, maka LN atau *range* level akan dibagi menjadi dua bagian, kemudian dua nilai tengah setelah dibagi menjadi dua bagian tersebut diisi dengan sebaran level yang lebih banyak jika dibandingkan dengan *cluster* level yang lain. Kemudian jika LC_N bernilai ganjil maka nilai tengah dari LN tersebutlah yang akan memiliki sebaran level lebih banyak jika dibandingkan dengan *cluster* level yang lain.

Munculah pertanyaan seperti berapa banyaknya level atau SL_i pada dua *cluster* level tengah pada kasus N_{LC} dengan nilai genap, dan berapakah SL_i pada *cluster* level pada bagian tengah untuk kasus N_{LC} dengan nilai ganjil.

Seperti pada persamaan 3.11 jika pada kondisi genap maka dua *cluster* level bagian tengah diisi dengan level sejumlah hasil pembagian LN dengan N_{LC} yang dibulatkan ke atas atau *ceil*, sedangkan pada kondisi ganjil maka satu *cluster* level bagian tengahlah yang diisi dengan hasil pembagian tersebut. Kemudian untuk banyaknya level pada *cluster* yang lain diisi dengan pembagian LN dengan LC_N yang dibulatkan ke bawah atau *floor*. Jika SL adalah set *cluster* level, maka perlu dicari juga jumlah cluster untuk musuh dengan menggunakan persamaan 3.12 dengan cara sama seperti pencarian jumlah *cluster* level.

$$SE' = \begin{cases} \sum_{i=0}^{LC_N} \frac{EN}{LC_N} & Saat\ 0 \equiv EN\ (mod\ LC_N) \\ \sum_{i=0}^{LC_N} \frac{EN}{LC_N} & Saat\ 0 \not\equiv EN\ (mod\ LC_N), \\ & 0 \not\equiv LC_N\ (mod\ 2) \quad \Rightarrow\ SE_{(\lceil NL_C/2 \rceil)} = \left\lceil \frac{EN}{LC_N} \right\rceil \\ & \quad \Rightarrow\ SE_i = \left\lfloor \frac{EN}{LC_N} \right\rfloor \\ \sum_{i=0}^{LC_N} \frac{EN}{LC_N} & Saat\ 0 \not\equiv EN\ (mod\ LC_N), \\ & 0 \equiv LC_N\ (mod\ 2) \quad \Rightarrow\ SE_{NL_C/2},\ SE_{(NL_C/2)+1} = \left\lceil \frac{EN}{LC_N} \right\rceil \\ & \quad \Rightarrow\ SE_i = \left\lfloor \frac{EN}{LC_N} \right\rfloor \end{cases} \tag{3.12}$$

Dalam proses pencarian SE atau *cluster* musuh yang berupa variabel vektor pada persamaan 3.12 digunakan cara yang sama seperti cara perhitungan pada persamaan 3.11 yang membagi musuh kedalam *cluster* menjadi $SE' = (SE_0, SE_1, ..., SL_i)$, yang dicontohkan menjadi tiga bagian $SE' = (SE_0, SE_1, SE_3)$, dan dilanjutkan dengan pengambilan beberapa keputusan yang harus dijalankan. Sama seperti pada perhitungan hasil bagi antara LN dan N_{LC} pada persamaan 3.11, nilai pembagian EN atau banyaknya musuh yang dibagi dengan LC_N atau jumlah *cluster* level yang ingin dibuat dengan tiga tingkatan "*Easy*", "*Medium*" dan "*Hard*" yang juga harus bernilai bilangan bulat positif, kemudian dilakukan juga proses pembulatan jumlah musuh terlebih dahulu dikarenakan jumlah musuh tidak boleh bernilai angka yang tidak bulat.

Sepeti pada persamaan 3.11, pada persamaan 3.12 juga dilakukan pengecekan apakah dapat dibulatkan atau tidak dengan melakukan operasi modulus

atau *mod*, jika habis maka SE' akan diisi secara merata oleh hasil bagi antara EN dan LC_N. Maka jumlah *cluster* sebanyak N pada LC_N dan banyaknya musuh adalah sebesar EN yang dibagi dengan LC_N, kemudian hasil akhir tersebut disimpan dalam variabel SE'. Untuk penjelasan langkah selanjutnya pada persamaan 3.12 sama persis dengan persamaan 3.11 yang sudah dijelaskan pada bagian sebelumnya, hanya saja objek pada persamaan 3.11 adalah level sedangkan pada persamaan 3.12 adalah musuh. Setelah SL' dan SE' diperoleh maka saatnya menuju bagian yang lebih dalam dan detail lagi. Bagaimana dengan pemberian level untuk setiap musuh, maka digunakanlah persamaan 3.13 dan 3.14 dengan penjelasan sebagai berikut.

$$SSL' = \begin{cases} \sum_{i=0}^{N_{LC}} \sum_{j=0}^{Sc} \frac{SL_i}{Sc} & Saat\ 0 \equiv SL_i\ (mod\ Sc) \\ \sum_{i=0}^{N_{LC}} \sum_{j=0}^{Sc} \frac{SL_i}{Sc} & Saat\ 0 \not\equiv SL_i\ (mod\ Sc), \\ & 0 \not\equiv Sc\ (mod\ 2) \quad \Rightarrow\ SSL_{(\lceil Sc/2 \rceil)} = \lceil \frac{SL_i}{Sc} \rceil \\ & \quad \Rightarrow\ SSL_j = \lfloor \frac{SL_i}{Sc} \rfloor \\ \sum_{i=0}^{N_{LC}} \sum_{j=0}^{Sc} \frac{SL_i}{Sc} & Saat\ 0 \not\equiv SL_i\ (mod\ Sc), \\ & 0 \equiv Sc\ (mod\ 2) \quad \Rightarrow\ SSL_{Sc/2},\ SSL_{(Sc/2)+1} = \lceil \frac{SL_i}{Sc} \rceil \\ & \quad \Rightarrow\ SSL_j = \lfloor \frac{SL_i}{Sc} \rfloor \end{cases} \tag{3.13}$$

Setelah diperolehnya SL dan SE yang masing-masing adalah variabel vektor untuk *cluster* level dan musuh, maka dicarilah *sub-cluster* level yang dinyatakan dengan SSL'. Bentuk dari SSL' adalah vektor yang akan memuat beberapa variabel *sub-cluster* level di dalamnya yang dinyatakan dengan SSL_i pada persamaan 3.13. Hal ini betujuan untuk mempersempit *range* dalam pembagian level musuh. Seperti pada penjelasan sebelumnya terdapat satu variabel lagi yang mempengaruhi proses ini, variabel tersebut adalah Sc atau skala yang digunakan untuk mentukkan kerapatan pembagian level pada musuh.

Jadi pada dasarnya persamaan 3.13 dijalankan setelah nilai SL' diperoleh dan jumlah nilai SSL' yang dapat berubah mengikuti nilai dari skala atau Sc yang menjadi salah satu masukan pada program ini seperti yang ditulis pada persamaan 3.13 dan 3.14. Perhitungan SSL' pada dasarnya sama

dengan perhitungan dalam mencari SL' atau SE' pada bagian sebelumnya, hanya saja pada bagian sebelumnya jumlah vektor atau *cluster* level dan musuh dipengaruhi oleh N_{LC} sedangkan pada SSL' jumlah set *sub-cluster* level dipengaruhi oleh Sc. Pada SL' yang terdiri dari beberapa *cluster* di dalamnya yang dinyatakan dengan SL_i, kemudian pada SL_i itulah baru dilakukan operasi pencarian SSL'. Pada SSL' sendiri kemudian dipecah lagi menjadi bagian yang lebih kecil atau *sub-cluster* yang dinyatakan dengan SSL_j.

Kemudian dilakukan juga pengecekan dengan operasi modulus atau *mod* pada setiap SL ke i dengan Sc, apakah SL_i akan habis jika dibagi dengan Sc. Jika ternyata SL_i habis dibagi dengan Sc, maka level pada range SSL atau *sub-cluster* tersebut akan langsung dibagi secara merata. Sedangkan pada kondisi sebaliknya yaitu saat SL_i tidak habis dibagi dengan Sc maka akan dilakukan proses pembulatan ganjil dan genap yang dibulatkan ke atas atau *ceil* seperti pada persamaan 3.11 dan persamaan 3.12, kemudian untuk nilai yang lain dibulatkan ke bawah atau *floor*.

$$SSE' = \begin{cases} \sum_{i=0}^{N_{LC}} \sum_{j=0}^{Sc} \frac{SE_i}{Sc} & Saat\ 0 \equiv SE_i\ (mod\ Sc) \\ \sum_{i=0}^{N_{LC}} \sum_{j=0}^{Sc} \frac{SE_i}{Sc} & Saat\ 0 \not\equiv SE_i\ (mod\ Sc), \\ & 0 \not\equiv Sc\ (mod\ 2) \quad \Rightarrow\ SSE_{(\lceil SL/2 \rceil)} = \left\lceil \frac{SE_i}{Sc} \right\rceil \\ & \quad \Rightarrow\ SSE_j = \left\lfloor \frac{SE_i}{Sc} \right\rfloor \\ \sum_{i=0}^{N_{LC}} \sum_{j=0}^{Sc} \frac{SE_i}{Sc} & Saat\ 0 \not\equiv SE_i\ (mod\ Sc), \\ & 0 \equiv Sc\ (mod\ 2) \quad \Rightarrow\ SSE_{N/2},\ SSE_{(N/2)+1} = \left\lceil \frac{SE_i}{Sc} \right\rceil \\ & \quad \Rightarrow\ SSE_j = \left\lfloor \frac{SE_i}{Sc} \right\rfloor \end{cases} \tag{3.14}$$

Kemudian untuk persamaan 3.14, sama seperti persamaan 3.13 hanya saja yang menjadi subjek disini adalah variabel vektor untuk *sub-cluster* atau bagian kumpulan jumlah musuh yang dinyatakan dengan SSE' dari setiap *cluster* atau kumpulan musuh yang dinyatakan dengan SE_i. Kemudian cara perhitungannya sama seperti persamaan 3.13, tetapi yang dibagi disini adalah setiap *cluster* atau SE_i dari musuh dibagi dengan skala atau Sc. Jadi dari setiap SE_i itu yang akan membentuk SSE' yang kemudian dibagi dengan Sc menjadi SSE_i.

Kemudian dilakukan juga pengecekan dengan operasi modulus atau *mod* pada setiap SE_i dengan Sc, apakah SE_i akan habis jika dibagi dengan Sc. Jika ternyata pada SE ke i habis dibagi dengan Sc, maka SE atau banyaknya musuh dalam *sub-cluster* ke i akan langsung dibagi secara merata ke dalam set *sub-cluster* . Sedangkan pada kondisi sebaliknya yaitu saat SE_i tidak habis dibagi dengan Sc maka akan dilakukan proses pembulatan ganjil dan genap yang dibulatkan ke atas atau *ceil* seperti pada persamaan 3.11 dan persamaan 3.12, kemudian untuk nilai yang lain dibulatkan ke bawah atau *floor*.

Saat semua sudah selesai dilakukan, khususnya yang ada pada persamaan 3.11 dan persamaan 3.12 tentang eksekusi setiap set *cluster* level dan musuh yang dilanjutkan dengan eksekusi setiap set *sub-cluster* level dan musuh, pada proses tersebutlah distribusi level pada setiap musuh dilakukan. Pada persamaan 3.15 adalah penjelasan tentang peluang diperolehnya level pada setiap musuh, yang beracuan pada metode *Naive Bayes* seperti yang sudah dijelaskan pada Sub-bab 2.3.3.

$$P(Elv_{jk}) = \sum_{i=0}^{SSL} \sum_{j=0}^{SSE} \sum_{k=0}^{SSE_j} \frac{Elv_{jk}}{SSL_{(i+1)} - SSL_i} \tag{3.15}$$

Pada persamaan 3.15 $P(Elv_{jk})$ adalah peluang munculnya level musuh yang dinyatakan dengan Elv, sedangkan SSL adalah set dari *sub-cluster* level yang merupakan alokasi persebaran level untuk musuh. Kemudian untuk SSL ke i dan SSL ke $(i+1)$ adalah range persebaran level untuk karakter musuh dengan alokasi setiap *sub-cluster* banyaknya musuh atau SSE_j yang kemudian setiap karakter musuh dalam *sub-cluster* SSE_j tersebut dinyatakan dengan Elv ke k. Hasil dari proses ini kemudian dijelaskan pada Sub-bab 4.3.1.

Selanjutnya adalah validasi dari keseimbangan persebaran level musuh, hal ini dilakukan dengan menggunakkan beberapa langkah seperti yang ditunjukan pada persamaan 3.16, 3.17, 3.18 dan persamaan 3.19. Konsep tersebut beracuan pada Sub-bab 2.3.4 tentang *Gaussian Naive bayes*, dengan harapan apakah setiap data yang dihasilkan sebelumnya sudah terdistribusi dengan

normal atau tidak. Penjelasaan berikut ini adalah beberapa langkah untuk validasi keseimbangan persebaran level musuh.

$$\bar{E}lv = \frac{\sum_{i=0}^{EN} Elv_i}{EN} \tag{3.16}$$

Adapun urutan metode dalam penggunaan *Gaussian Naive Bayes* adalah dengan mencari rata-rata dari data tersebut, kemudian diikuti dengan perhitungan standar deviasi. Pada persamaan 3.16 adalah rata-rata dari level yang dihasilkan sama seperti pembahasan pada persamaan 2.25 yang kemudian disimbolkan dengan variabel $\bar{E}lv$. Selanjutnya adalah $\bar{E}lv_i$ adalah setiap level dari musuh yang terus dijumlahkan sebanyak LE, yang mana EN sendiri adalah banyaknya karakter musuh yang sudah dibuat. Setelah diperoleh rata-rata level maka dapat dilanjutkan dengan pencarian nilai *standar deviasi* seperti pada persamaan 3.18.

$$\sigma(Elv)^2 = \frac{\sum_{i=0}^{EN} (Elv_i - \bar{E}lv)^2}{EN} \tag{3.17}$$

$$\sigma(Elv) = \sqrt{\frac{\sum_{i=0}^{EN} (Elv_i - \bar{E}lv)^2}{EN}} \tag{3.18}$$

Pada persamaan 3.18 adalah persamaan untuk mencari standar deviasi atau $\sigma(Elv)$ setelah diperolehnya rata-rata dari persebaran karakter musuh yang dinyatakan dengan $\bar{E}lv$ melalui persamaan 3.16 yang kemudian dilanjutkan dengan pencarian varian atau $\sigma(Elv)^2$ melalui persamaan 3.17 yang selanjutnya diakar kuadratkan untuk memperoleh nilai standar deviasi. Jika standar deviasi, varian dan rata-rata sudah diperoleh maka dapat dilanjutkan menuju pencarian nilai distribusi normal melalui *Gausian Naive Bayes* atau Gaussian PDF (*Probability Density Function*) seperti pada persamaan 3.19 berikut ini. Hasil dari proses ini kemudian juga dijelaskan pada Sub-bab 4.3.1, yang dibuktikan dengan persebaran musuh yang terdistribusi secara normal.

$$PDF(ELv,\ \bar{E}lv,\ \sigma) = \frac{1}{\sqrt{2\pi}\sigma}\ exp\left(-\frac{(Elv - \bar{E}lv)^2}{2\sigma^2}\right) \tag{3.19}$$

Konsep pada persamaan 3.19 sudah sangat dijelaskan pada Sub-bab 2.3.4. Hasil perhitungaan sebelumnya yang berupa varian σ^2, rata-rata $\bar{E}lv$ dan standar deviasi σ menjadi masukan pada persamaan tersebut. Variabel π adalah konstanta numerik pada umumnya, kemudian fungsi $exp()$ atau e adalah konstanta numerik yang betujuan untuk membentuk hasil prediksi dengan pendekatan exponensial.

3.3.2 Distribusi Tipe Musuh

Di bagian distribusi tipe musuh akan dijelaskan tentang pembagian tipe musuh, dengan masukan seperti yang disebutkan pada Tabel 3.2. Beracuan pada tabel tersebut beberapa variabel utama yang akan digunakan diantaranya adalah *Enemy Type* yang menjelaskan tipe musuh yang akan dibuat. Kemudian *Distribute Percentage* adalah persentase distribusi tipe yang ingin dibuat atau prentase dari *Enemy Type* secara berurutan. Kemudian variabel level musuh yang diambil dan dijelaskan pada Sub-bab 3.3.1 juga turut menentukan tipe musuh yang akan dibuat.

Terdapat juga variabel pendukung yang membantu proses distribusi tipe, diantaranya adalah *Distribute Number* dan *Distribute Level* yang keduanya perlu dicari nilainya melalui persamaan 3.20 dan 3.21. Kemudian dilanjutkan denagn beberapa proses seperti pada persamaan 3.22, dan 3.24, sehingga hal tersebut dapat direpresentasikan dengan probabilitas distribusi tipe musuh yang ditunjukan pada persamaan 3.23 dan 3.25.

$$DN' = \sum_{i=0}^{N_{DP}} \; EN \times \frac{DP_i}{100} \; \begin{cases} Saat \;\; \lceil DN_i \rceil - DN_i < DN_i - \lfloor DN_i \rfloor, & DN_i = \lceil DN_i \rceil \\ Saat \;\; \lceil DN_i \rceil - DN_i > DN_i - \lfloor DN_i \rfloor, & DN_i = \lfloor DN_i \rfloor \end{cases} \tag{3.20}$$

Pada persamaan 3.20 adalah persamaan untuk mencari *Distribute Number* atau distribusi musuh yang dinyatakan dengan DN' yang berupa variabel vektor menjadi $DN' = (DN_0, DN_1, DN_2, ..., DN_i)$. Dimana nilai dari variabel tersebut menyimpan jumlah musuh disetiap tipe yang diperoleh melalui jumlah musuh EN, yang sebelumnya sudah dibahas pada Sub-bab 3.3.1. Ke-

mudian EN dikalikan dengan presentase yang disiapkan sebelumnya melalui variabel masukan *Distribute Percentage* atau DP_i, dengan i adalah indeks dari variabel vektor DP yang totalnya dinyatakan dengan N_{DP}, yang secara berurutan memuat nilai pada variabel *Distribute Percentage* dalam Tabel 3.2. Keluaran dari DN sendiri berupa vektor yang berisi angaka bulat, maka dari itu setiap angka yang membentuk *cluster* jumlah musuh tersebut harus dilakukan operasi pembulatan seperti pada DN' dengan menggunakan syarat seperti pada persamaan 3.20. Bisa dikatakan bahwa DN' adalah vektor berisi jumlah musuh yang ditargetkan setiap tipenya. Pendistribusian jumlah musuh pada bagian ini mengikuti jumlah angka dalam variabel vektor DP, secara berurutan angka tersebut merealisasikan persentase jumlah tipe musuh.

Selanjutnya adalah operasi pembagian musuh dan tipenya setelah ditemukannya DN'. Setelah digunakan sebagai batasan dalam membagi setiap musuh berdasarkan presentase setiap tipenya, DP juga digunakan sebagai pembagi tipe musuh ke setiap musuh seperti pada persamaan 3.21, padahal DP adalah sebuah variabel vektor yang memuat presentase pembagian jumlah musuh berdasarkan tipe saja. Hal ini bersifat opsional dan dapat terus dikembangkan, mungkin dengan menambah variabel baru untuk mendistribusikan tipe musuh ke setiap musuh dengan cara tidak menggunakan variabel DP.

$$DL' = \sum_{i=0}^{N_{DP}} \left\lfloor \frac{DN_i}{DP_i} \right\rfloor \begin{cases} Saat\ i = 0, & DL_i = DN_i \\ Lainnya, & DL_{i-1} < DL_{(i-1)+DN_i} \end{cases} \tag{3.21}$$

Kemudian pada persamaan 3.21, dicarilah nilai DL' yang berupa variabel vektor tentang distribusi musuh DN_i yang masing-masing dibagi lagi dengan setiap presentase tipe DP_i seperti pada persamaan 3.20. Hal tersebut bertujuan membentuk sebuah vektor yang berisi distribusi musuh dan tipenya dengan skala yang lebih kecil lagi. Hal ini juga merujuk dari hasil proses yang dijelaskan pada Sub-bab 3.3.1, yang membentuk persebaran level dan dari setiap musuh yang ditunjukan pada Gambar 4.10. Dari persebaran musuh itulah kemudian dibagi ke dalam presentase tipe musuh, yang kemudian dibagi la-

gi ke dalam presentase yang lebih kecil dengan menggunakan jumlah variabel vektor DP yang dinyatakan dengan N_{DP}. Bila membagi setiap musuh tersebut ke dalam presetase tipe maka terdapat kemungkinan bahwa tidak semua musuh dapat terbagi secara merata atau habis, seperti halnya dengan proses pembulatan dalam pendistribusian tipe pada persamaan 3.20. Maka dicarilah juga sisa hasil bagi antara distribusi musuh DN_i dengan jumlah presentase tipe DP_i dengan persamaan 3.22.

$$rDL' \equiv \sum_{i=0}^{N_{DP}} DN_i \ (mod\ DP_i) \tag{3.22}$$

Pada persamaan 3.22, diperolehlah sisa musuh yang tidak terdistribusi dari persamaan 3.21 yang dinyatakan dengan variabel vektor rDL', saat dibaginya setiap jumlah musuh yang dinyatakan dengan DN_i dengan setiap presentase tipe yang dinyatakan dengan DP_i, dengan nilai i. Karena masing-masing dari variabel tersebut adalah variabel DN' dan DP yang berupa variabel vektor dengan dimensi yang sama. Musuh yang belum terdistribusi nantinya akan ditambahkan atau didistribusi ke tipe sesuai hasil dari persamaan 3.22, yang juga berupa variabel vektor yang menyimpan jumlah musuh yang belum terdistribusi pada tipe ke i yang dapat dinyatakan dengan rDL_i.

$$P(ET_j) = \sum_{i=0}^{M_{DL}} \sum_{j=0}^{N_{EN}} \frac{ET_j}{DL_i} \begin{cases} Saat\ ET_j \leq DL_i \\ Saat\ DL_{(j-1)} < ET_j \leq DL_i \end{cases} \tag{3.23}$$

Dari setiap musuh yang terdistribusi dan disimpan dalam variabel vektor DL', dimensi vektor tersebut kemudian dinyatakan dengan M_{DL} yang kemudian dilakukan pemetaan tipe, pada kasus ini digunakanlah konsep probabilitas seperti yang dijelaskan pada persamaan 2.3.2, bila disesuaikan dengan kasus ini maka persamaan yang diperoleh akan menjadi seperti pada persamaan 3.23. Pada setiap karakter musuh ET_j dengan N_{EN} adalah jumlah musuh secara keseluruhan akan meilih satu tipe yang akan digunakannya, tipe tersebut akan menentukan atribut *gameplay* yang akan dijelaskan pada Sub-bab 3.3.4. Kemudian probabilitas tipe yang dipilih ET_j dinyatakan dengan variabel $P(ET_i)$.

$$rET = \sum_{i=0}^{M_{EN}} EN_i \; - \; \sum_{i=0}^{N_{ET}} ET_i \tag{3.24}$$

Selanjutnya pada persamaan 3.24 adalah langkah untuk melihat apakah ada sisa musuh yang belum terdistribusi rET dengan cara melakukan pengurangaan antara jumlah setiap musuh EN_i yang sudah dijelaskan pada Sub-bab 3.3.1, yang mana M_{EN} pada variabel tersebut menandakan jumlah musuh sedangan ET_i adalah setiap musuh yang sudah memiliki tipe dengan N_{ET} sebagai jumlah musuh yang sudah memiliki tipe. Pada persamaan 3.24 variabel rET dapat bernilai 0 saat jumlah tipe musuh DN_i habis terbagi kedalam presentase dari tipe DP_i seperti pada persamaaan 3.21. Jika sisa musuh yang belum terdistribusi atau memiliki tipe, maka kemudian disimpan pada variabel rDL_i pada persamaan 3.22. Kemudian selanjutnya adalah mendistribusikan sisa musuh yang belum memilliki tipe tersebut, seperti yang dilakukan oleh persaman 3.25.

$$P(rET_j) = \sum_{i=0}^{M_{rDL}} \sum_{j=0}^{N_{rET}} \; \frac{ET_j}{rDL_i} \begin{cases} Saat\ ET_j \leq rDL_i, rET > 0 \\ Saat\ rDL_{(i-1)} < ET_j \leq rDL_i,\ rET > 0 \\ Lainnya\ 0 \end{cases} \tag{3.25}$$

Dilanjutkan pada persamaan 3.25, dengan dilakukannya distribusi sisa musuh yang belum memilikitipe pada rDL' yang berupa variabel vektor tempat penyimpan setiap sisa musuh, yang jumlah variabelnya dinyatakan dengan M_{rDL}. Kemudian setiap musuh dengan tipenya yang dinyatakan dengan ET_i, kemudian terdapat jumlah musuh belum memiliki tipe pada rET . Jumlah dari rET kemudian dinyatakan dengan N_{rET}, sebagai pembatas jumlah musuh yang belum memiliki tipe, hasil dari proses tersebut kemudian digabung dengan tipe musuh yang sudah terdistribusi, yang disimpan pada variabel DL'. Maka variabel $P(ET_i)$ adalah probabilitas tipe musuh yang dipilih dari distribusi level rDL_j pada persaamaan 3.23.

3.3.3 Distribusi Elemen dan Kelemahan Musuh

Pada persamaan 3.26 adalah penjelasan elemen yang digunakan oleh musuh seperti dideskripsikan pada Tabel 3.2 pada variabel *List Element*, di mana pada elemen teresebut terdapat kelemahan atau *weaknesses* dan kekebalan atau *repel* seperti yang dideskripsikan pada variabel *List Damage* pada Tabel 3.2. Seperti yang sudah dijelaskan pada Sub-bab 3.1.1 tentang Elemen dan Efektifitas Serangan.

Pada bagian ini elemen tersebut akan dibagi ke setiap musuh dengan kondisi yang berbeda, maksudnya nanti akan ada musuh yang memiliki kelemahan dan kekebalan yang berbeda antara musuh satu dengan musuh yang lain. Misalkan *Enemy 1* memiliki kelemahan api atau *fire* yang mana HP akan berkurang dua kali jika diserang dengan menggunakan elemen api dan memiliki kekebalan air atau *water* yang mana karakter tersebut kebal saat diserang dengan *skill* dari elemen air. Kemudian pada *Enemy 2* berbeda dengan *Enemy 1*, misalnya pada *Enemy 1* memiliki kelemahan api sedangkan pada *Enemy 2* memiliki kelemahan air misal dan kebal terhadap angin atau *wind*.

Kemudian daftar elemen yang akan digunakan mengacu pada Tabel 3.2 pada variabel *List Element* yang berisi *Phys* atau fisik, *Water* atau air, *Wind* atau angin, *Earth* atau tanah dan *Fire* atau api. Kemudian urutan efek serangan juga dinyatakan pada Tabel 3.2 dengan variabel List Damage yang berisi *Normal*, *Repel* dan *Weak*. Dimana penjelasan untuk setiap elemen dan efek serangan sudah dijelaskan sebelumnya pada Sub-bab 3.1.1.

$$ElN = \begin{cases} Phys \\ Water \\ Wind \\ Earth \\ Fire \\ ... \\ n \end{cases} \tag{3.26}$$

Pada dasarnya satu karakter musuh tidak akan menggunakan seluruh elemen yang ada pada persamaan 3.26, yang merupakan penggambaran dari varaibel *List Element* pada Tabel 3.2, mungkin hanya sekitar satu sampai dengan tiga elemen. Maka dari itu kondisi tersebut dapat dinyatakan dengan kondisi $DmgNa \in ElN$, yang mana seluruh elemen bisa menjadi kelemahan atau kekebalan dari musuh. Sama seperti pada persamaan 3.26 yang mana ElN adalah nama elemen, sedangkan $DmgNa$ adalah *damage name* atau nama dari efek serangan yang dilakukan. Di mana pada persamaan 3.27 $DmgNa$ akan dipilih secara acak dengan persamaan 3.28, yang akan menjadi respon terhadap serangan dari pemain seperti pada persamaan 3.27. Pada persamaan 3.27 adalah penggambaran dari variabel *List Damage* pada Tabel 3.2.

$$DmgNu' = \begin{cases} 0, & Normal \\ 1, & Repel \\ 2, & Weak \\ ... & \\ n, & Defined\ Status\ Name \end{cases} \tag{3.27}$$

Kemudian pada persamaan 3.27, variabel $DmgNu'$ adalah variabel vektor yang menyimpan respon atau efek dari serangan yang yang dikonversi menjadi angka, yang pada mulanya berupa nama respon atau efek setelah diserang. Kemudian hasil pemilihan tersebut disimpan pada variabel $DmgNu_0$ yang digunakan dalam persamaan 3.28, maksud dari variabel $DmgNu_0$ ini adalah mewakili satu respon atau efek serangan yang terpilih.

$$P(DmgNu_0) = \frac{DmgNu_0}{\sum_{i=0}^{N_{Na}} DmgNa_i} \tag{3.28}$$

Pada persamaan 3.27, terdapat variabel $DmgNa$ yang terdiri dari banyak nama elemen, kemudian dipilihlah secara acak. Nantinya digunakan sebagai elemen yang memiliki respon khusus terhadap serangan yang dinyatakan dengan $DmgNu_0$, selanjutnya angka yang deperoleh berupa indeks yang dipilih dari jumlah element yang dinyatakan dengan N_{Na} seperti yang pada persamaan 3.28. Jadi dari $DmgNa$ akan dipilih satu elemen secara acak yang

kemudian menjadi kelemahan atau kekebalan dari karakter musuh tersebut. Maka dilanjutkan dengan persamaan 3.29, agar dalam satu karter musuh bisa memiliki lebih dari satu efek serangan.

$$P(DmgNu_i) = \sum_{i=0}^{M_{Nu}} \frac{DmgNu_i}{\sum_{j=0}^{N_{Na}} DmgNa_j} \tag{3.29}$$

Sedangkan pada persamaan 3.29 adalah penjelasan tentang proses munculnya setiap respon atau efek serangan pada satu karakter musuh yang dinyatakan dengan $DmgNu'$ yang berupa vektor, sehingga variabel tersebut berubah menjadi $DmgNu_i$ dengan M_{Nu} adalah jumlah dari respon atau efek saat musuh menerima serangan. Seperti penjelasan sebelumnya bahwa pada sebuah karakter musuh dapat memiliki lebih dari satu kelemahan atau kekebalan $DmgNu'$ dari $DmgNa$ yang dipilih secara acak seperti yang sudah dijelaskan pada bagian sebelumnya. Berkaca pada persamaan 3.28 dan 3.29, penggunaan Naive bayes dengan dasar probabilitas diterapkan pada kedua persamaan tersebut yang mengacu dengan penjelasa pada Sub-bab 2.3.3.

$$DmgEN' = \sum_{i=0}^{EN} \sum_{j=0}^{N_{Nu}} DmgNu'_{ij} \tag{3.30}$$

Selanjutnya adalah proses penerapan persamaan 3.29 ke seluruh karakter musuh yang dinyatakan dengan $DmgEN'$ yang berupa vektor seperti $DmgEN'$ $= (DmgNu'_0, DmgNu'_1, ..., DmgNu'_n)$. Seperti pada persamaan 3.30, $DmgEN'$ berupa variabel vektor yang berisi persebaran $DmgNu'$ dan elemennya $DmgNa$ pada setiap karakter musuh. Sesuai dengan persamaan 3.30, yang mana EN adalah jumlah musuh sedangkan N_{Nu} adalah jumlah dari $DmgNu'$ pada satu karakter musuh yang berjumlah satu sampai dengan tiga, masing-masing dinyatakan dengan i dan j sehingga terbentuklah $DmgNu'_{ij}$

3.3.4 Distribusi HP, MP, dan Atribut Gameplay Musuh

Bila melihat pada Tabel 3.2 terdapat beberapa variabel seperti *List Attribute Name*, *Max Attribute Value*, dan *Min Attribute Value*. Variabel terse-

but akan nantinya akan digunakan untuk membuat atribut *gameplay* untuk karakter musuh dengan basis algorima Naive Bayes. Di awali dengan persamaan 3.31 yang menjadi pembagi setiap operasi pembagian atribut *gameplay* berdasarkan tipe.

$$ET' = \sum_{i=0}^{N} ET_i \begin{cases} ET_0 = 0, & MX \\ ET_1 = 1, & HM \\ ET_2 = 2, & SM \\ ET_3 = 3, & HS \\ ET_4 = 4, & SS \end{cases} \tag{3.31}$$

Pada persamaan 3.31 yang merepresentasikan masukan variabel *Enemy Type* pada Tabel 3.2, sedangkan pada persaamaan 3.32 adalah pengembangan jika jumlah tipe musuh ditambahkan melebihi yang dicontohkan pada Tabel 3.2 dalam variabel *Enemy Type*. Pada kedua persamaan tersebut terdaapat beberapa variabel ET yang menyatakan tipe musuh ke i yang berupa urutannya dan N adalah jumlah dari tipe musuh, selanjutnya tipe yang dapat dipilih secara berurutan masing-masing dicontohkan dengan MX, HM, SM, HS, dan SS adalah musuh dengan tipe *mixed*, *hard magic*, *soft magic*, *hard strength*, dan *soft strength*.

$$ET' = \sum_{i=0}^{N} ET_i \begin{cases} ET_0 = 0, & MX \\ ET_1 = 1, & HM \\ ET_2 = 2, & SM \\ ET_3 = 3, & HS \\ ET_4 = 4, & SS \\ \quad ... & \quad ... \\ ET_N = N, & ST_N \end{cases} \tag{3.32}$$

Kemudian pada persamaan 3.32 secara spesifik terdapat variabel ET_N yang merupakan variabel tipe musuh ke N atau batas jumlah tipe musuh. Variabel i adalah urutan dari indeks setiap tipe musuh yang dideskripsikan, berikut juga variabel ST_N untuk penamaan nama tipe musuh. Maka dapat disimpulkan bahawa ET' adalah variabel berbentuk vektor yang ber-

isi variabel jumlah setiap tipe musuh didalamnya, yang dinyatakan dengan $ET' = (ET_0, ET_1, ..., ET_i)$. Selanjutnya dilakukan pembagian tipe musuh jika musuh tersebut memiliki tipe MX' atau *mixed* seperti pada persamaan 3.33. Bila mengacu pada persamaan 3.32, maka MX' adalah sama dengan ET_0.

$$MX' = \sum_{i=0}^{N_{MX}} MX_i \begin{cases} MX_0 = 0, & MP\ Focused \\ MX_1 = 1, & HP\ Focused \end{cases} \tag{3.33}$$

Pada persamaan 3.33 terdapat variabel MX_i yang berupa variabel vektor yang berisi urutan atribut *gameplay* dari musuh yang bertipe *mixed*. Kemudian pada variabel MX' yang juga berupa variabel vektor, dinyatakan dengan $MX' = (MX_0, MX_1, ..., MX_i)$ yang nantinya akan memuat jumlah musuh dengan tipe *mixed* sesuai dengan golongan. Pada persamaan 3.33 sendiri dijelaskan bahwa pada tipe *mixed* digolongkan menjadi dua tipe, yang satu berfokus pada HP dan satu lagi pada MP. Jadi isi dari variabel MX' nantinya akan terdiri dari musuh bertipe *mixed* atau MX_i yang memuat atribut *gameplay* dengan fokus pada HP dan MP.

$$MX' = \sum_{i=0}^{N_{MX}} MX_i \begin{cases} MX_0 = 0, & MP\ Focused \\ MX_1 = 1, & HP\ Focused \\ \quad ... & \quad ... \\ MX_{N_{MX}} = N_{MX}, & Lainnya \end{cases} \tag{3.34}$$

Pada persamaan 3.34 adalah lajutan penjelasan dari persamaan 3.33 yang lebih detail tentang persebaran musuh dengan tipe *mixed* yang dinyatakan dengan MX'. Jika ingin melakukan penambahan fokusan atau golongan untuk MX, maka dari persamaan tersebut ditambahkan variabel MX yang ke N_{MX} dengan N_{MX} yang bisa juga digunakan sebagai penomoran untuk fokusan tersebut.

Jika sebelumnya adalah pembahasan tentang variabel untuk jumlah musuh berdasarkan tipenya, yang nantinya akan menampung jumlah dari setiap karakter musuh. Selanjutnya adalah perhitungan probabilitas dipilihnya ti-

pe musuh dari keseluruhan tipe musuh yang berada pada variabel masukan dengan nama *Enemy Type* dalam Tabel 3.2. Hal ini tentu saja berhubungan dengan Sub-bab 3.3.2 yang membahas tentang pendistribusian jumlah tipe musuh, jadi terdapat batasan jumlah musuh setiap tipenya. Dalam menyelesaikan kasus ini, digunakanlah Naive bayes yang berbasis pada *conditional probability* seperti yangg dijelaskan pada Sub-bab 2.3.3.

$$\begin{aligned} ET_0 &= P(ET|ET_0) \times P(ET_0) \\ ET_1 &= P(ET|ET_1) \times P(ET_1) \\ ET_2 &= P(ET|ET_2) \times P(ET_2) \\ ET_3 &= P(ET|ET_3) \times P(ET_3) \\ ET_4 &= P(ET|ET_4) \times P(ET_4) \end{aligned} \tag{3.35}$$

Pada persamaan 3.35 adalah proses kemunculan tipe musuh tertentu misal ET_0, ET_1, ET_2, ET_3, dan ET_4, secara berurutan tipe musuh tersebut adalah MX, HM, SM, HS, dan SS, yang sebelumnya sudah dideskripsikan pada persamaan 3.31.

Sedangkan jika jumlah tipe musuh tidak beracuan pada variabel *Enemy Type* dalma Tabel 3.2 atau tipe musuh berjumlah lebih dari empat maka persamaan 3.35 akan berubah menjadi persamaan 3.36. Semula berawal dari variabel ET_0 sampai ET_4 pada persamaan 3.35, kemudian berubah dari variabel ET_0 sampai ET_N dengan N adalah batas akhir jumlah tipe musuh seperti pada persamaan 3.36.

$$\begin{aligned} ET_0 &= P(ET|ET_0) \times P(ET_0) \\ ET_1 &= P(ET|ET_1) \times P(ET_1) \\ ET_2 &= P(ET|ET_2) \times P(ET_2) \\ &... \\ ET_N &= P(ET|ET_N) \times P(ET_N) \end{aligned} \tag{3.36}$$

Saat sudah diperolehnya tipe untuk setiap musuh pada seperti yang dijelaskan pada persamaan 3.35 dan 3.36 maka selanjutnya yang harus dibuat

adalah atribut *gameplay* dari musuh itu sendiri, seperti halnya *Strength*, *Magic*, *Endurance*, dan lain sebagainya. Seperti yang tercantum pada Tabel 3.2 dalam variabel *List Attributes Name* ditambah variabel HP dan MP.

$$bHP = minHP \tag{3.37}$$

Pada persamaan 3.37 adalah penentuan batas bawah untuk HP atau bHP, nilai pada variabel $minHP$ yang juga dicontohkan dalam daftar variabel masukan dengan nama *Min HP* pada Tabel 3.2, yang merupakan nilai minimal dari HP.

$$tHP = \sum_{i=0}^{NE} bHP + \left| \frac{Elv_i}{100} \times maxHP \right| \tag{3.38}$$

Selanjutnya dilanjutkan dengan pencarian batas atas untuk HP atau tHP. Pada persamaan 3.38 terdapat beberapa variabel yang berpengaruh terhadap nilai dari tHP diantaranya adalah bHP yang merupakan batas bawah dari HP, Elv_i yang merupakan level dari setiap karakter musuh sebanyak NE dengan tipenya masing-masing, dan $maxHP$ sendiri yang merupakan nilai maksimum dari HP dicontohkan pada Tabel 3.2 dengan nama variabel *Max HP*. Pencarian batas atas dilakukan dengan dilakukannya penjumlahan antara bHP dengan presentase dari $maxHP$ yang disesuaikan dengan Elv_i. Jadi dari perhitungan tersebut, HP dari karakter musuh akan menyesuaikan dengan levelnya masing-masing.

$$P(HP_i) = \sum_{i=0}^{NE} \frac{HP_i}{bHP_i - tHP_i} \tag{3.39}$$

Kemdian dilanjutkan dengan perhitungan probabilitas $P(HP_i)$ atau munculnya HP pada setiap karakter musuh pada persamaan 3.39. HP dari setiap karakter musuh itu sendiri dinyatakan dengen HP_i. Kemudian range nilai dari HP_i yang akan muncul terhitung mulai dari bHP_i sampai dengan tHP_i dari setiap karakter musuh. Variabel i pada persamaan tersebut merepresentasikan setiap musuh dengan variabel NE yang menyatakan batas atau jumlah dari

musuh yang ingin dibuat. Pada proses ini menggambarkan penggunaan Naive Bayes dengan dasar probabilitas yang digunakan.

$$bMP = minMP \tag{3.40}$$

Mengulang persamaan 3.37 pada persamaan 3.40, hanya mengganti variabelnya saja. Jika pada persamaan 3.37 kasus yang ingin diselesaikan adalah pencariaan HP, maka pada persamaan 3.40 kasus yang ingin diselesaikan adalah pencarian MP. Pada persamaan 3.37 dengan variabel bHP diganti dengan bMP dan variabel $minHP$ diganti dengan $minMP$ yang mana pada Tabel 3.2 dicontohkan dengan variabel Min MP.

$$tMP = \sum_{i=0}^{NE} bMP + \left| \frac{Elv_i}{100} \times maxMP \right| \tag{3.41}$$

Masih sama seperti penjelasan sebelumnya dimana pada persaamaan 3.40 pada persamaan 3.41 juga sama dengan persamaan 3.38. Di lakukannya penggantian variabel HP menjadi MP, sudah dijelaskan juga pada bagian sebelumnya tentang variabel Elv_i, dan pada variabel $maxMP$ yang merupakan variabel yang dicontohkan pada Tabel 3.2 dengan nama Max MP.

$$P(MP_i) = \sum_{i=0}^{NE} \frac{MP_i}{bMP_i - tMP_i} \tag{3.42}$$

Pada persamaan 3.42 adalah probabilitas $P(MP_i)$ atau munculnya MP pada setiap karakter musuh pada persamaan 3.42. MP dari setiap karakter musuh itu sendiri dinyatakan dengen MP_i. Kemudian range nilai dari MP_i yang akan muncul terhitung mulai dari bMP_i sampai dengan tMP_i. Kasus ini sama seperti pada persamaan 3.39 hanya saja sama seperti pada pembahasan sebelumnya yang mana pada persamaan tersebut bertujuan untuk mencari HP, sedangkan pada kasus ini bertujuan untuk mencaru MP.

Pada bagian ini akan membahas secara khusus untuk kondisi *mixed* melanjutkan seperti yang ada pada persamaan 3.33. Dalam kondisi *mixed* sendiri

terdapat dua kondisi yaitu HP *focused* atau yang berfokus kepada HP, kemudian MP *focused* atau yang befokus pada MP. Bila masing-masing tipe tersebut dinyatakan dengan persamaan secara berurutan untuk HP *focused* dan MP *focused* dengan persamaan 3.43 dan persamaan 3.44.

$$MX_1 = P(MX|MX_1) \times P(MX_1) \tag{3.43}$$

$$MX_2 = P(MX|MX_2) \times P(MX_2) \tag{3.44}$$

Maka Mx adalah sebuah set variabel yang berisi seluruh musuh yang bertipe *mixed*, kemudian dipilihlah musuh dengan tipe *mixed* yang berfokus pada HP atau MX_1. Beracuan pada konsep *conditional probability* yang dijelaskan pada Sub-bab 2.3.2. Jika MX_1 adalah HP *focused* maka MX_2 adalah MP *focused* maka sama seperti persamaan 3.43 yang sebelumnya juga dicontohkan pada persamaan 3.33. Dengan MX adalah seluruh musuh yang bertipe *mixed*, kemudian dipilihlah musuh dengan tipe *mixed* yang berfokus pada MP atau MX_2. Pembahasan secara khusus lainnya adalah penambahan HP untuk musuh dengan tipe *mixed* yang berfokus pada HP seperti yang dinyatakan pada persamaan 3.45 dengan NH_{hp} adalah jumlah musuh yang bertipe *mixed* dan fokus ke HP.

$$tHP = \sum_{i=0}^{NM_{hp}} bHP + bHP + \left| \frac{Elv_i}{100} \times maxHP \right| \tag{3.45}$$

Sedangkan pada musuh dengan tipe *mixed* yang berfokus pada MP tidak perlu diberi perlakuan khusus seperti tipe *mixed* yang berfokus pada HP, hal tersebut dikarenakan dengan melihat nilai MP pada setiap karakter musuh dengan tipe tersebut sudah tergolong tinggi. Kondisi ini beracuan terhadap konsep keseimbangan dalam mendesain seperti yang dijelaskan pada Sub-bab 2.2.3 dan Sub-bab 3.1.2 tentang penyesuaian keseimbangan dalam pertarungan antara pemain dengan musuh. Sedikit berbeda dengan persamaan 3.39, pada persaamaan 3.45 dilakukan penambahan dua kali pada variabel bHP atau batas bawah HP. Hal tersebut dilakukan agar nilai HP pada musuh dengan

tipe *mixed* yang berfokus terhadap HP memiliki nilai HP yang lebih tinggi jika dibandingkan dengan tipe *mixed* yang berfokus MP.

$$P(HP_i) = \sum_{i=0}^{NM_{hp}} \frac{HP_i}{bHP_i - tHP_i} \tag{3.46}$$

Kemudian perhitungan probabilitas $P(HP_i)$ atau munculnya HP pada setiap karakter musuh pada persamaan 3.46. HP dari setiap karakter musuh itu sendiri dinyatakan dengen HP_i. Kemudian range nilai dari HP_i yang akan muncul terhitung mulai dari bHP_i sampai dengan tHP_i. Persamaan tersebut masih sama dengan probabilitas kemunculan HP dari *range* antara bHP dengan tHP pada persamaan 3.39.

Selain pembuatan HP, MP, dan atribut *gameplay* lain seperti halnya *Strength*, *Magic*, *Endurance*, *Speed* dan *Luck* juga harus dicari. Daftar dari atribut *gameplay* tersebut dicontohkan pada Tabel 3.2 dengan nama variabel *List Attributes Name*. Dalam kasus pembuatan atribut *gameplay* musuh terdapat beberapa variabel lain digunakan selain *List Attributes Name* diantaranya adalah *Max Attributes Value* dan *Min Attributes Value*, seperti yang ditunjukan pada Tabel 3.2. Dalam pembuatan atribut *gameplay* musuh tersebut digunakanlah persamaan yang mirip dengan persamaan 3.37, 3.38, dan 3.39, hanya saja dilakukan sedikit perubahan pada persamaan tersebut seperti yang dinyatakan pada persamaan 3.47 sampai dengan persamaan 3.50.

$$bSt = minSt \tag{3.47}$$

Jika hanya berlaku satu atribut *gameplay* saja maka dapat digunakan persamaan 3.47 yang pada dasarnya sama seperti persamaan-persamaan sebelumnya dalam mencari batas bawah dari HP pada bHP dan MP pada bMP. Sedangkan pada persamaan 3.47 batas bawahnya adalah bSt yang diisi dengan nilai minimum dari atribut *gameplay* atau $minSt$.

$$bSt = \sum_{i=0}^{NE} \sum_{j=0}^{N_{st}} minSt_{ij} \tag{3.48}$$

Di karenakan atribut *gameplay* dari musuh berjumlah lebih dari satu maka persamaan 3.47 harus disesuaikan dengan jumlah atribut *gameplay* tersebut. Jumlah atribut *gameplay* dari musuh sendiri berjumlah lima seperti yang tertulis pada variabel *List Attributes Name*, maka persamaan 3.47 disesuaikan menjadi seperti persamaan 3.48. Dengan variabel N adalah batas maksimal dari jumlah atribut *gameplay*, dengan nilai minimum atribut *gameplay* atau $minSt$ yang jumlahnya mengikuti jumlah maksimal atribut *gameplay* menjadi $minSt_i$ dan disimpan dalam setiap variabel bSt ke i. Kemudian selanjutnya adalah pencarian batas atas untuk masing-masing atribut *gameplay*, sama halnya dengan pencarian batas bawah dari setiap atribut *gameplay* dalam cara pencarian batas atas pada dasarnya juga sama dengan pencarian batas dari HP pada tHP dan MP pada tMP.

$$tSt = \sum_{i=0}^{NE} \sum_{i=0}^{N_{st}} bSt_{ij} + \left| \frac{Elv_i}{100} \times maxSt_{ij} \right| \tag{3.49}$$

Pada persamaan 3.49 adalah persamaan untuk mencari batas atas yang dinyatakan dengan tSt yang diisi dengan nilai maximum dari atribut *gameplay* atau $maxSt$. Namun pada kasus ini jumlah atribut *gameplay* musuh berjumlah lebih dari satu seperti yang sudah dibahas pada bagian sebelumnya. Sama seperti pada pembahasan sebelumnya bahwa variabel N_{st} adalah jumlah atribut *gameplay* musuh, maka dari itu nilai maksimum $maxSt$ dan variabel penyimpan hasil batas maksimum tSt yang berupa set dieksekusi satu-satu sesuai jumlah atribut *gameplay*. Maka nilai maksimum pada setiap atribut *gameplay* menjadi $maxSt_j$. Dalam pencarian nilai batas atas atau tSt tentunya harus melibatkan batas bawah. Jika melihat pada persamaan 3.49 maka terdapatlah variabel bSt_j. Karena musuh berjumlah tidak hanya satu maka variabel NE digunakan sebagai batasan maksimal jumlah atribut *gameplay* pada setiap karakter musuh. Dengan varaiabel i yang merepresentasikan urutan musuh yang sedang diproeses. Maka dari itu bSt_i mejadi bSt_{ij} dan $maxSt_j$ berubah menjadi $maxSt_{ij}$. Pada persamaan 3.49 juga sama seperti pencarian batas atas pada HP dan MP hanya saja jumlah atribut *gameplay* yang dibuat

berjumlah lebih banyak. Kemudian level juga menjadi acuan presentase nilai atribut *gameplay* yang dinyatakan dengan Elv_i untuk setiap musuh.

$$P(St_{ij}) = \sum_{i=0}^{NE} \sum_{j=0}^{N_{st}} \frac{St_{ij}}{bSt_{ij} - tSt_{ij}} \tag{3.50}$$

Kemudian pada persamaan 3.50 adalah probabilitas saat munculnya atribut *gameplay* St_{ij} pada setiap musuh yang dinyatakan dengan $P(St_{ij})$ dari batas bawah atribut *gameplay* bSt_i sampai batas atas atribut *gameplay* tSt_i yang juga untuk setiap musuh. Hal ini merujuk pada penggunaan Naive Bayes seperti yang ditunjukan pada persamaan 2.3.3, tentang penggunaan probabilitas dalam distribusi atribut *gameplay*. Untuk hasil keluaran pada proses ini akan dibahas dan ditunjukan pada Sub-bab 4.3.4, begitu juga dengan hasil dari Sub-bab sebelumnya seperti distribusi level musuh pada Sub-bab 3.3.1, distribusi tipe musuh pada Sub-bab 3.3.2 dan distribusi elemen dan kelemahan musuh pada Sub-bab 3.3.3 yang secara berurutan ditunjukan dan dibahas pada Sub-bab 4.3.1, 4.3.2, dan Sub-bab 4.3.3.

3.4 Klasifikasi Hero permainan Dota 2 dengan Neural Network Multiclass Classification

Pada permaian Dota 2, *hero* (sebutan khusus untuk karakter pada Dota 2) terbagi menjadi 3 kelompok yaitu *Strength* (STR), *Agility* (AGI) dan *Intelligent* (INT). Tiap *hero* memiliki atribut *gameplay* yang berbeda-beda seperti *Base* STR, *Base* AGI, *Base* INT, STR *Growth*, Min DMG atau *minimum damage*, Max DMG atau *maximum damage*, *Movement Speed*, dan lain-lain. Data tersebut yang nantinya akan digunakan pada penelitian ini, seluruhnya terangkum dalam situs *Table of Hero Attribute* (Fandom, 2020).

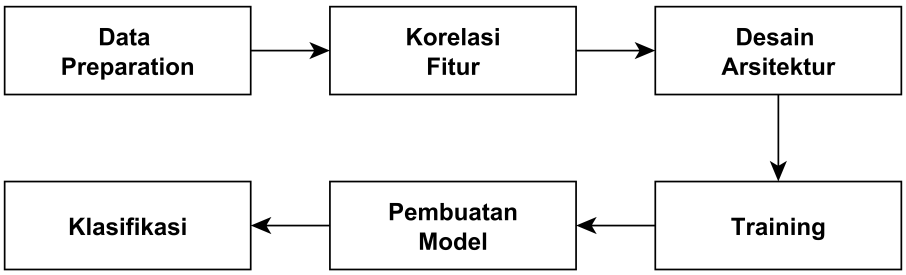
Dalam kasus ini diambil contoh Centaur, hero tersebut adalah bertipe STR, jika dilihat nilai *Base* STR dan STR *Growth* pada *hero* ini sangat tinggi, namun ada cukup banyak hero lain yang mempunyai nilai atribut *gameplay* tinggi pada salah satu atribut tapi bukan merupakan *hero* dengan tipe tersebut.

HERO	A	STR	STR+	STR 25	AGI	AGI+	AGI 25	INT	INT+	INT 25
Centaur Warrunner		23	4.3	126.2	15	1.6	53.4	15	1.6	53.4
Treant Protector		25	3.6	111.4	15	2	63	17	1.8	60.2
Doom		26	3.5	110	11	0.9	32.6	13	2.1	63.4
Pudge		25	3.5	109	14	1.5	50	14	1.5	50
Ogre Magi		23	3.5	107	14	1.55	51.2	17	2	65
Tiny		26	3.3	105.2	0	0	0	17	1.6	55.4
Lycan		25	3.3	104.2	16	1	40	17	1.55	54.2
Kunkka		24	3.3	103.2	14	1.3	45.2	18	1.5	54

Gambar 3.13: Atribut *gameplay* pada *hero* Centaur.

Seperti Ogre Magi, *hero* tersebut adalah type INT, tapi memililki nilai *Base* STR, STR *Growth* dan Max STR lebih tinggi dari pada atribut *gameplay* *intelligent* seperti yang terlihat pada Gambar 3.13.

Maka hal yang akan dilakukan disini adalah mencoba melakukan klasifikasi *hero* berdasarkan atribut *gameplay* yang ada, dataset dari *hero* Dota 2 dapat dilihat pada Tabel 5.31 dan 5.32 untuk data *training* dan Tabel 5.33 untuk data *testing*. Selanjutnya adalah tahapan klasifikasinya seperti yang direpresentasikan pada Gambar 3.14.



Gambar 3.14: Proses klasifikasi *hero* pada permainan Dota 2.

3.4.1 Data Pre-processing

Di lakukan *parsing* halaman tersebut (Fandom, 2020) dan dilakukan konversi dari tabel HTML menjadi CSV. Maka diperolehlah 115 buah data. Jika melihat Gambar 3.15 terdapat beberapa data yang sifatnya spesifik untuk interaksi pada *hero* tersebut didalam permainan. Hal tersebut ikut tercantum

dalam dataset, maka akan dibuang, karena dianggap kurang relevan seperti *Day Vision*, *Night Vision*, *Collision Size*, dan *Legs*. Sehingga diperoleh 22 fitur yang akan digunakan untuk melakukan klasifikasi. Tipe *hero* akan digunakan sebagai target dikategorikan dalam bentuk angka menjadi 0 untuk STR, 1 untuk AGI dan 2 untuk INT.

Gambar 3.15: *Gameplay* pada permainan Dota 2.

Dari seluruh data *hero* tersebut, enam belas *hero* diambil sebagai data *testing* yang tercantum pada Tabel 5.33 dan sisanya digunakan sebagai data *training* pada Tabel 5.31 dan Tabel 5.32 yang semua terlampir dalam LAMPIRAN. Untuk data *testing* pada Tabel 5.33, diambilah beberapa *hero* yang memiliki atribut *gameplay* aneh seperti Jakiro, Winter Wyvern (STR tinggi tetapi masuk ke dalam kategori INT *hero*), Phoenix, IO (STR rendah tetapi termasuk ke dalam kategori STR *hero*), Bloodseeker, Undying, dan lain-lain.

3.4.2 Korelasi Fitur

Dari 22 fitur yang dibahas pada Sub-bab 3.4.1, bisa jadi tidak semua fitur mempunyai kontribusi terhadap tipe *hero* atau bisa dibilang proses naik-turun sebuah fitur tersebut tidak memberikan pengaruh terhadap proses klasifikasi tipe *hero*. Dalam pencarian fitur tersebut digunakanlah *korelasi matrix* pada setiap variabel, hasil perhitungan korelasi matrix dari atribut *gameplay* permainan Dota 2 yang dijadikan fitur dapat dilihat pada Tabel 3.3.

Tabel 3.3: Hasil perhitungan korelasi matrix untuk fitur

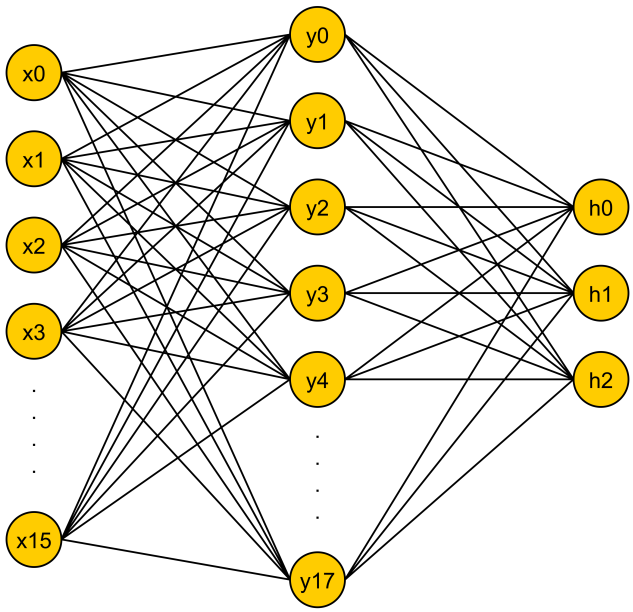
Fitur	Korelasi Matrix
type	1.000000
baseStr	-0.542666
strGrowth	-0.542322
maxStr	-0.581514
baseAgi	-0.038086
agiGrowth	-0.123072
maxAgi	-0.110488
baseInt	0.688410
intGrowth	0.657814
maxInt	0.699666
totalBaseAttr	0.179483
totalAttrGrowth	0.063110
totalMaxAttr	0.114713
moveSpeed	0.067136
baseArmor	-0.113657
minDmg	-0.384719
maxDmg	-0.312799
range	0.687704
baseAttackTime	-0.297982
attackPoint	-0.106053
attackBackswing	0.011558
turnRate	-0.034759
regeneration	-0.090677

Berikut penjelasan sederhana mengenai *korelasi matrix*, hal tersebut membahas tentang sebuah variabel yang memuat nilai antara -1 dan $+1$. Sebagai contoh jika ada korelasi antara variabel X dan Y yang benilai -1, maka pa-

da saat nilai X turun, nilai Y akan naik. Jika bernilai $+1$, maka hubungan antara X dan Y adalah linear. Namun jika nilai korelasi semakin dekat dengan 0, maka variabel tersebut bisa dikatakan tidak mempengaruhi satu dengan yang lain. Selanjutnya akan dibuang semua fitur yang memiliki nilai korelasi lebih kecil dari 0.1 atau -0.1 seperti *regeneration rate*, *turn rate*, *movement speed* dan yang paling aneh adalah fitur *base* AGI, AGI *growth* dan *max* AGI memiliki nilai yang sangat kecil. Padahal fitur tersebut adalah termasuk kemampuan dasar dari *hero*, sehingga didapatkan 16 fitur yang akan digunakan sebagai masukan untuk *Neural Network*.

3.4.3 Desain Arsitektur Neural Network

Arsitektur *Neural Network* yang akan dibuat mempunyai enam belas *neuron* pada layer masukan, delapan belas *neuron* pada *hidden layer* dengan fungsi aktivasi *sigmoid* dan tiga *neuron* yang mewakili (STR, AGI dan INT) pada layer keluaran seperti pada Gambar 3.16 dengan fungsi aktivasi *Softmax* karena keluaran dari model yang dibuat bersifat distribusi probabilitas dari seluruh nilai target.



Gambar 3.16: Arsitektur *Neural Network* untuk klasifikasi *hero* Dota 2.

Misalkan model 100% yakin jika *hero* A adalah STR, maka keluaran dari model adalah [1, 0, 0] dan jika model tersenut 50% yakin jika *hero* B adalah AGI, 25% yakin jika hero B adalah STR atau INT, maka keluaran dari model ini adalah [0.25, 0.5, 0.25]. Harus diingat bahwa total nilai dari distribusi probabilitas untuk semua target adalah 1.

Sedangkan untuk *loss function* yang digunakan adalah *Cross Entropy* dan *Optimizer* yang digunakan adalah SGD (*Stochasstic Gradient Descent*) dengan *learning rate* sebesar 0.001. Sebagai catatan penambahkan *metrics accuracy* bertujuan untuk melihat seberapa bagus model ini dalam melakukan klasifikasi.

3.4.4 Training

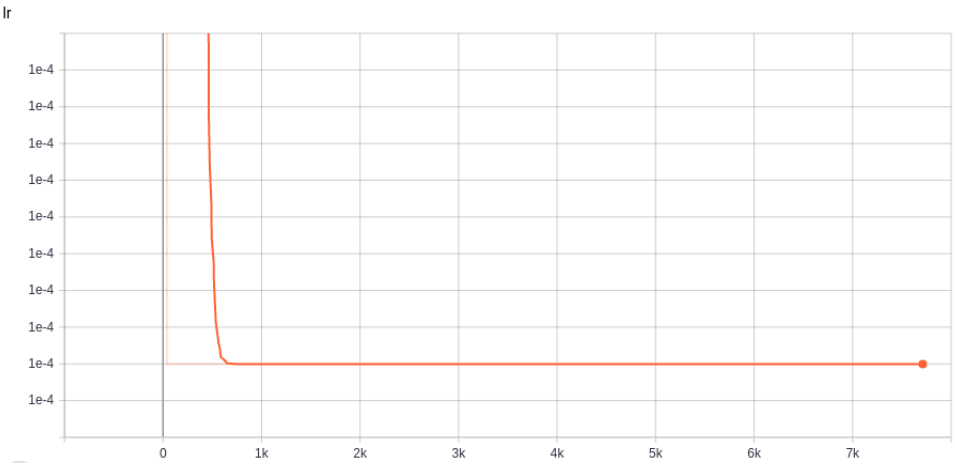
Data hasil dari korelasi fitur pada Sub-bab 3.4.2, karena tipe yang ditunjukan pada kolom ke satu dengan kepala yang bertuliskan "Fitur" adalah pedoman hasil prediksi maka tidak akan diikutkan proses *training*. Pada kolom *type* tipe karakter dinyatakan dengan 0, 1, dan 2 yang menyatakan STR, AGI, dan INT. Nantinya keluaran dari hasil *training* yang menunjukan prediksi tipe dari karakter dinyatakan dalam bentuk *one-hot vector*: STR $\rightarrow [1, 0, 0]$, AGI $\rightarrow [0, 1, 0]$ dan INT $\rightarrow [0, 0, 1]$.

Pada proses *training* hasil prediksi akan dibandingkan dengan data *testing* untuk mendapatkan nilai *loss/error* dan validasi untuk melihat performa model *Neural Network* yang dibuat. Nilai *loss* kemudian digunakan untuk meningkatkan akurasi model. Proses *training* dilakukan untuk mendapatkan model dengan akurasi optimal dan tidak *overfitting*.

Penentuan nilai *batch size* dan *learning rate* berperan penting dalam peningkatan akurasi hal ini biasa disebut dengan *hyperparameter tuning*. Dengan *Batch size* yang besar akan dapat memberikan jenis data yang bervariasi sehingga saat dilakukan perhitungan *loss*, nilai perhitungan *loss* yang diperoleh dapat mewakili variasi dataset secara keseluruhan. Sebaliknya *batch size* yang kecil hanya dapat memberikan perwakilan sedikit dataset sehingga nilai per-

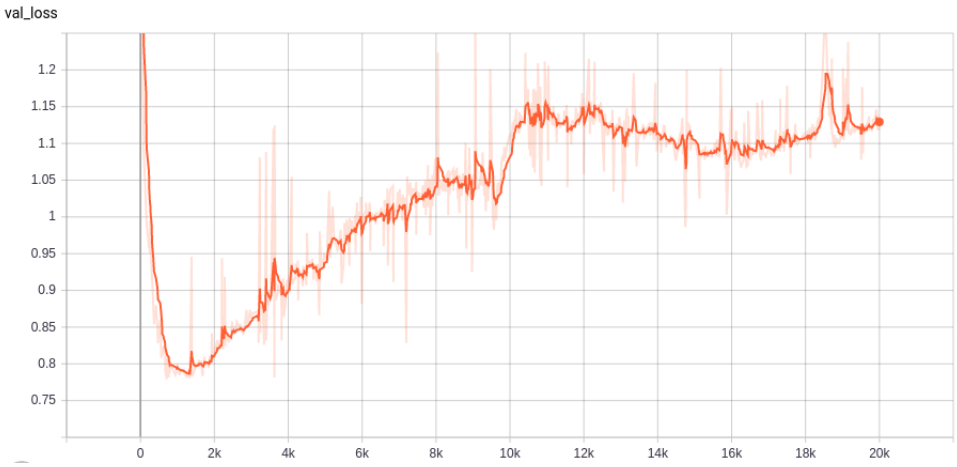
ubahan *loss* untuk memperoleh nilai yang stabil akan semakin sulit. Karena itu *batch size* yang besar dapat diberikan *learning rate* yang besar karena arah perpindahan *loss* yang lebih stabil. Sedangkan *batch size* yang kecil harus diberikan *learning rate* yang kecil juga agar perpindahannya tidak berpindah pindah terlalu jauh.

Proses *training* akan dilakukan sebanyak 2×10^4 *epoch*/iterasi. Untuk satu *epoch* dataset akan dibagi menjadi 16 *batch size* tiap *batch* hingga semua data terproses. Nilai *learning rate* sebesar 1×10^{-3} dan jika nilai *loss* tidak mengalami perubahan maka *learning rate* akan diperkecil sampai dengan 1×10^{-4} seperti pada Gambar 3.17. Namun dalam kasus ini digunakannya optimasi untuk menurunkan *learning rate* saat nilai *loss* tidak terjadi penurunan dan pemberhentian proses *training* tanpa menunggu keseluruhan *epoch* selesai. Pada Gambar 3.18 adalah hasil *training* tanpa menggunakan perubahan learning rate secara otomatis, sedangkan pada Gambar 3.19 adalah hasil *training* dengan perubahan *learning rate* otomatis.

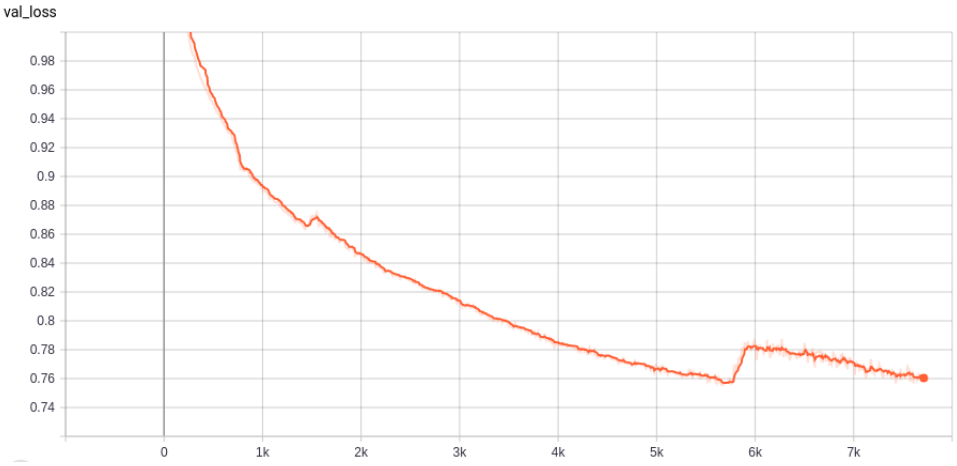


Gambar 3.17: Proses pengurangan *learning rate* saat *training*.

Pemberhentian proses *training* tersebut dilakukan bukan tanpa alasan, jika terjadi kenaikan *loss* khususnya pada proses validasi yang rawan terjadi kenaikan *loss* atau *loss validation* seperti pada Gambar 3.18. Saat terjadinya kenaikan *loss* juga diberikan toleransi sebesar 1.5×10^3 *epoch*, hal tersebut bertujuan untuk mendeteksi apakah dalam *range* 1.5×10^3 *epoch* akan terjadi



Gambar 3.18: Validasi *loss* saat 2×10^4 *epoch*.



Gambar 3.19: Validasi *loss* dengan pemberhentian otomatis.

penurunan *loss* yang lebih kecil dari sebelumnya. Jika tidak terjadi penurunan dalam kurun 1.5×10^3 *epoch* tersebut, maka proses *training* akan dihentikan seperti pada Gambar 3.19. Dapat dilihat pada Gambar 3.19 *training* dihentikan pada *epoch* antara 7×10^3 dan 8×10^3.

3.4.5 Pembuatan Model

Model yang dihasilkan dari proses *training* pada Sub-bab 3.4.4 dapat digunakan untuk menguji atau menampilkan hasil klasifikasi pada data *testing*, hasil tersebut akan ditampilkan dan dibahas pada Sub-bab 4.4. Pada Sub-bab ini akan lebih membahas mengenai model terbaik yang dihasilkan dari proses

training. Melanjutkan pembahasan sebelumnya pada Sub-bab 4.4 tentang *training* yang telah dilakukan meliputi penurunan *learning rate*, pemberhentian *training*, toleransi *epoch* saat tidak terjadinya penurunan *learing rate* dan yang paling berkataitan dengan Sub-bab ini pengambilan model terbaik. Pengambilan tersebut dilakukan pada saat *epoch* dikurangi dengan 1.5×10^3, dengan asumsi model terbaik diperoleh saat presentase *validation loss* paling rendah dan 1.5×10^3 tersebut adalah toleransi *epoch*. Selama 1.5×10^3 *epoch* tersebut, kemudian *training* berhenti, berarti tidak ditemukannya *validation loss* yang lebih rendah dalam *range* 1.5×10^3 *epoch* tersebut.

3.5 Klasifikasi Karakter Pemain dengan Neural Network Multiclass Classification

Terdapat banyak sekali jenis permainan RPG seperti yang sudah dijelaskan pada Sub-bab 2.4, tidak semua karakter yang dapat dimainkan oleh pemain berjumlah sangat banyak. Hanya beberapa genre RPG saja yang dapat menggunakan karakter pemain dengan jumlah sangat banyak. Jika mengacu genre permainan pada Sub-bab 2.4 dan judul permainan dijumpai saat ini permainan dengan genre JRPG dan MMORPG yang memungkinkan untuk memainkan banyak karakter dari pemain.

Pada pembahasan sebelumnya pada Sub-bab 3.4, Dota 2 adalah permainan dengan genre MOBA (*Multiplayer Online Battle Arena*) dengan beraneka karakter yang masing-masing dapat dipilih dan dimainkan oleh pemain. Jika dibandingkan dengan RPG yang dibahas pada Sub-bab 2.4, mungkin RPG dengan banyak karakter yang dapat dilakukan proses klasifikasi. Maka genre yang memungkinkan dilakukan proses klasifikasi adalah JRPG, TRPG dan MMORPG dengan *gameplay* yang mirip dengan JRPG. Sedangkan untuk WRPG, ARPG, SRPG, dan MMORPG dengan *gameplay* karakter tunggal yang tidak mirip dengan JRPG atau TRPG.

Maka digunakanlah data masukan yang digunakan untuk membuat karakter pemain yang sebelumnya sudah dibahas pada Sub-bab 3.2 pada Tabel

3.1. Data tersebut dibuat berdsarkan skenario seolah mendesain sebuah permainan. Pada permainan yang sebelumnya sempat dibahas pada Sub-bab 2.4 pasti terdapat juga karakter yang memiliki HP, *strength* yang tinggi dan biasa diikuti dengan *vitality* yang lumayan tinggi tetapi memiliki kecepatan atau *agility* rendah. Kemudian mungkin terdapat juga karakter yang memiliki HP sedang, *strength* yang rendah memiliki *magic* dan *agility* yang tinggi. Berikut adalah contoh skenario yang digunakan untuk mendesain karakter pemain berdasarkan tipe pada penelitian ini:

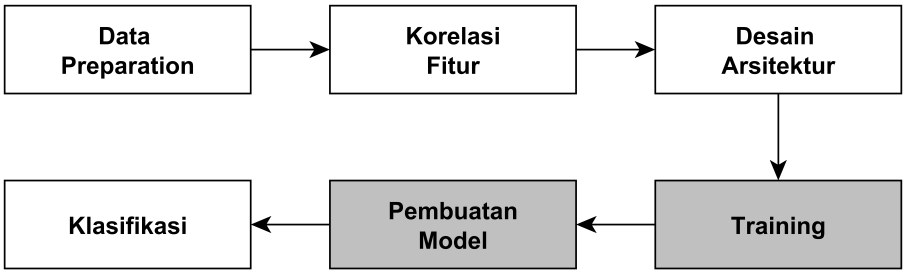
1). Misalkan terdapat karakter dengan HP, *strength* yang tinggi dan biasa diikuti dengan *endurance* yang lumayan tinggi tetapi memiliki kecepatan atau *agility* dan *magic* rendah, maka dalam skenario ini dikategorikan sebagai ***knight*** yang berfungsi sebagai penahan serangan dan bisa juga melakukan serangan.

2). Kemudian terdapat karakter yang memiliki *strength* yang tidak begitu tinggi, tetapi memiliki *endurance* yang cukup tinggi diikuti dengan *magic* yang tinggi tetapi *agility* rendah. Pada Skenario ini karakter tersebut adalah ***priest*** yang bertugas untuk menyembuhkan *party member* yang terluka atau mengalami pengurangan HP.

3). Karakter dengan *agility* atau kecepatan dan ketepatan yang tinggi dengan tingkat *luck* atau keberuntungan yang tinggi mengakibatkan berlipat gandanya *strength* atau kekuatan saat melakukan serangan. Pada skenario ini, karakter tersebut adalah ***assassin*** yang berperan sebagai penyerang dari *party* yang dapat membunuh karakter musuh dengan cepat dengan kemampuan fisik.

4). Terdapat juga karakter yang memiliki HP sedang, *strength* yang rendah, mungkin dengan *endurance* yang sedang juga, tetapi memiliki *magic* dan *agility* yang tinggi. Dalam skenario ini, karakter tersebut adalah ***magician*** yang berperan sebagai penyerang dari *party* dengan kemampuan *magic*.

Maka hal yang akan dilakukan disini adalah mencoba melakukan klasifikasi karakter berdasarkan atribut *gameplay* yang ada, dataset dari karakter pemain dapat dilihat pada Tabel 5.34 untuk data *training* dan Tabel 5.35 untuk *testing*. Selanjutnya adalah tahapan klasifikasinya seperti yang direpresentasikan pada Gambar 3.20.



Gambar 3.20: Proses klasifikasi Karakter Pemain.

Pada dasarnya Gambar 3.20 mirip dengan Gambar 3.14, dikarenakan proses yang dilalui sama persis, hanya saja pada bagian proses *training* dan pembuatan model yang ditandai dengan warna abu-abu penjelasannya akan banyak dilewati dikarenakan proses tersebut hampir sama dengan Gambar 3.14 yang kemudian dijelaskan pada Sub-bab 3.4.3 dan Sub-bab 3.4.5. Kemudian pada Gambar 3.14, data *pre-processing* diganti menjadi data *preparation* dikarenakan data dibuat sesuai dengan masukan data dalam pembuatan karakter pemain yang sudah dijelaskan pada Sub-bab 3.2.

3.5.1 Data Preparation

Dataset dari karakter pemain dapat dilihat pada Tabel 5.34 untuk data *training* dan Tabel 5.35 untuk data *testing*. Data tersebut diperoleh dari Tabel 3.1 dengan kenaikan pada masing-masing variabel misalkan pada HP yang diawali dengan *Start HP* yang kemudian akan naik sebesar *Next HP* begitu juga dengan MP. Sedangkan untuk kenaikan atribut *gameplay* seperti *Strength*, *Magic*, *Endurance*, *Speed* dan *Luck* diperoleh berdasarkan penambahan sejumlah variabel *Attributes to Assign* dangan hasil akhir sejumlah variabel *Max Attributes Value*.

Bila dijabarkan dalam proses kenaikan atribut *gameplay* setiap level maka hasilnya akan seperti Tabel 5.2 sampai dengan Tabel 5.5 yang diperoleh dari proses yang dijelaskan pada Sub-bab 3.2, dan bila setiap kenaikan atribut *gameplay* dirata-rata maka hasilnya akan seperti data di baris pertama pada Tabel 5.34 terlampir dalam LAMPIRAN. Sedangkan untuk baris lain pada Tabel 5.34 dan Tabel 5.35 dibuat satu-satu dengan variabel masukan seperti pada Tabel 3.1, namun dengan nilai yang berbeda. Nilai tersebut menyesuaikan dengan skenario yang dijelaskan diawal Sub-bab 3.5. Jumlah karakter pemain yang dibuat berjumlah 8 karakter yang dibagi menjadi 6 data *training* dan 2 data *testing*. Kemudian untuk kelemahan dari pemain tidak diguanakan sebagai fitur, dikarenakan nilainya yang cenderung kurang bervariasi yang mengakibatkan akan semakin mendekati nol saat dilakukan korelasi fitur.

3.5.2 Korelasi Fitur

Dilakukan proses korelasi fitur seperti yang dilakukan pada Sub-bab 3.4.2, hanya saja kali ini data yang dikorelasi fitur berbeda dengan data sebelumnya. Jika data sebelumnya adalah atribut *gameplay* untuk permainan Dota 2, maka data yang digunakan adalah atribut *gameplay* dari karakter pemain seperti penjelasan pada paragraf sebelumnya. Hasil perhitungan korelasi fitur menggunakan korelasi matrix dari atribut *gameplay* karakter pemain dapat dilihat pada Tabel 3.4.

Tabel 3.4: Hasil perhitungan korelasi matrix pada data pemain

Fitur	Korelasi Matix
type	1.000000
minHP	-0.629568
maxHp	-0.678790
hpGrowth	-0.408248
minMP	0.481794
maxMp	0.619225

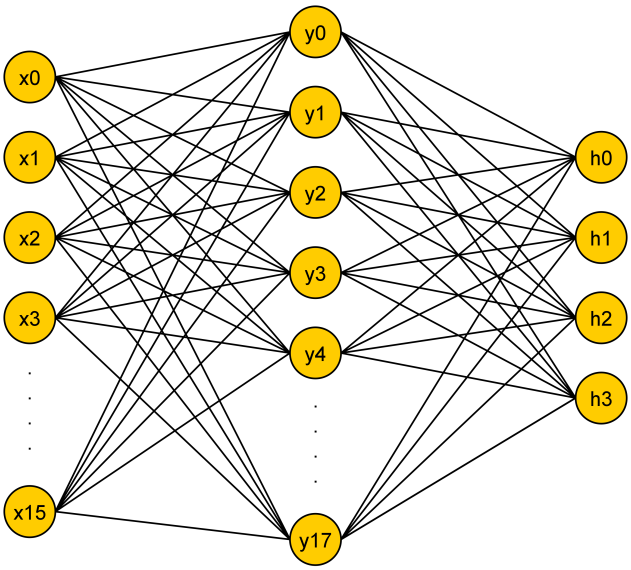
mpGrowth	0.653197
minStr	NaN
maxStr	-0.745061
strGrowth	-0.745061
minMag	NaN
maxMag	0.723322
magGrowth	0.723322
minEnd	NaN
maxEnd	-0.842417
endGrowth	-0.842417
minSpd	NaN
maxSpd	0.668815
spdGrowth	0.668815
minLuck	NaN
maxLuck	0.338762
luckGrowth	0.338762

Selanjutnya akan dibuang semua fitur yang memiliki nilai korelasi lebih kecil dari 0.1, -0.1 dan NaN seperti minStr atau *minimum strength*, minMag atau *minimum Magic*, minEnd atau *minimum endurance*, minSpd atau *minimum speed*, dan minLuck atau *minimum luck*. Sehingga didapatkan 16 fitur yang akan digunakan sebagai masukan untuk *Neural Network*.

3.5.3 Desain Arsitektur Neural Network

Arsitektur *Neural Network* yang akan dibuat mempunyai enam belas *neuron* pada layer masukan, delapan belas *neuron* pada *hidden layer* dengan fungsi aktivasi *sigmoid* dan lima *neuron* yang mewakili (*Knight*, *Priest*, *Assassin*, dan *Magician*) pada layer keluaran seperti pada Gambar 3.21 dengan fungsi aktivasi *Softmax* karena keluaran dari model yang dibuat bersifat distribusi probabilitas. Misalkan model kita 100% yakin jika karakter A adalah

Knight, maka keluaran dari model adalah [1, 0, 0, 0] dan jika model kita 50% yakin jika hero B adalah *Priest*, 25% yakin jika hero B adalah *Knight* atau *Magician*, maka output dari model ini adalah [0.25, 0.5, 0.25, 0]. Harus diingat bahwa total nilai dari distribusi probabilitas untuk semua target adalah 1. Sedangkan untuk *loss function* yang digunakan adalah *Cross Entropy* dan *Optimizer* yang digunakan adalah SGD (*Stochasstic Gradient Descent*) dengan *learning rate* sebesar 0.001. Sebagai catatan penambahkan *metrics accuracy* bertujuan untuk melihat seberapa bagus model dalam melakukan klasifikasi.



Gambar 3.21: Arsitektur *Neural Network* untuk klasifikasi karakter pemain.

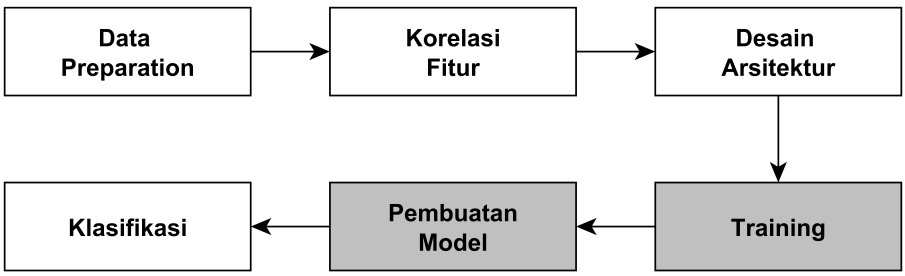
3.5.4 Training dan Pembuatan Model

Data hasil dari korelasi fitur pada Sub-bab 3.5.2, karena tipe yang ditunjukan pada kolom ke satu dengan kepala bertuliskan "Fitur" adalah pedoman hasil prediksi maka tidak akan diikutkan proses *training*. Pada kolom *type* tipe karakter dinyatakan dengan 0, 1, 2, dan 3 yang menyatakan *Knight*, *Priest*, *Assassin* dan *Magician*. Nantinya keluaran dari hasil *training* yang menunjukan prediksi tipe dari karakter dinyatakan dalam bentuk *one-hot vector*: *Knight* $\rightarrow [1,0,0,0]$, *Priest* $\rightarrow [0,1,0,0]$, *Assassin* $\rightarrow [0,0,1,0]$ dan *Magician* $\rightarrow [0,0,0,1]$. Kemudian untuk optimasi dan pembuatan model sama seperti

Sub-bab 3.4.5. Model yang dihasilkan dari proses *training* pada Sub-bab 3.5.4 dapat digunakan untuk menguji atau menampilkan hasil klasifikasi pada data *testing* yang dibahas pada Sub-bab 4.4.

3.6 Klasifikasi Karakter Musuh dengan Neural Network Multiclass Classification

Hal yang akan dilakukan adalah dilakukannya klasifikasi karakter musuh berdasarkan atribut *gameplay* yang ada, dataset dari karakter musuh dapat dilihat pada Tabel 5.16 sampai dengan Tabel 5.30 untuk musuh dengan urutan ke 1 sampai 280 digunakan sebagai data *training* dan Tabel 5.16 sampai dengan Tabel 5.30 untuk musuh dengan urutan ke 281 sampai paling akhir digunakan sebagai data *testing*. Tanpa perlu adanya skenario dikarenakan jumlah musuh yang banyak dan bermacam-macam yang telah dibuat melalui serangkaian proses pada Sub-bab 3.3 yang dilanjukan oleh proses pada Gambar 3.22.



Gambar 3.22: Proses klasifikasi Karakter Musuh.

Pada dasarnya Gambar 3.22 mirip dengan Gambar 3.14, dikarenakan proses yang dilalui sama persis, hanya saja pada bagian proses *training* dan pembuatan model yang ditandai dengan warna abu-abu karena penjelasannya akan banyak dilewati dikarenakan proses tersebut hampir sama dengan Gambar 3.14 yang sudah dijelaskan pada Sub-bab 3.4.3 dan Sub-bab 3.4.5. Kemudian pada Gambar 3.14, data *pre-processing* diganti menjadi data *preparation* dikarenakan data dibuat secara manual sesuai dengan masukan data dalam membuat karakter pemain yang sudah dijelaskan pada Sub-bab 3.3.

3.6.1 Data Preparation

Dataset dari karakter musuh dapat dilihat pada Tabel 5.16 sampai dengan Tabel 5.30 untuk musuh dengan urutan ke 1 sampai 280 digunakan sebagai data *training* dan Tabel 5.16 sampai dengan Tabel 5.30 untuk musuh dengan urutan ke 281 sampai 400 digunakan sebagai data *testing* dan *testing*. Data tersebut diperoleh dari Tabel 3.2 yang kemudian dijelaskan setiap proses pembuatan data tersebut pada Sub-bab 3.2. Kemudian untuk kelemahan dari musuh tidak diguanakan sebagai fitur sama seperti pada karakter pemain, hal tersebut dikarenakan nilainya yang cenderung kurang bervariasi yang mengakibatkan akan semakin mendekati nol saat dilakukan korelasi fitur.

Maka digunakanlah data masukan yang digunakan untuk membuat karakter musuh yang sebelumnya sudah dibahas pada Sub-bab 3.3 pada Tabel 3.2. Data tersebut dibuat berdasarkan skenario seolah mendesain sebuah permainan. Pada permainan yang sebelumnya sempat dibahas pada Sub-bab 3.1.2 pasti terdapat juga karakter musuh yang susah atau mudah dikalahkan, memiliki kemampuan *magic* yang tinggi, sedang, atau rendah, memiliki kemampuan *strength* yang tinggi, sedang, atau rendah, dan lain sebagainya. Berikut adalah contoh skenario yang digunakan untuk mendesain karakter musuh berdasarkan tipe pada penelitian ini:

1). Misalkan terdapat karakter dengan HP, *strength* yang sangat tinggi sampai dengan sedang dan biasa diikuti dengan *endurance* juga tetapi memiliki kecepatan atau *agility* dan *magic* rendah, maka dalam skenario ini dikategorikan sebagai ***Hard Strength***.

2). Misalkan terdapat karakter dengan HP, *strength* yang lumayan tinggi sampai dengan sedang dan biasa diikuti dengan *endurance* juga tetapi memiliki kecepatan atau *agility* dan *magic* rendah, maka dalam skenario ini dikategorikan sebagai ***Soft Strength***.

3). Misalkan terdapat karakter dengan *strength* dari sedang sampai dengan rendah dan biasa diikuti dengan *endurance* juga tetapi memiliki *magic*

yang sangat tinggi, maka dalam skenario ini dikategorikan sebagai ***Hard Magic***.

4). Misalkan terdapat karakter dengan *strength* dari sedang sampai dengan rendah dan biasa diikuti dengan *endurance* juga tetapi memiliki *magic* yang lumayan tinggi sampai dengan sedang, maka dalam skenario ini dikategorikan sebagai ***Soft Magic***.

5). Misalkan terdapat karakter nilai atribut *gameplay* yang tidak menentu dengan *strength* dari rendah sampai dengan tinggi, begitu juga dengan magic, namum dalam distribusi atribut *gameplay* tetap seimbang. Maka dalam skenario ini dikategorikan sebagai ***Mixed***.

3.6.2 Korelasi Fitur

Dilakukan proses korelasi fitur seperti yang dilakukan pada Sub-bab 3.4.2, hanya saja kali ini data yang dikorelasi fitur berbeda dengan data sebelumnya. Jika data sebelumnya adalah atribut *gameplay* untuk permainan Dota 2, maka data yang digunakan adalah atribut *gameplay* dari karakter musuh seperti penjelasan pada paragraf sebelumnya. Hasil perhitungan korelasi fitur menggunakan korelasi matrix dari atribut *gameplay* karakter pemain dapat dilihat pada Tabel 3.5.

Tabel 3.5: Hasil perhitungan korelasi matrix pada data musuh

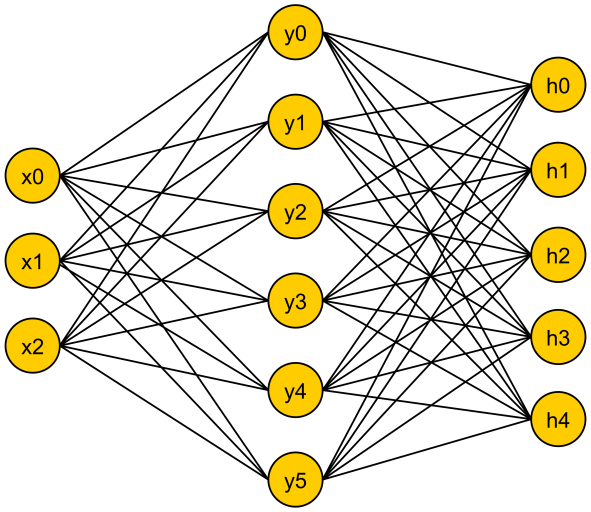
Fitur	Korelasi Matrix
type	1.000000
Levels	0.089835
HP	0.140933
MP	-0.250142
Strength	0.056565
Magic	0.051114
Endurance	-0.290492

Speed	0.043601
Luck	0.076972

Selanjutnya akan dibuang semua fitur yang memiliki nilai korelasi lebih kecil dari 0.1, -0.1 dan NaN sehingga yang mampu dijadikan sebagai fitur hanya HP, MP dan *Endurance*. Sehingga perolehlah tiga fitur yang akan digunakan sebagai masukan untuk *Neural Network* diantaranya adalah HP, MP, dan *Endurance*. Jadi dari sekian banyaknya fitur, hanya tersisa tiga fitur yang dapat digunakan.

3.6.3 Desain Arsitektur Neural Network

Arsitektur *Neural Network* yang akan dibuat mempunyai tiga *neuron* pada layer masukan, enam *neuron* pada *hidden layer* dengan fungsi aktivasi *sigmoid* dan lima *neuron* yang mewakili (*Knight*, *Priest*, *Assassin*, dan *Magician*) pada layer keluaran seperti pada Gambar 3.23 dengan fungsi aktivasi *Softmax* karena keluaran dari model yang dibuat bersifat distribusi probabilitas dari seluruh nilai target.



Gambar 3.23: Arsitektur *Neural Network* untuk klasifikasi karakter musuh.

Misalkan model kita 100% yakin jika karakter A adalah *Hard Strength*, maka keluaran dari model adalah [1, 0, 0, 0, 0] dan jika model kita 50% yakin

jika hero B adalah *Soft Strength*, 25% yakin jika hero B adalah *Hard Strength* atau *Hard Magic*, maka keluaran dari model ini adalah [0.25, 0.5, 0.25, 0, 0]. Harus diingat bahwa total nilai dari distribusi probabilitas untuk semua target adalah 1. Sedangkan untuk *loss function* yang digunakan adalah *Cross Entropy* dan *Optimizer* yang digunakan adalah SGD (*Stochasstic Gradient Descent*) dengan *learning rate* sebesar 0.001. Sebagai catatan penambahkan *metrics accuracy* bertujuan untuk melihat seberapa bagus model ini dalam melakukan klasifikasi.

3.6.4 Training dan Pembuatan Model

Data hasil dari korelasi fitur pada Sub-bab 3.6.2, karena tipe yang ditunjukan pada kolom ke satu dengan kepala bertuliskan "Fitur" adalah pedoman hasil prediksi maka tidak akan diikutkan proses *training*. Pada kolom *type* tipe karakter dinyatakan dengan 0, 1, 2, 3, dan 4 yang menyatakan *Mixed*, *Hard Strength*, *Soft Strength*, *Hard Magic* dan *Soft Magic*. Nantinya keluaran dari hasil *training* yang menunjukan prediksi tipe dari karakter dinyatakan dalam bentuk *one-hot vector*: *Mixed* $\rightarrow [1,0,0,0,0]$, *Hard Strength* $\rightarrow [0,1,0,0,0]$, *Soft Strength* $\rightarrow [0,0,1,0,0]$, *Hard Magic* $\rightarrow [0,0,0,1,0]$, dan *Soft Magic* $\rightarrow [0,0,0,0,1]$. Kemudian untuk optimasi dan pembuatan model sama seperti Sub-bab 3.4.5. Model yang dihasilkan dari proses *training* pada Sub-bab 3.6.4 dapat digunakan untuk menguji atau menampilkan hasil klasifikasi pada data testing yang dibahas pada Sub-bab 4.4.

BAB 4

HASIL DAN PEMBAHASAN

Inti dari BAB 4 pada penelitian ini adalah pengujian dengan berbagai parameter atribut *gameplay* yang dihasilkan untuk pemain dan musuh. Jika pada BAB 3 dalam proses analisan desain permainan RPG, permainan RPG digolongkan berdasarkan pertarungan menjadi *real-time* dan *turn-based*.

Kemudian jika melihat jumlah karakter yang dimainkan oleh pemain maka permainan RPG dapat digolongkan menjadi *single-character* dan *multi-character*. Hal ini diperlukan dalam pengujian pembuatan atribut *gameplay* yang dihasilkan, genre dari RPG yang termasuk *single-character* adalah WRPG, ARPG, SRPG dan MMORPG, sedangkan yang termasuk *multi-character* adalah TRPG dan JRPG. Sedangkan untuk karakter musuh dianggap sama. Dalam beberapa permainan bergenre RPG, pemain dan musuh tidak menggunakan sistem elemen sebagai kelemahan. Biasanya pada permainan RPG dengan genre ARPG, SRPG dan sebagian WRPG, sedangkan untuk TRPG dan JRPG hampir sebagian besar memakai sistem elemen sebagai kelemahan dalam sistem pertarungan.

4.1 Hasil Atribut Gameplay Pemain (Single-Character)

Pada bagian ini setiap langkah yang akan diuji sudah dijelaskan pada Sub-bab 3.2, dimana pada bagian ini membahas tentang pembuatan atribut *gameplay* pemain untuk permaian bergenre RPG. Melalui berbagai proses seperti yang dijelaskan pada Sub-bab 3.2, maka pada beberapa Sub-bab dibawah ini adalah langkah-langkah dalam proses pengujiannya. Maksud dari *single-character* disini adalah pemain hanya menjalankan satu buah karakter saja selama permainan, contoh dari RPG jenis ini biasanya ada pada WRPG, SRPG, ARPG, dan MMORPG.

4.1.1 Hasil Distibusi Level, HP, dan MP Pemain

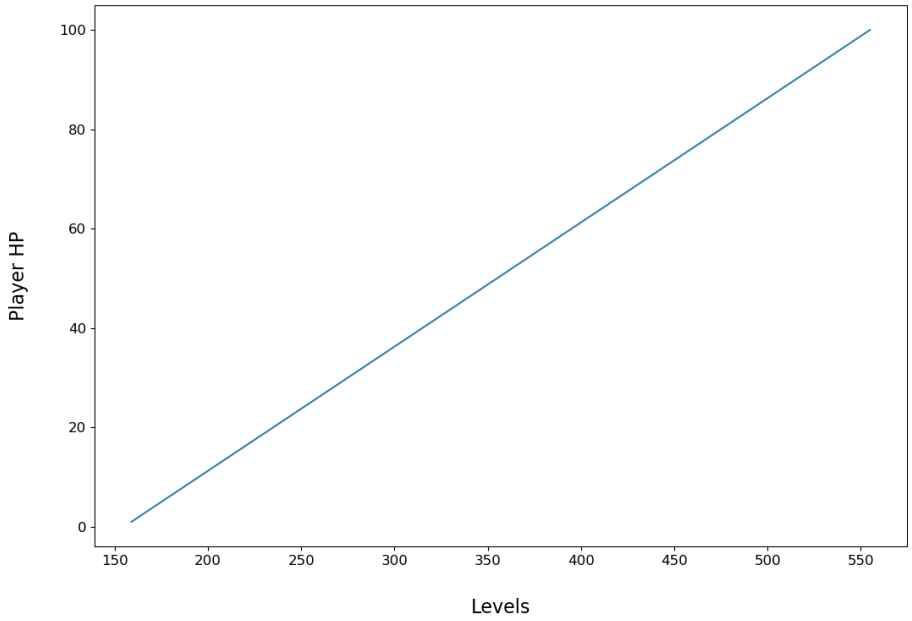
Seperti yang sudah dijelaskan pada Sub-bab 3.2.1, yang membahas tentang pembuatan level, HP dan MP pemain untuk permaian dengan genre RPG. Berikut adalah hasil dari prosess pembuatan Level, HP, dan MP yang diperoleh dari perhitungan pada persamaan 3.2 dan 3.3 dengan data masukan pada Tabel 3.1 yang kemudian menghasilkan data seperti yang ditunjukkan pada Tabel 4.1.

Tabel 4.1: Hasil Perhitungan HP dan MP

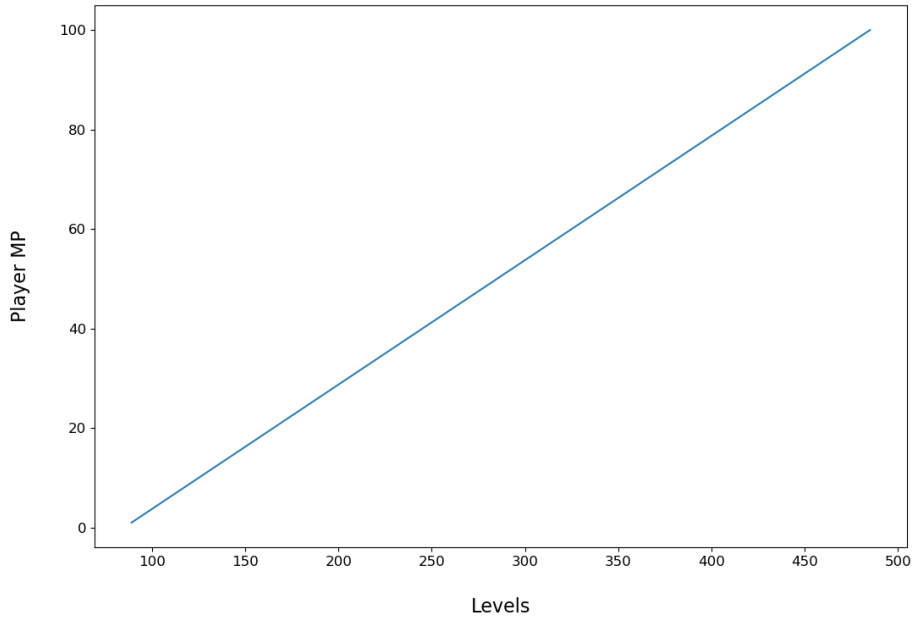
Levels	**HP**	**MP**
1	159	89
2	163	93
3	167	97
4	171	101
5	175	105
6	179	109
7	183	113
...	...	...
100	**555**	**485**

Hasil perhitungan tersebut terlihat membentuk pola linier, yang mana nilai HP dan MP terus naik secara konstan ke atas sesuai dengan kenaikan levelnya seperti yang direpresentasikan pada Gambar 4.1 dan Gambar 4.2. Jika melihat Gambar 4.1 dengan jumlah HP dari pemain yang terus naik mengikuti pola yang dihasilkan pada Tabel 4.1, yang mana nilai tersebut diperoleh dari inisiasi variabel pada Tabel 3.1 yaitu *Max Level*, *Start HP* dan *Next HP*. Variabel-variabel tersebut dihitung dengan menggunakan persamaan 3.2 agar membentuk pola kenaikan HP setiap levelnya seperti yang ditunjukan pada Gambar 4.1. Sama seperti pada Gambar 4.1, pada Gambar 4.2 dengan jumlah MP dari pemain yang terus naik mengikuti pola yang dihasilkan pada Tabel

4.1, yang mana nilai tersebut diperoleh dari inisiasi variabel pada Tabel 3.1 yaitu *Max Level*, *Start MP* dan *Next MP*. Kemudian variabel-variabel tersebut dihitung dengan menggunakan persamaan 3.3 agar membentuk pola kenaikan MP setiap levelnya seperti yang ditunjukan pada Gambar 4.2.



Gambar 4.1: Kenaikan HP setiap levelnya.



Gambar 4.2: Kenaikan MP setiap levelnya.

Pada setiap proses pengujian dalam Sub-bab ini, yang dihasilkan pada Tabel 4.1 menyajikan sebagian data saja. Hasil selengkapnya dapat dilihat pada Tabel 5.1 dalam LAMPIRAN yang kemudian keseluruhan hasil tersebut divisualisasikan pada Gambar 4.1.

4.1.2 Hasil Distibusi Elemen dan Kelemahan Pemain

Pada bagian ini hanya akan mengulas tentang distribusi elemen dan kelemahan pemain pada Sub-Bab 3.2.2, yang sejatinya didefinisikan secara manual pada program yang dibuat dan dijelaskan pada Sub-Bab 3.2. Di mana elemen pada permainan RPG bersifat opsional, tidak dibatasi apakah permainan RPG tersebut tergolong ke dalam *turn-base* atau *real-time*, tergolong *singe-character* atau *multi-character*. Hal tersebut murni bergantung pada pengembang dan desainer permainan. Maka dari itu melalui program yang dibuat pada Sub-Bab 3.2 hal ini diwujudkan dalam bentuk opsi yang dapat digunakan atau ditinggalkan, sesuai kebutuhan pengembang atau desainer permainan. Bila opsi distribusi elemen digunakan maka hasilnya akan tampak seperti pada Tabel 4.2, dengan asumsi tidak adanya perubahan elemen dari karakter pemain dalam permainan.

Tabel 4.2: Distribusi elemen pada karakter pemain

Levels	Water	Wind	Earth	Fire
1	2	0	0	1
2	2	0	0	1
3	2	0	0	1
4	2	0	0	1
5	2	0	0	1
6	2	0	0	1
7	2	0	0	1
...	...	...	...	...
100	**2**	**0**	**0**	**1**

Pada setiap proses pengujian dalam Sub-bab ini, yang dihasilkan pada Tabel 4.2 menyajikan sebagian data saja. Hasil selengkapnya dapat dilihat pada Tabel 5.2 sampai dengan Tabel 5.5 dalam LAMPIRAN. Hal seperti penggunaan elemen pada pemain seperti pembahasan pada Sub-bab ini an sebelumnya, biasanya dijumpai pada permain RPG dengan Sub-genre JRPG, TRPG, WRPG dan MMORPG. Namun itu juga tidak semua permainan RPG dengan genre tersebut menggunakannya, hanya yang sering dijumpai saja.

4.1.3 Hasil Distibusi Atribut Gameplay Pemain

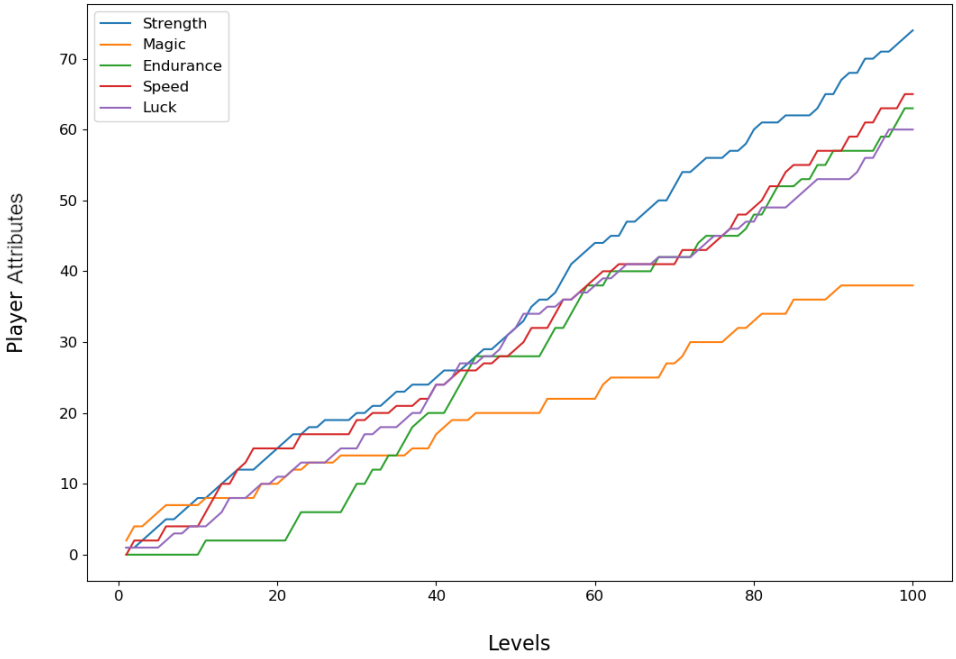
Pada bagian ini setiap langkah yang akan diuji sudah dijelaskan pada Sub-bab 3.2.3, dimana pada bagian ini membahas tentang pembuatan atribut *gameplay* pemain untuk permaian RPG. Melalui berbagai proses seperti yang dijelaskan pada Sub-bab 3.2.3, melalui data masukan pada Tabel 3.1 kemudian persamaan 3.7 dan beberapa persamaan lain yang disebutkan pada Sub-bab 3.2.3 dihasilkan data seperti yang ditunjukan pada Tabel 4.3.

Tabel 4.3: Distribusi atribut *gameplay* pada karakter pemain

Levels	**Strength**	**Magic**	**Endurance**	**Speed**	**Luck**
1	1	2	0	0	1
2	0	2	0	2	0
3	1	0	0	0	0
4	1	1	0	0	0
5	1	1	0	0	0
6	1	1	0	2	1
7	0	0	0	0	1
...	...	...	...	...	...
100	**74**	**38**	**63**	**65**	**60**

Pada Tabel 4.3 adalah data distribusi atribut *gameplay* dari pemain yang dihasilakn melalui penambahan nilai pada atribut *gameplay* secara acak pada setiap atribut *gameplay*. Seperti yang dijelaskan pada persaamaan 2.15, 3.6,

dan persamaan 3.7 saat nilai atribut *gameplay* ditambahkan secara acak dari 1 sampai dengan 2 pada level yang berjarak antara 1 sampai dengan 100. Selanjutnya representasi dari hasil penambahan atribut *gameplay* tersebut yang ditunjukan melalui Gambar 4.3 yang mana nilai dari setiap atribut *gameplay* terus naik sesuai dengan level pemain yang juga terus naik.



Gambar 4.3: Kenaikan atribut *gameplay* pemain setiap levelnya.

Pada setiap proses pengujian dalam Sub-bab ini, yang dihasilkan pada Tabel 4.3 menyajikan sebagian data saja. Hasil selengkapnya dapat dilihat pada Tabel 5.2 sampai dengan Tabel 5.5 dalam LAMPIRAN yang kemudian keseluruhan hasil tersebut divisualisasikan pada Gambar 4.3.

4.2 Hasil Atribut Gameplay Pemain (Multi-Character)

Pada bagian ini setiap langkah yang akan diuji sudah dijelaskan pada Sub-bab 3.2, dimana pada bagian ini membahas tentang pembuatan atribut *gameplay* pemain untuk permaian dengan genre RPG. Melalui berbagai proses seperti yang dijelaskan pada Sub-bab 3.2, maka pada beberapa Sub-bab dibawah ini adalah langkah-langkah dalam proses pengujiannya. Maksud dari Multi-Character disini adalah pemain dapat menjalankan beberapa karakter

sekaligus, menjalankan dengan mode *turn-based* atau *switch* antar karakter seperti yang dijelaskan pada Sub-Bab 2.4.2. Maka dari itu dibuatlah Tabel masukan baru pada Tabel 4.4 dan 4.5 yang mirip seperti Tabel 3.1. Hal ini bertujuan membandingkan perbedaan atribut *gameplay* yang dihasilkan pada tiga karakter pemain.

Tabel 4.4: Data masukan untuk pembuatan atribut *gameplay* karakter pertama.

Variabel	Input
Start Level	1
Max Level	100
Start HP	224
Next HP	228
Start MP	100
Next MP	102
List Element	['Phys', 'Water', 'Wind', 'Earth', 'Fire']
List Weaknesess	[0, 0, 2, 1, 0]
List Attributes Name	['Strength', 'Magic', 'Endurance', 'Speed', 'Luck']
Max Attributes Value	[88, 32, 81, 43, 56]
Attributes to Assign	[2, 1]

Tabel 4.5: Data masukan untuk pembuatan atribut *gameplay* karakter kedua.

Variabel	Input
Start Level	1
Max Level	100
Start HP	223
Next HP	226
Start MP	154
Next MP	158
List Element	['Phys', 'Water', 'Wind', 'Earth', 'Fire']
List Weaknesess	[0, 1, 0, 0, 2]

List Attributes Name	['Strength', 'Magic', 'Endurance', 'Speed', 'Luck']
Max Attributes Value	[60, 68, 57, 58, 57]
Attributes to Assign	[2, 1]

Kemudian dari Tabel 4.4 dan Tabel 4.5 dihasilkan dua atribut *gameplay* yang berbeda untuk dua karakter dari pemain. Pada mulanya pemilihan atribut *gameplay* pemain yang dinyatakan pada ke dua Table tersebut bukanlah tidak beralasan, bila melihat pada Tabel yang sebelumnya dipakai untuk pengujian *single-character* pada Sub-bab 4.1 dan penjabaran metodologi pembuatan atribut *gameplay* karakter pemain pada Sub-bab 3.2 tentu saja masih berkaitan. Ketiga atribut *gameplay* tersebut membentuk sebuah *party member* dalam permainan JRPG atau bahkan TRPG, hal ini sengaja dibuat dikarenakan hanya genre permainan RPG tersebut yang memungkinkan pemain menggunakan lebih dari satu karakter. Penjelasaan dan penggunaan dari ke empat atribut *gameplay* tersebut akan diperjelas pada beberapa Sub-bab berikut ini.

4.2.1 Hasil Distibusi Level, HP, dan MP Pemain

Seperti yang sudah dijelaskan pada Sub-bab 3.2.1, dimana pada bagian ini membahas tentang pembuatan level, HP dan MP pemain untuk permaian dengan genre RPG. Sama seperti pembahasan pada Sub-bab 4.1.1, hanya saja pada bagian ini, karakter pemain berjumlah lebih banyak dari pembahasan sebelumnya. Berikut adalah hasil dari prosess pembuatan Level, HP, dan MP yang diperoleh dari perhitungan pada persamaan 3.2 dan 3.3 dengan data masukan pada Tabel 4.4 dan Tabel 4.5 yang kemudian menghasilkan data seperti yang ditunjukkan pada Tabel 4.6 dan Tabel 4.7.

Tabel 4.6: Hasil Perhitungan HP dan MP karakter pertama

Levels	**HP**	**MP**
1	224	100

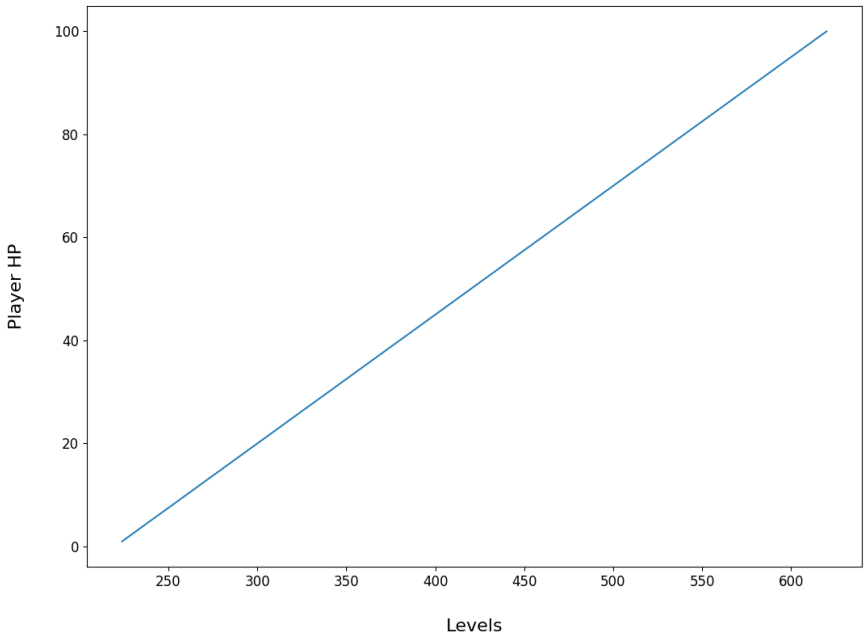
2	228	102
3	232	104
4	236	106
5	240	108
6	244	110
7	248	112
...	...	...
100	**620**	**298**

Tabel 4.7: Hasil Perhitungan HP dan MP karakter kedua

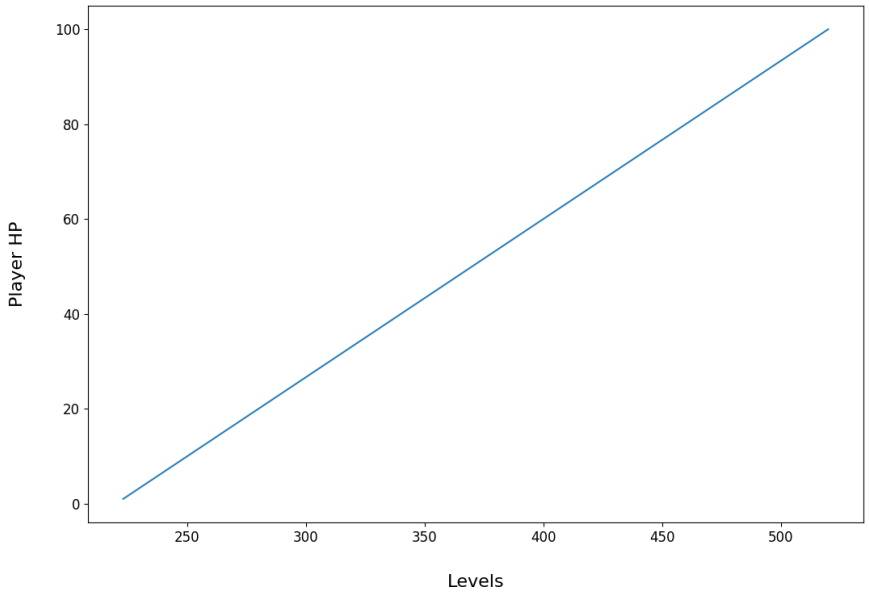
Levels	**HP**	**MP**
1	223	154
2	226	158
3	229	162
4	232	166
5	235	170
6	238	174
7	241	178
...	...	...
100	**520**	**550**

Hasil perhitungan tersebut terlihat membentuk pola linier, yang mana nilai HP terus naik secara konstan ke atas sesuai dengan kenaikan levelnya seperti yang direpresentasikan pada Gambar 4.4, dan Gambaer 4.5. Jika melihat Gambar 4.4 dan Gambar 4.5 dengan jumlah HP dari pemain yang terus naik mengikuti pola yang dihasilkan pada Tabel 4.6 dan Tabel 4.7, yang mana nilai tersebut diperoleh dari inisiasi variabel pada Tabel 4.4 dan Tabel 4.5 yaitu *Max Level*, *Start* HP dan *Next* HP. Variabel-variabel tersebut dihitung

dengan menggunakan persamaan 3.2 agar membentuk pola kenaikan HP setiap levelnya seperti yang ditunjukan pada Gambar 4.4 dan Gambar 4.5.



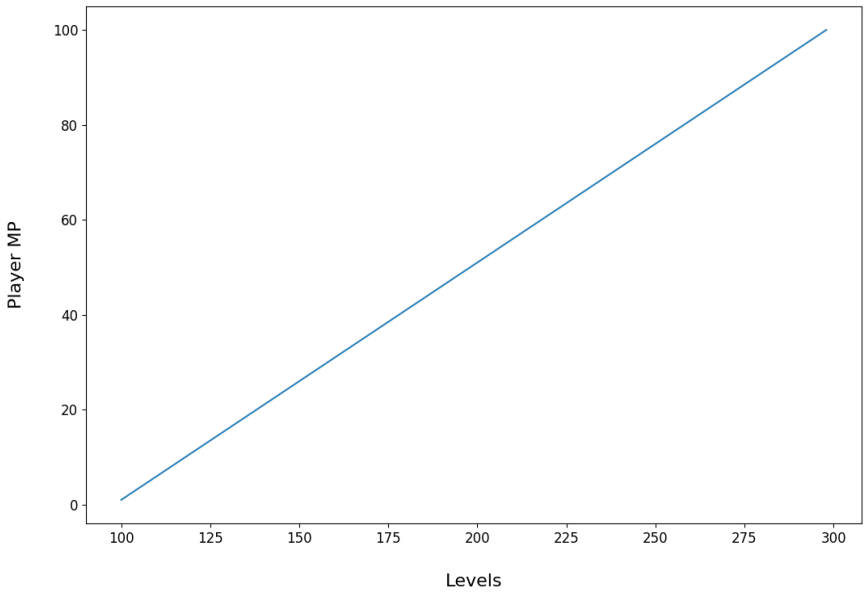
Gambar 4.4: Kenaikan HP setiap levelnya pada karakter pertama.



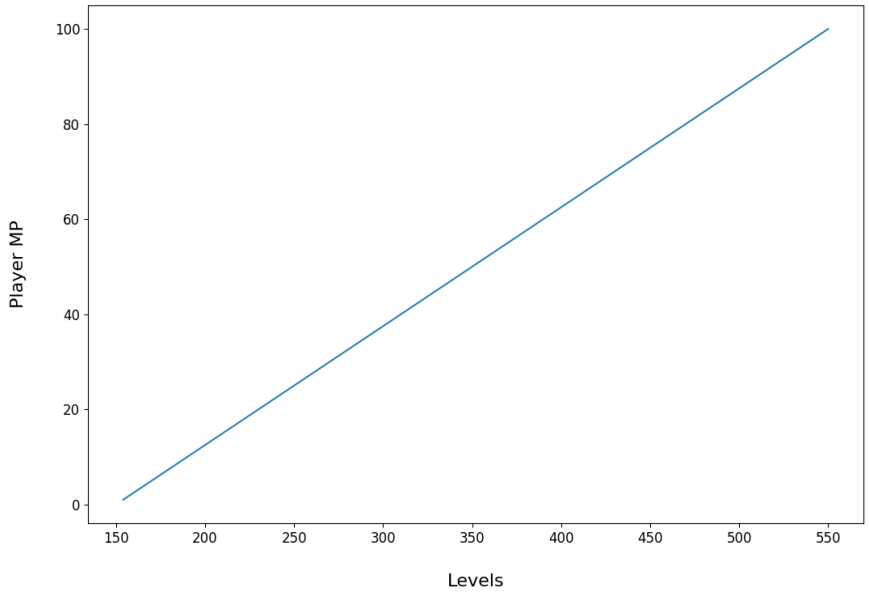
Gambar 4.5: Kenaikan HP setiap levelnya pada karakter kedua.

Sama seperti pada Gambar 4.6 dan Gambar 4.7 dengan jumlah MP dari pemain yang terus naik mengikuti pola yang dihasilkan pada Tabel 4.6 dan

Tabel 4.7, yang mana nilai tersebut diperoleh dari inisiasi variabel pada Tabel 3.1 yaitu *Max Level*, *Start* MP dan *Next* MP. Kemudian variabel-variabel tersebut dihitung dengan menggunakan persamaan 3.3 agar membentuk pola kenaikan MP setiap levelnya seperti yang ditunjukan pada Gambar 4.6 dan Gambar 4.7.



Gambar 4.6: Kenaikan MP setiap levelnya pada karakter pertama.



Gambar 4.7: Kenaikan MP setiap levelnya pada karakter kedua.

Pada setiap proses pengujian dalam Sub-bab ini, yang dihasilkan pada Tabel 4.6 dan 4.7 menyajikan sebagian data saja. Hasil selengkapnya dapat dilihat pada Tabel 5.6 untuk karakter pertama dan pada 5.11 untuk karakter kedua yang terlampir dalam LAMPIRAN. Kemudian keseluruhan hasil tersebut divisualisasikan pada Gambar 4.4, dan Gambar 4.5 untuk HP, Gambar 4.6, dan Gambar 4.7 untuk MP.

4.2.2 Hasil Distibusi Elemen dan Kelemahan Pemain

Sama seperti penjelasan pada Sub-bab 4.1.2, bila opsi distribusi elemen digunakan maka hasilnya akan tampak seperti pada Tabel 4.8 yang mengacu pada inputan dari Tabel 4.4 untuk karakter pemain pertama, sedangkan unt karakter pemain kedua hasilnya seperti pada Tabel 4.9 yang mengacu pada inputan dari Tabel 4.5. Keduanya diasumsikan tidak adanya perubahan elemen dari karakter pemain dalam permainan.

Tabel 4.8: Distribusi elemen pada karakter pemain pertama

Phys	Water	Wind	Earth	Fire
0	0	2	1	0
0	0	2	1	0
0	0	2	1	0
0	0	2	1	0
0	0	2	1	0
0	0	2	1	0
...	...	...	...	...
0	**0**	**2**	**1**	**0**

Tabel 4.9: Distribusi elemen pada karakter pemain kedua

Phys	Water	Wind	Earth	Fire
0	1	0	0	2
0	1	0	0	2

0	1	0	0	2
0	1	0	0	2
0	1	0	0	2
0	1	0	0	2
...	...	...	...	...
0	**0**	**2**	**1**	**0**

Pada setiap proses pengujian dalam Sub-bab ini, yang dihasilkan pada Tabel 4.8 dan 4.9 menyajikan sebagian data saja. Hasil selengkapnya dapat dilihat pada Tabel 5.2 sampai dengan Tabel 5.5 dalam LAMPIRAN. Hal seperti penggunaan elemen pada pemain seperti pembahasan pada Sub-bab ini an sebelumnya, biasanya dijumpai pada permain RPG dengan Sub-genre JRPG, TRPG, WRPG dan MMORPG. Namun itu juga tidak semua permainan RPG dengan genre tersebut menggunakannya, hanya yang sering dijumpai saja.

4.2.3 Hasil Distibusi Atribut Gameplay Pemain

Pada bagian ini setiap langkah yang akan diuji sudah dijelaskan pada Sub-bab 3.2.3, dimana pada bagian ini membahas tentang pembuatan atribut *gameplay* pemain untuk permaian RPG. Melalui berbagai proses seperti yang dijelaskan pada Sub-bab 3.2.3, melalui data masukan pada Tabel 4.4 untuk karakter pertama dan Tabel 4.4 untuk karakter kedua, kemudian persamaan 3.7 dan beberapa persamaan lain yang disebutkan pada Sub-bab 3.2.3 dihasilkan data seperti yang ditunjukan pada Tabel 4.10 dan Tabel 4.11.

Tabel 4.10: Distribusi atribut *gameplay* pada karakter pertama pemain

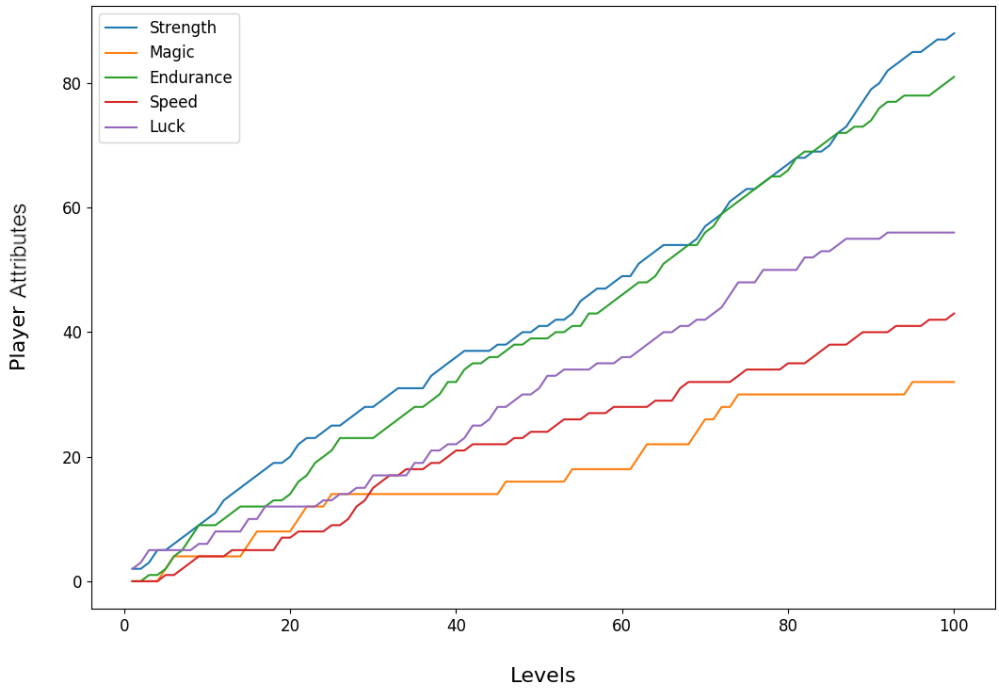
Levels	Strength	Magic	Endurance	Speed	Luck
1	1	1	0	0	0
2	2	0	2	0	0
3	1	0	1	1	0
4	2	2	2	1	0

5	0	0	0	0	0
6	2	0	0	1	0
7	0	0	0	1	0
...	...	...	...	...	...
100	**74**	**38**	**63**	**65**	**60**

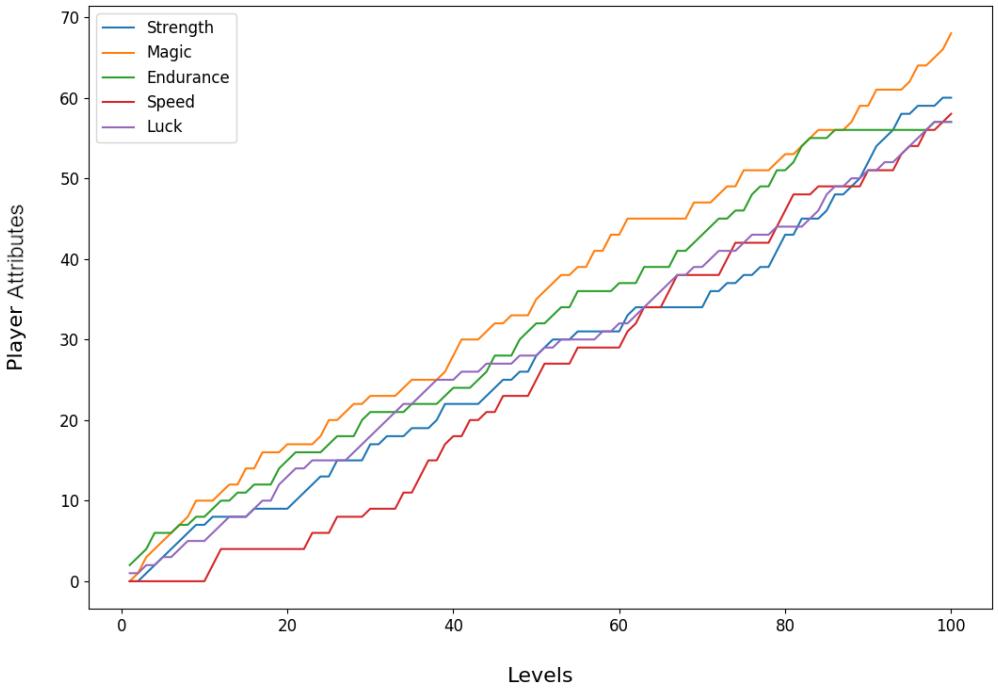
Tabel 4.11: Distribusi atribut *gameplay* pada karakter pemain

Levels	**Strength**	**Magic**	**Endurance**	**Speed**	**Luck**
1	1	2	0	0	1
2	0	2	0	2	0
3	1	0	0	0	0
4	1	1	0	0	0
5	1	1	0	0	0
6	1	1	0	2	1
7	0	0	0	0	1
...	...	...	...	...	...
100	**60**	**68**	**57**	**58**	**57**

Pada Tabel 4.10 dan Tabel 4.11 yang masing-masing adalah data atribut *gameplay* dari karakter pemain pertama dan kedua yang dihasilakn melalui penambahan nilai pada atribut *gameplay* secara acak pada setiap atribut *gameplay*. Seperti yang dijelaskan pada persaamaan 2.15, 3.6, dan persamaan 3.7 saat nilai atribut *gameplay* di tambahkan secara acak dari 1 sampai dengan 2 pada level yang berjarak antara 1 sampai dengan 100. Selanjutnya representasi dari hasil penambahan atribut *gameplay* tersebut yang ditunjukan melalui Gambar 4.8 dan Gambar 4.9 yang masing-masing adalah karakter pertama dan kedua dari pemain secara berurutan, nilai dari setiap atribut *gameplay* pada masing-masing karakter terus naik sesuai dengan level dari masing-masing karakter yang juga terus naik.



Gambar 4.8: Kenaikan atribut *gameplay* karakter pertama pemain setiap levelnya.



Gambar 4.9: Kenaikan atribut *gameplay* karakter kedua pemain setiap levelnya.

Pada setiap proses pengujian dalam Sub-bab ini, yang dihasilkan pada Tabel 4.3 menyajikan sebagian data saja. Hasil selengkapnya dapat dilihat

pada Tabel 5.2 sampai dengan Tabel 5.5 dalam LAMPIRAN yang keseluruhan hasil tersebut divisualisasikan pada Gambar 4.8 dan Gambar 4.8.

4.3 Hasil Atribut Gameplay Musuh

Pada bagian ini setiap langkah yang akan diuji sudah dijelaskan pada Bagian 3.3, dimana pada bagian ini membahas tentang pembuatan atribut *gameplay* musuh untuk permaian dengan genre RPG. Melalui berbagai proses seperti yang dijelaskan pada Bagian 3.2, maka pada beberapa Sub-bab dibawah ini adalah langkah-langkah dalam proses pengujiannya.

4.3.1 Hasil Distibusi Level Musuh

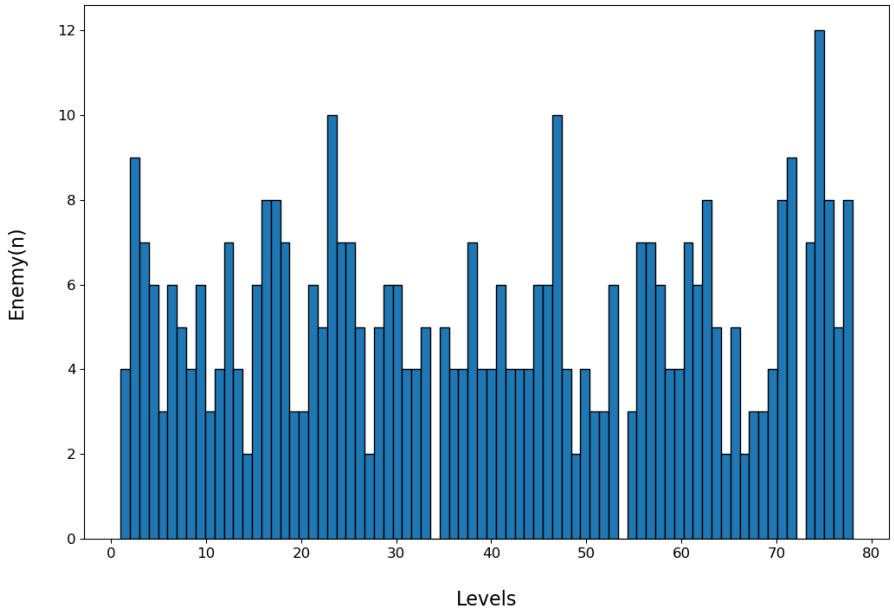
Seperti yang sudah dijelaskan pada Sub-bab 3.3.1, dimana pada bagian ini membahas tentang pendistribusian level musuh untuk permaian dengan genre RPG. Berikut adalah hasil dari proses distribusi level yang diperoleh dari perhitungan pada persamaan 3.11, 3.12, 3.13, 3.14 dan persamaan 3.15. Setelah melalui tahapan tersebut dan dengan variabel masukan pada Tabel 3.2 maka level yang dihasilkan akan seperti pada Tabel 4.12.

Tabel 4.12: Hasil level yang dibuat untuk musuh.

No.	Name	Levels
1	Enemy 1	1
2	Enemy 2	1
3	Enemy 3	1
4	Enemy 4	1
5	Enemy 5	2
6	Enemy 6	2
7	Enemy 7	2
8	Enemy 8	2
9	Enemy 9	2
10	Enemy 10	2
11	Enemy 11	2

12	Enemy 12	2
13	Enemy 13	2
14	Enemy 14	3
15	Enemy 15	3
...	...	...
400	**Enemy 400**	**78**

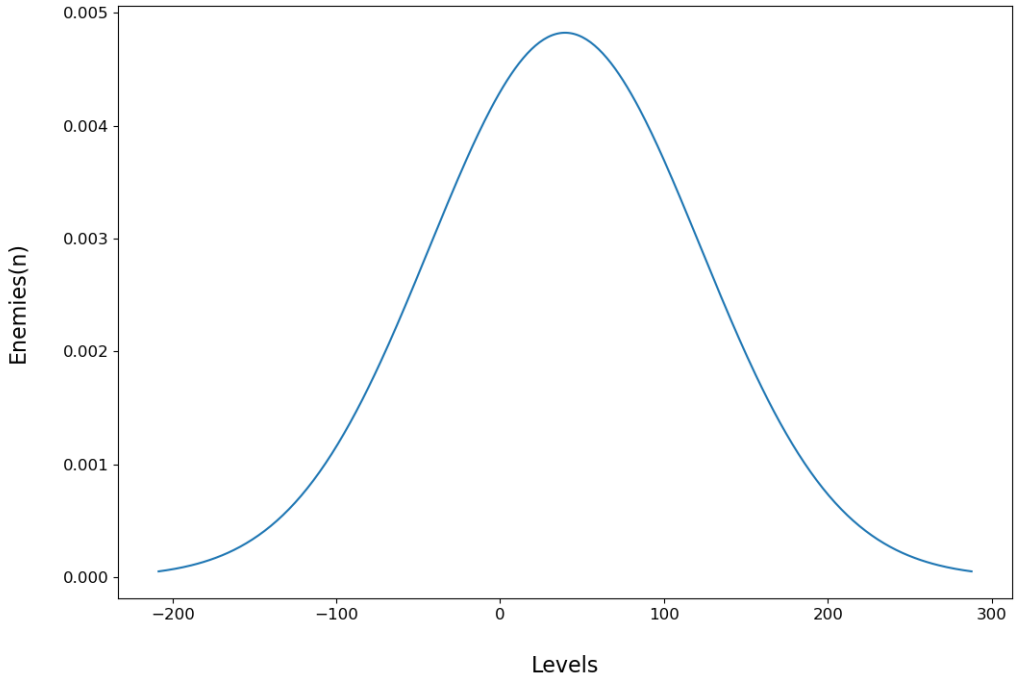
Pada Tabel 4.12 adalah sebagian data dari level yang dihasilkan oleh program, untuk hasil lebih lengkapnya bisa dilihat pada Tabel 5.16 sampai Tabel 5.30 dalam BAB LAMPIRAN pada Tabel 5.16 sampai dengan Tabel 5.30 di kolom *Levels*. Kemudian persebaran level yang dihasilkan tersebut bila divisualisasikan maka hasilnya akan seperti yang ditujukkkan pada Gambar 4.10. Grafik atau histogram yang ditunjukan pada Gambar 4.10 tersebut sangatlah tidak merata, hal tersebut dikarenakan proses penentuan level yang ditentukan secara acak.



Gambar 4.10: Distribusi Level Musuh.

Di lanjutkan dengan validasi dari keseimbangan persebaran level musuh, hal ini dilakukan dengan menggunakkan persamaan melalui beberapa langkah

seperti yang ditunjukan oleh persamaan 3.16, 3.17, 3.18 dan persamaan 3.19. Konsep tersebut beracuan pada Sub-bab 2.3.4 tentang *Gaussian Naive bayes*, dengan harapan apakah setiap data yang dihasilkan sebelumnya sudah terdistribusi dengan normal atau tidak. Jika memang benar data hasil program ini valid maka ketika musuh bertambah maka pola dari level dan jumlah musuh masih akan membentuk pola distribusi normal seperti pada Gambar 4.11.



Gambar 4.11: Distribusi level musuh dalam bentuk distribusi normal.

Pada Gambar 4.11 jika dilihat persebaran datanya dari kiri ke kanan menggambarkan tingkat kesulitan musuh berdasarkan level, jumlah musuh pada sisi kiri berjumlah sedikit semakin ke kanan jumlahnya meningkat dan semakin ke kanan jumlah musuh kembali menjadi sedikit jumlahnya. Hal tersebut menggambarkan tingkat keseimbangan dari musuh yang dibuat, yang mana jumlah musuh dengan level menengah berjumlah paling banyak dan jumlah musuh yang sangat sulit dan sangat mudah dikalahkan berjummlah sedikit. Tujuan utama dari kondisi tersebut adalah terjadinya keseimbangan saat terjadinya pertarungan antara pemain dan musuh.

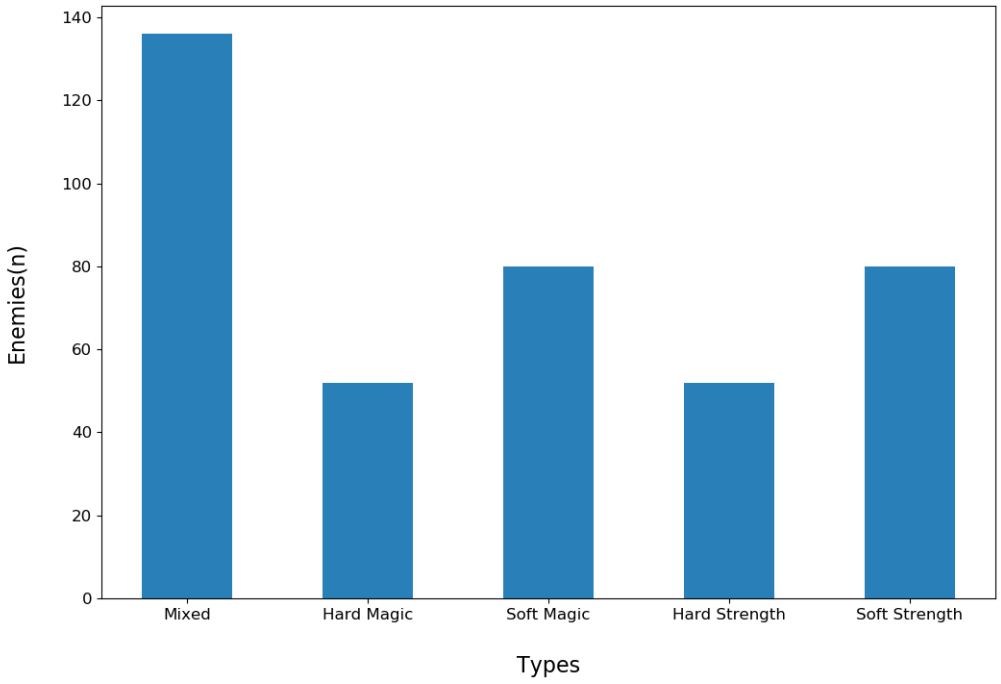
4.3.2 Hasil Distibusi Tipe Musuh

Seperti yang sudah dijelaskan pada Sub-bab 3.3.2, dimana pada bagian ini membahas tentang pendistribusian tipe musuh untuk permaian dengan genre *turn-based* RPG. Berikut adalah hasil dari proses distribusi tipe musuh yang diperoleh dari perhitungan pada persamaan 3.20, 3.21, 3.22, 3.23, 3.24 dan persamaan 3.25. Setelah melalui tahapan tersebut dengan variabel masukan pada Tabel 3.2 maka hasil distribusi dan penjelasan level yang dilakukan oleh beberapa persamaan tersebut akan menjadi seperti Tabel 4.13.

Tabel 4.13: Hasil level yang dibuat untuk musuh.

No.	Name	Levels	Type
1	Enemy 1	1	0
2	Enemy 2	1	0
3	Enemy 3	1	2
4	Enemy 4	1	1
5	Enemy 5	2	2
6	Enemy 6	2	1
7	Enemy 7	2	2
8	Enemy 8	2	0
9	Enemy 9	2	2
10	Enemy 10	2	2
11	Enemy 11	2	4
12	Enemy 12	2	0
13	Enemy 13	2	0
14	Enemy 14	3	4
15	Enemy 15	3	4
16	Enemy 16	3	2
...	...	...	...
400	**Enemy 400**	**78**	**3**

Dari data yang tedapat pada Tabel 4.13 hanya sebagian data saja, lebih lengkapnya dapat dilihat langsung pada LAMPIRAN dalam Tabel 5.16 sampai dengan Tabel 5.30 di kolom *Type*. Pada Tabel 4.13 dan tabel lain yang memuat tentang *Type* yang berisi data ET_i atau jenis musuh yang diisi dengan angka 0 sampai dengan 4, maksud dari angka tersebut adalah untuk mewakili indeks dari variabel *Enemy Type* pada Tabel 3.2. Secara berurutan dari 0 sampai dengan 4 adalah *Mixed*, *Hard Magic*, *Soft Magic*, *Hard Strength*, dan *Soft Magic*. Seluruh data tersebut jika divisualisasikan maka hasilnya seperti Gambar 4.12 sesuai yang ditargetkan oleh variabel *Distribute Percentage* pada Tabel 3.2 atau variabel DP pada persamaan 3.20.



Gambar 4.12: Distribusi tipe musuh.

Kemudian pada Gambar 4.12 adalah hasil dari proses pembuatan dan pengelompokan musuh berdasarkan tipe yang sudah sesuai dengan variabel masukan pada Tabel 3.2. Dengan musuh bertipe *Mixed* berjumlah yang paling banyak, diikuti *soft magic* dan *soft strength* dengan harapan bahwa ini adalah musuh yang memiliki karakter *magic* dan *strength* namun masih mudah dikalahkan, dilanjutkan dengan yang paling sedikit adalah *hard magic* dan

hard strength dengan harapan menjadi musuh berkarakter strength dan magic yang sulit dikalahkan.

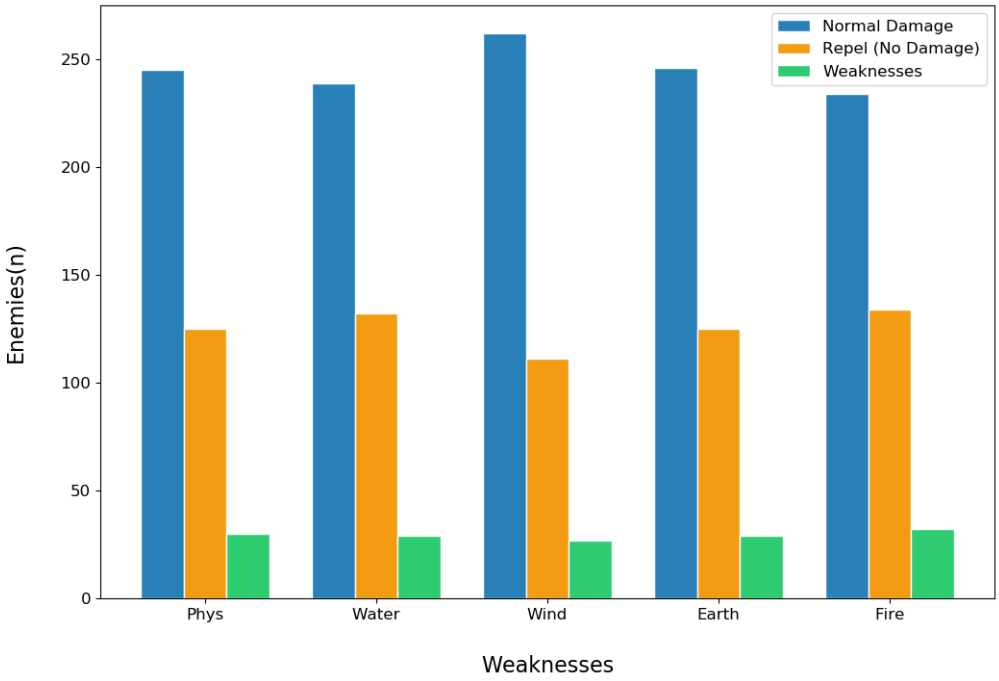
4.3.3 Hasil Distribusi Element dan Kelemahan Musuh

Seperti yang sudah dijelaskan pada Sub-bab 3.3.3, dimana pada bagian ini membahas tentang pendistribusian elemen dan kelemahan musuh untuk permaian dengan genre *turn-based* RPG. Berikut adalah hasil dari proses distribusi tipe musuh yang diperoleh dari perhitungan pada persamaan 3.26, 3.27, 3.28, 3.29, dan 3.30. Setelah melalui tahapan tersebut dengan variabel masukan pada Tabel 3.2 maka hasil persebaran kelemahan dan kekebalan musuh ditunjukkan pada Tabel 4.14 dan selengkapnya dapat dilihat pada LAMPIRAN dalam Tabel 5.16 sampai dengan Tabel 5.30.

Tabel 4.14: Hasil level yang dibuat untuk musuh.

No.	Name	Phys	Water	Wind	Earth	Fire
1	Enemy 1	1	1	0	1	0
2	Enemy 2	0	0	0	0	0
3	Enemy 3	0	1	0	1	1
4	Enemy 4	1	0	1	0	0
5	Enemy 5	0	1	0	0	1
6	Enemy 6	1	1	0	1	0
7	Enemy 7	1	0	0	0	0
8	Enemy 8	0	0	0	0	1
9	Enemy 9	1	1	0	0	0
10	Enemy 10	0	0	0	0	0
11	Enemy 11	1	0	0	1	0
12	Enemy 12	0	0	0	0	1
13	Enemy 13	0	0	2	1	0
...	...	...	...	...	...	...
400	**Enemy 400**	**0**	**0**	**2**	**1**	**0**

Jika melihat Tabel 4.14 dengan melihat pada kolom *Phys*, *Water*, *Wind*, *Earth*, dan *Fire*, isi dari kolom tersebut berupa angka nol sampai dengan dua adalah hasil dari pembuatan atribut *gameplay* kelemahan musuh pada variabel $DmgEN_M = \{DmgNu_0, DmgNu_1, DmgNu_2, ..., DmgNu_M\}$ jika jumlah kelemahan musuh lebih dari tiga atau yang dicontohkan pada Tabel 3.2 dalam variabel *List Damage*. Hasil dari proses yang sebelumnya dijelaskan pada Sub-bab ini menghasilkan persebaran kelemahan dan kekebalan musuh yang ditunjukkan pada Gambar 4.13 dibawah ini.



Gambar 4.13: Distribusi kelemahan musuh.

Pada Gambar 4.13 dapat dilihat distribusi musuh dengan kelemahan dan kekebalannya. Terlihat pada kondisi tersebut jumlah musuh dengan kondisi *Normal* memiliki nilai paling tunggi pada setiap kelemahan, hal tersebut dapat diartikan bahwa sebagian besar elemen dari musuh dapat memperoleh *damage* secara normal jika diserang. Kemudian jumlah musuh pada kondisi *Repel* berjumlah terbanyak kedua, hal tersebut dapat diartikan musuh memmiliki jumlah kekebalan terhadap serangan sejumlah angka pada setiap elemen pada Gambar 4.13. Begitu juga dengan kondisi *Weaknesses* dengan jumlah paling

sedikit, hal tersebut bertujuan agar menjadikan pertarungan antara pemain dan musuh menjadi tidak mudah dimenangkan oleh pemain. Tidak hanya dengan efek *Weaknesses*, efek *Repel* juga sangat berpengaruh dalam menjadikan permainan seperti layaknya sebuah *puzzle* atau teka-teki.

Sedangkan pada persamaan 3.29 adalah penjelasan tentang proses munculnya setiap $DmgNu$ pada satu karakter musuh, sehingga variabel tersebut berubah menjadi $DmgNu_i$. Seperti penjelasan sebelumnya bahwa pada sebuah karakter musuh dapat memiliki lebih dari satu kelemahan atau kekebalan $DmgNu$ terhadap $DmgNa$ yang diset secara acak seperti yang sudah dijelaskan pada bagian sebelumnya.

4.3.4 Hasil Distribusi HP, MP, dan Atribut Gameplay pada Musuh

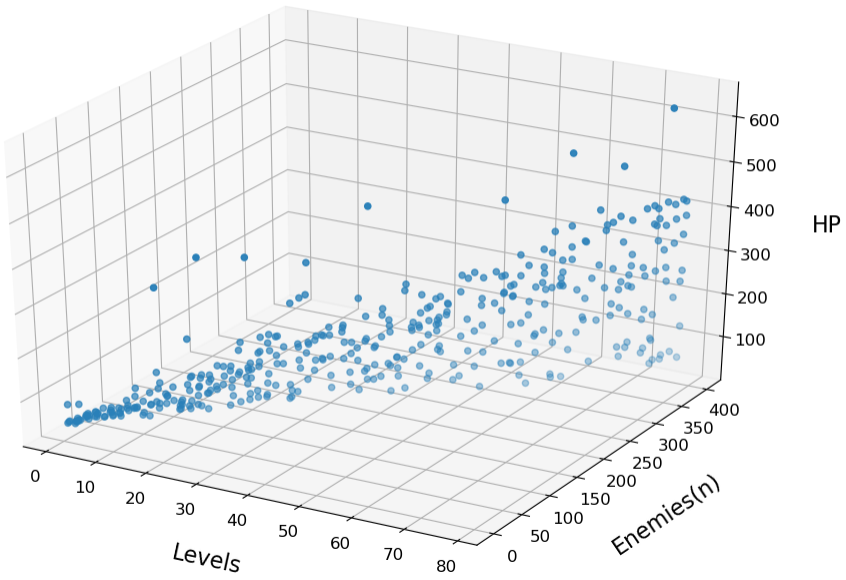
pada Sub-bab 3.3.4, dimana pada bagian ini membahas tentang pendistribusian HP dan MP musuh untuk permaian dengan genre *turn-based* RPG. Berikut adalah hasil dari proses distribusi tipe musuh yang diperoleh dari perhitungan pada persamaan 3.31 sampai dengan persamaan 3.46, yang mana pada persamaan 3.31 sampai dengan persamaan 3.36 adalah penenentuan tipe musuh. Setelah melalui tahapan tersebut dengan variabel masukan pada Tabel 3.2 maka hasil persebaran kelemahan dan kekebalan musuh ditunjukkan pada Tabel 4.14.

Tabel 4.15: Distribusi atribut *gameplay* HP dan MP musuh.

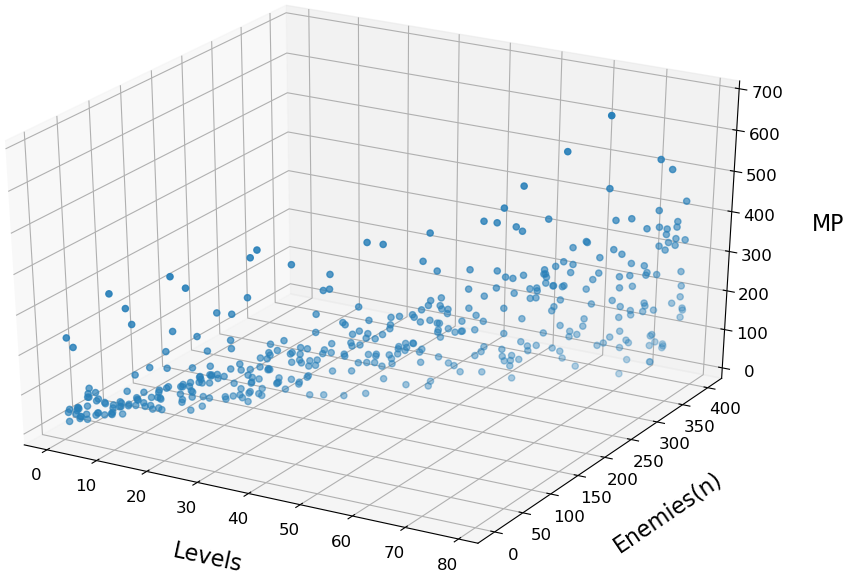
No.	Name	Levels	HP	MP
1	Enemy 1	1	42	45
2	Enemy 2	1	84	21
3	Enemy 3	1	43	51
4	Enemy 4	1	44	231
5	Enemy 5	2	40	40
6	Enemy 6	2	44	208

7	Enemy 7	2	44	47
8	Enemy 8	2	42	53
9	Enemy 9	2	49	53
10	Enemy 10	2	42	49
11	Enemy 11	2	44	23
12	Enemy 12	2	79	27
13	Enemy 13	2	41	26
14	Enemy 14	3	50	34
15	Enemy 15	3	54	20
...	...	...	...	...
400	**Enemy 400**	**78**	**389**	**161**

Pada Tabel 4.15 hanya sebagian data saja yang ditampilkan, untuk selengkapnya dapat dilihat pada LAMPIRAN dalam Tabel 5.16 sampai dengan Tabel 5.30 pada kolom HP dan MP. Selanjutnya adalah penggambaran persebaran HP dan MP yang masing-masing digambarkan pada Gambar 4.14 dan Gambar 4.15.



Gambar 4.14: Distribusi atribut *gameplay* HP musuh.



Gambar 4.15: Distribusi atribut *gameplay* MP musuh.

Masih pada Sub-bab 3.3.4, dimana pada bagian ini membahas tentang pendistribusian atribut *gameplay* musuh untuk permaian dengan genre *turn-based* RPG. Berikut adalah hasil dari proses distribusi tipe musuh yang diperoleh dari perhitungan mulai dari persamaan 3.31 secara berurutan sampai dengan persamaan 3.36 yang digunakan untuk penentuan tipe musuh yang ingin dipakai.

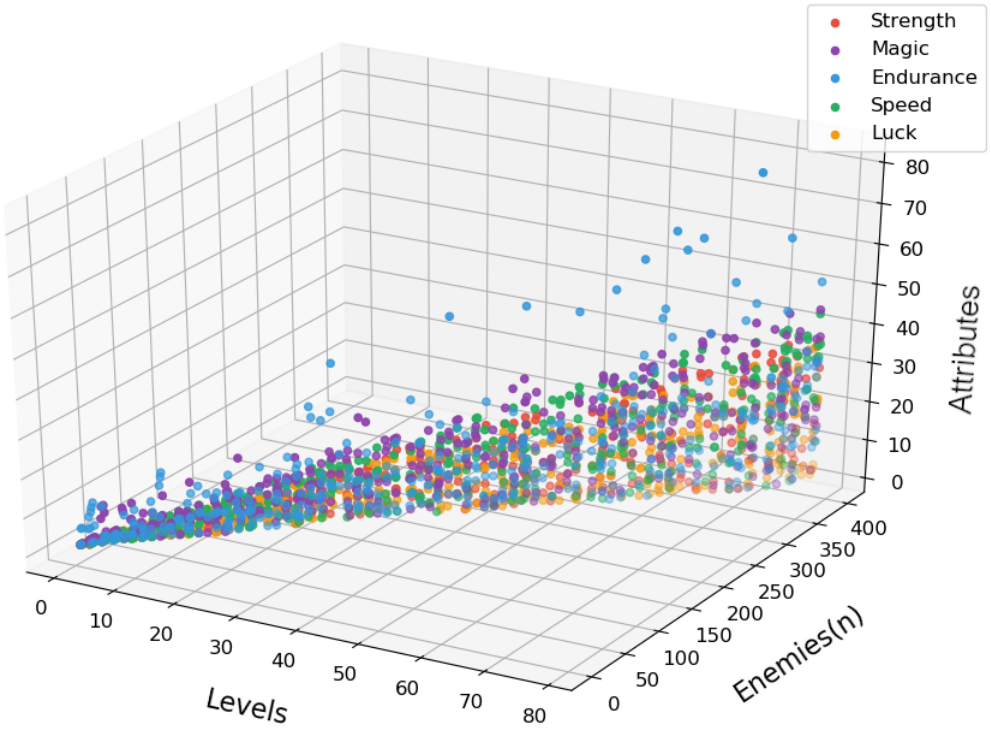
Sedangkan proses kalkulasi untuk menentukan atribut *gameplay* musuh dimulai dari persamaan 3.47 sampai dengan persamaan 3.50. Setelah melalui tahapan tersebut dengan variabel masukan pada Tabel 3.2 maka hasil persebaran atribut *gameplay* musuh dengan komposisi *Strength*, *Magic*, *Endurance*, *Speed*, dan *Luck* ditunjukkan pada Tabel 4.16.

Tabel 4.16: Distribusi atribut *gameplay* musuh.

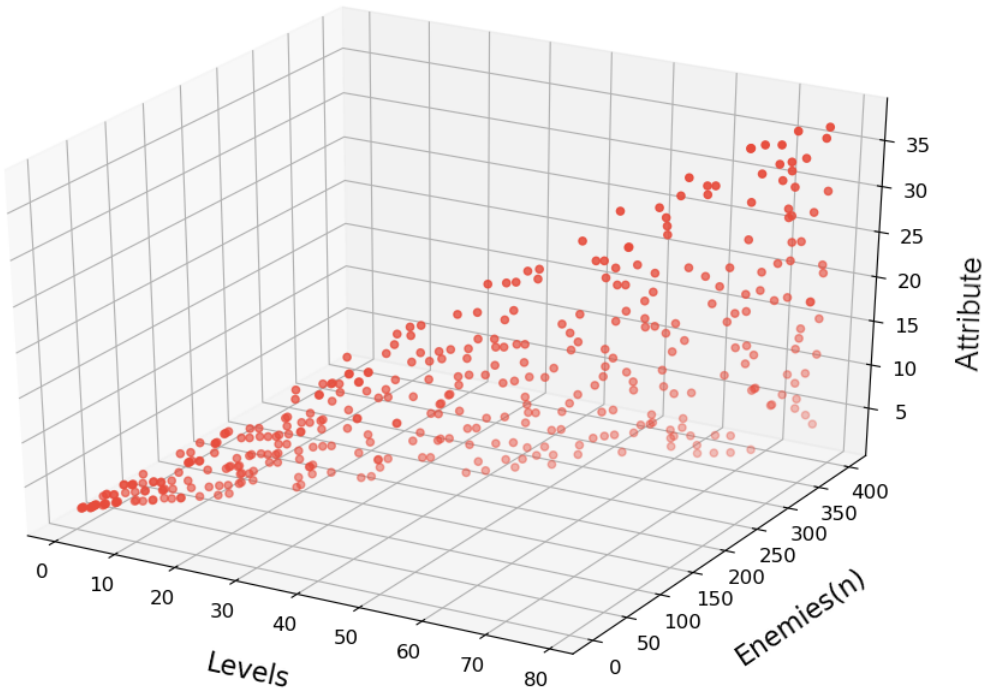
No.	Name	Lv.	Str	Mag	Endr	Spd	Luck
1	Enemy 1	1	2	2	2	2	2
2	Enemy 2	1	2	2	2	2	2
3	Enemy 3	1	2	2	6	2	2

4	Enemy 4	1	2	2	6	2	2
5	Enemy 5	2	2	3	6	3	2
6	Enemy 6	2	2	3	9	2	2
7	Enemy 7	2	2	3	6	2	2
8	Enemy 8	2	2	3	2	2	2
9	Enemy 9	2	2	3	10	3	2
10	Enemy 10	2	2	3	11	3	2
11	Enemy 11	2	2	3	2	2	2
12	Enemy 12	2	2	3	12	3	2
13	Enemy 13	2	2	3	6	2	2
14	Enemy 14	3	2	8	2	2	2
15	Enemy 15	3	3	11	2	3	2
16	Enemy 16	3	2	2	11	2	2
...	...	...	...	...	...	...	...
400	**Enemy 400**	**78**	**30**	**15**	**11**	**36**	**22**

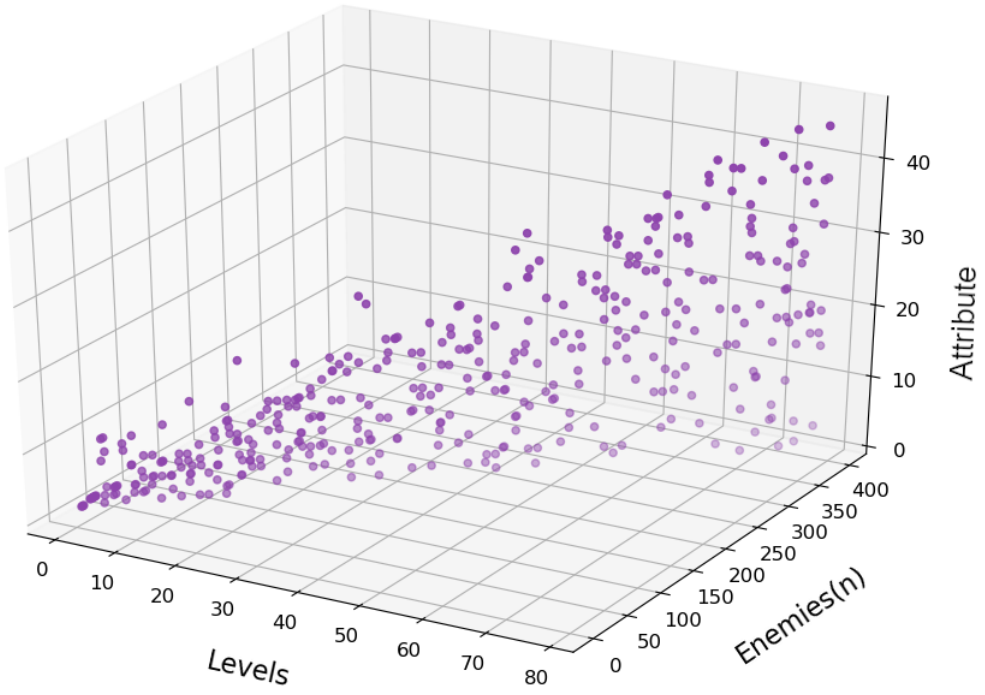
Selanjutnya adalah penggambaran persebaran atribut *gameplay* yang disebutkan pada Tabel 4.16 yang digambarkan pada Gambar 4.16. Pada Gambar 4.17, 4.18, 4.19, 4.20 dan Gambar 4.21 adalah pecahan dari Gambar 4.16 secara berurutan diantaranya adalah *Strength*, *Magic*, *Endurance*, *Speed* dan *Luck*. Data atribut *gameplay* selengkapnya dapat dilihat pada LAMPIRAN dalam Tabel 5.16 sampai dengan Tabel 5.30 pada kolom *Strength*, *Magic*, *Endurance*, *Speed*, *Luck* dan atribut *gameplay* yang lain jika dilakukan penambahan atribut *gameplay* pada program.



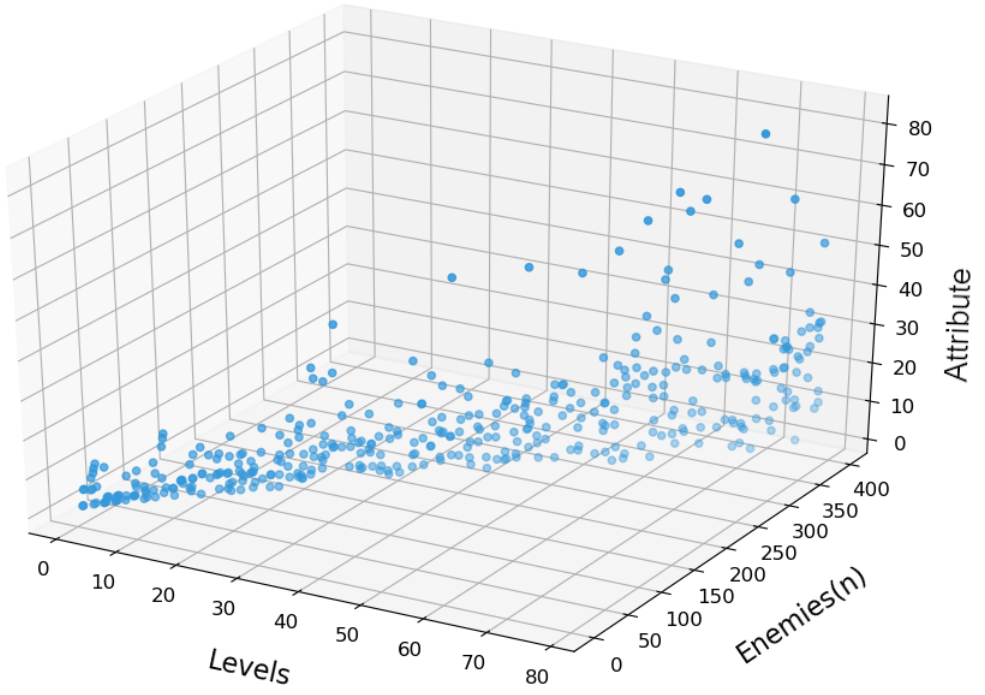
Gambar 4.16: Distribusi atribut *gameplay* musuh secara keseluruhan.



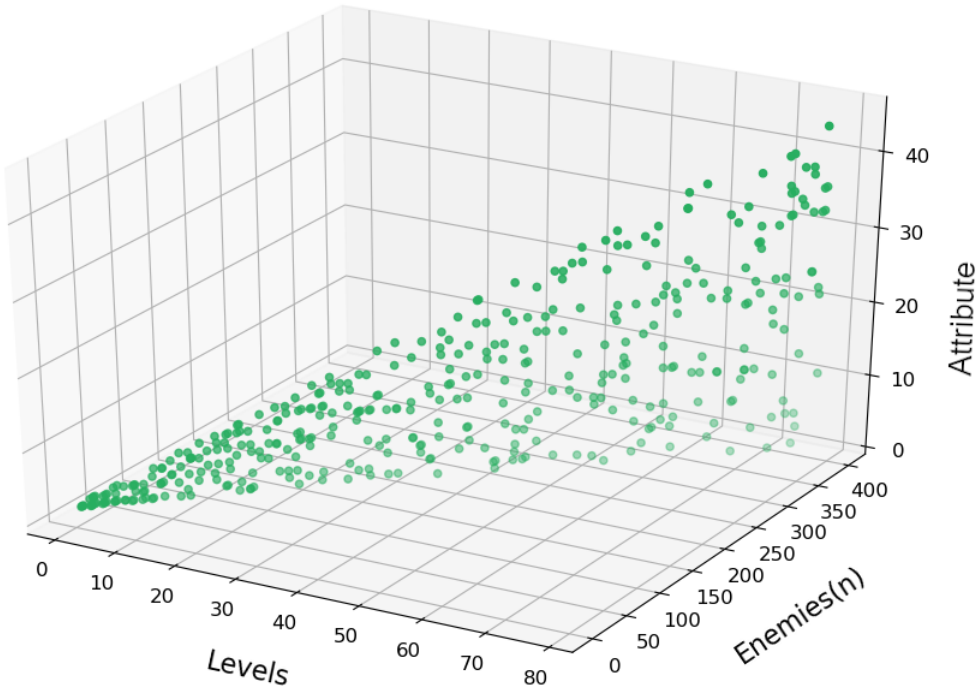
Gambar 4.17: Distribusi *Strength* musuh.



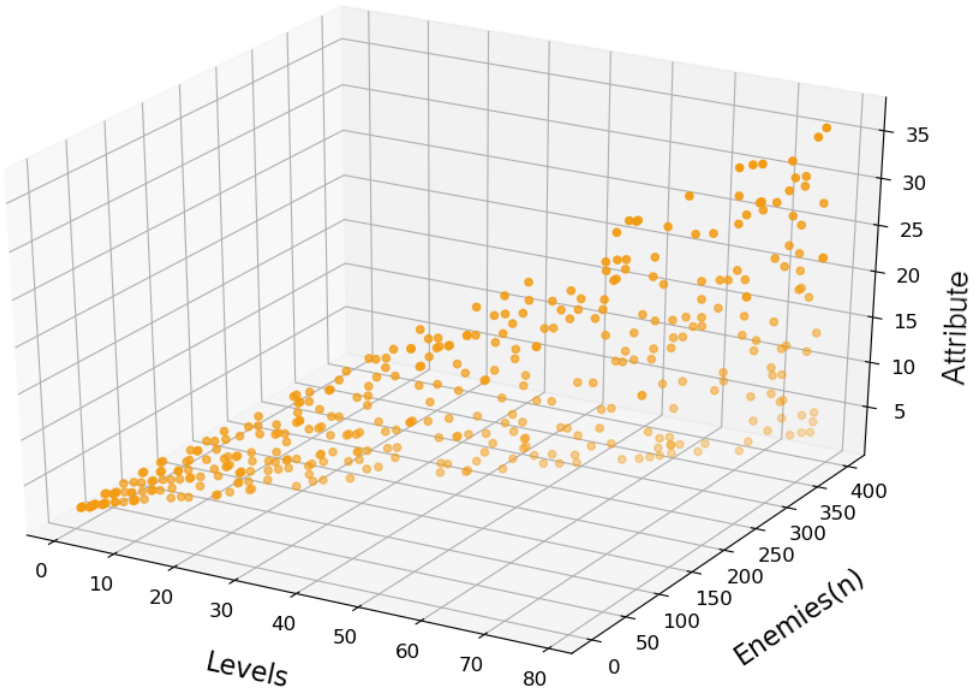
Gambar 4.18: Distribusi *Magic* musuh.



Gambar 4.19: Distribusi *Endurance* musuh.



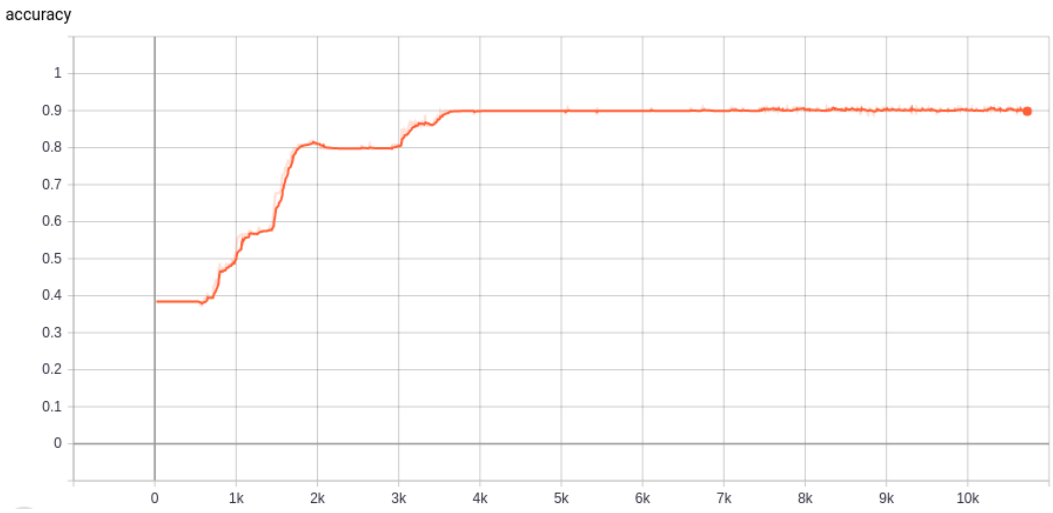
Gambar 4.20: Distribusi *Speed* musuh.



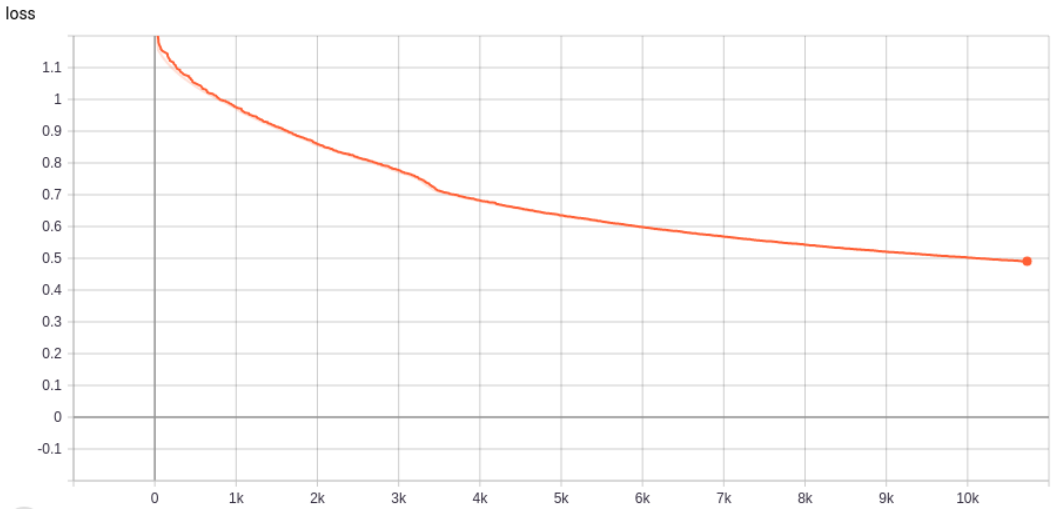
Gambar 4.21: Distribusi *Luck* musuh.

4.4 Hasil Klasifikasi Karakter pada Permainan Dota 2

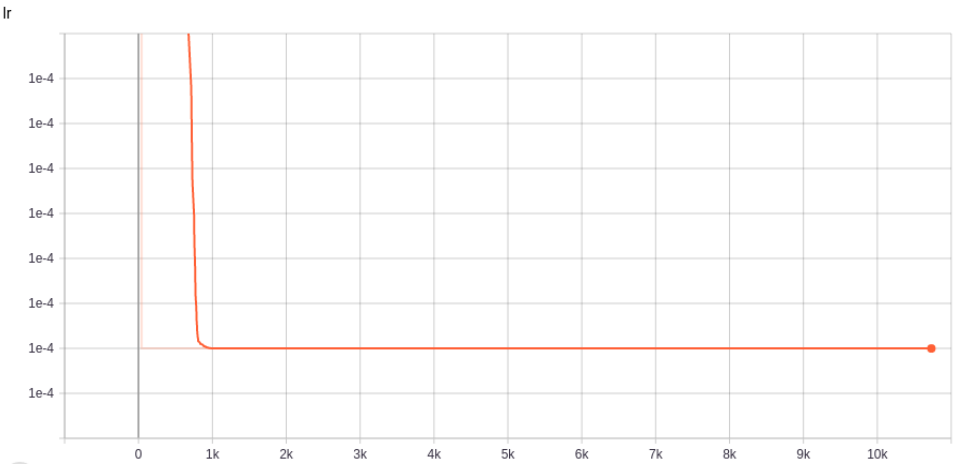
Pada Sub-bab ini akan membahas hasil dari Sub-bab 3.4, bila dijelaskan melalui diagram blok pada Gambar 3.14 maka bagian yang akan dibahas pada Sub-bab ini adalah proses saat klasifikasi. Grafik hasil *training* dapat dilihat pada Gambar 4.22, 4.23, 4.24, 4.25, dan Gambar 4.26. Masing-masing Gambar tersebut secara berurutan adalah akurasi saat *training*, *loss* saat *training*, *learning rate* saat *training*, akurasi saat evaluasi dan *loss* saat evaluasi.



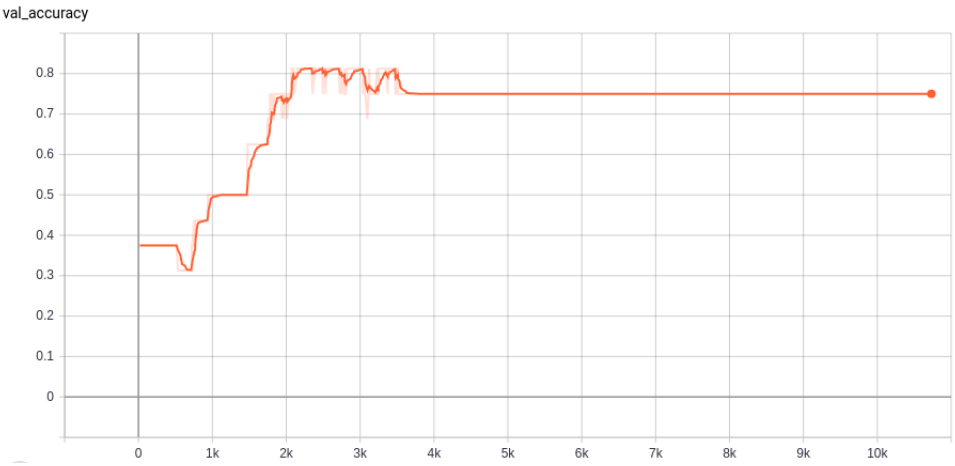
Gambar 4.22: Akurasi saat *training* pada *hero* Dota 2.



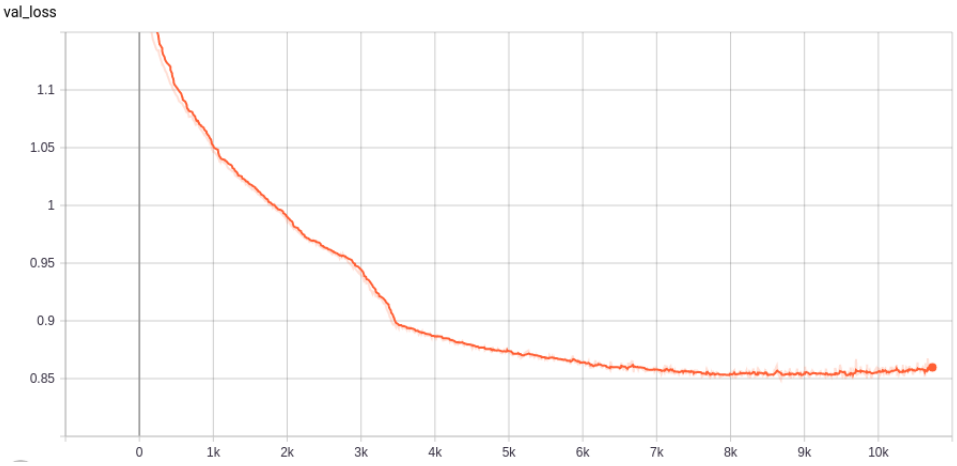
Gambar 4.23: *Loss* saat *training* pada *hero* Dota 2.



Gambar 4.24: *Learning rate* saat *training* pada *hero* Dota 2.



Gambar 4.25: Akurasi pada saat proses evaluasi pada *hero* Dota 2.



Gambar 4.26: *Loss* saat proses evaluasi pada *hero* Dota 2.

Seperti penjelasan tentang proses *training* pada Sub-bab 3.4.4, *training* berlangsung selama 10.730 *epoch*. Akurasi saat *training* mampu mencapai 0.9 dan mengalami *loss* sebesar 0.55, kemudian akurasi saat evaluasi mampu mencapai 0.75 dan mengalami *loss* saat evaluasi sebesar 0.86 . Hasil klasifikasi dari data atribut *gameplay* karakter atau *hero* Dota 2 yang dijadikan sebagai data *testing* dapat dilihat pada Tabel 5.33 yang kemudian digunakan pada proses evalusi dan *testing* dengan hasil yang ditampilkan pada Tabel 4.17.

Tabel 4.17: Hasil proses *testing* dengan data *testing* pada *hero* Dota 2.

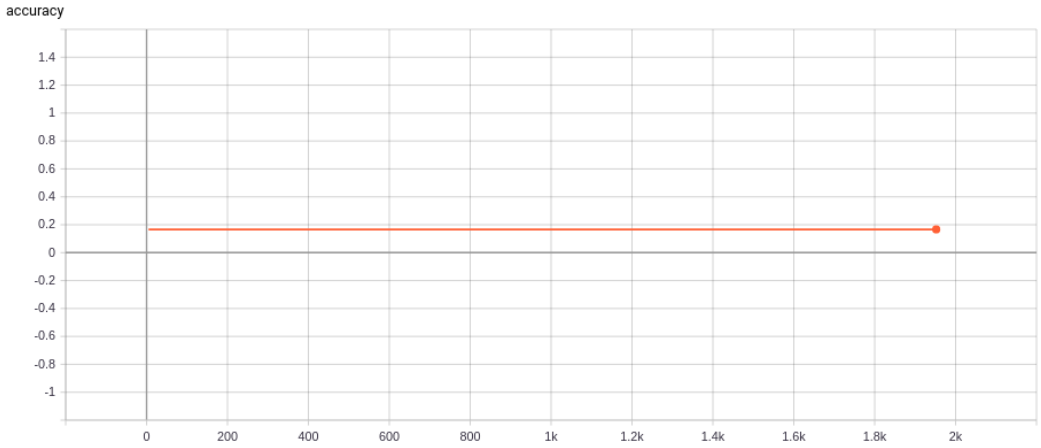
Hero Name	Strength	Agility	Intelligent
ArcWarden	0.053558	0.206808	0.739634
Jakiro	0.205992	0.241550	0.552458
WinterWyvern	0.205992	0.241552	0.552456
Bloodseeker	0.161604	0.760779	0.077617
Phoenix	0.053562	0.206812	0.739627
Io	0.053558	0.206813	0.739629
Leshrac	0.053559	0.206871	0.739570
OutworldDevourer	0.205848	0.245712	0.548440
Brewmaster	0.624197	0.189504	0.186300
Tusk	0.377984	0.510914	0.111102
Weaver	0.098899	0.560487	0.340614
Undying	0.645770	0.127254	0.226976
Alchemist	0.643663	0.126769	0.229569
Phantom Lancer	0.154832	0.768835	0.076333
CrystalMaiden	0.053558	0.206809	0.739633
Invoker	0.053558	0.206809	0.739633

Cara membaca hasil klasifikasi pada Tabel 4.17 sebelumnya sudah dijelaskan pada Sub-bab 3.4.3 dan Sub-bab 3.4.4. Hasil ditampilkan pada Tabel 4.17 terdapat 4 *hero* yang mengalami salah klasifikasi yang kemudian diberi warna merah, dengan beracuan pada kolom *type* dalam Tabel 5.33 yang ter-

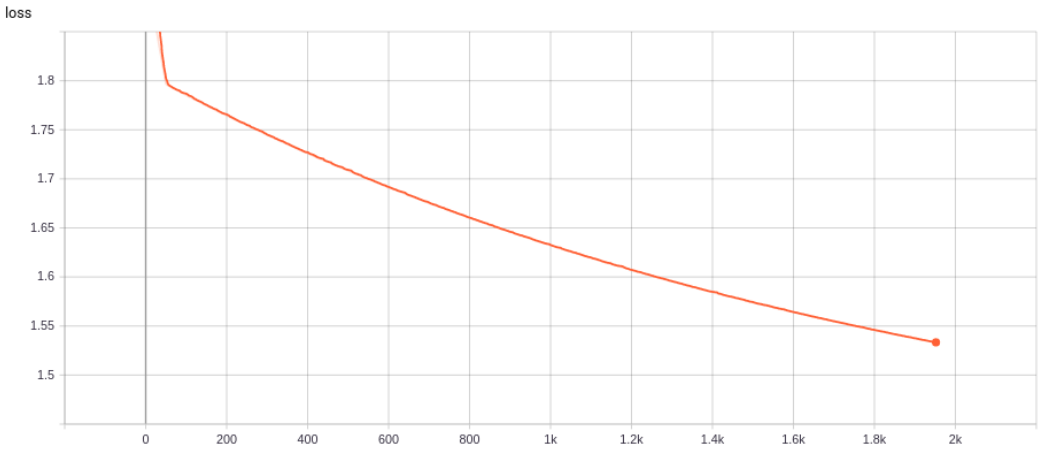
lampir pada LAMPIRAN. Maka dari 16 *hero* terdapat 4 hero yang mengalami salah klasifikasi, jadi presentasenya 64% *hero* berhasil terklasifikasi.

4.5 Hasil Klasifikasi Karakter Pemain

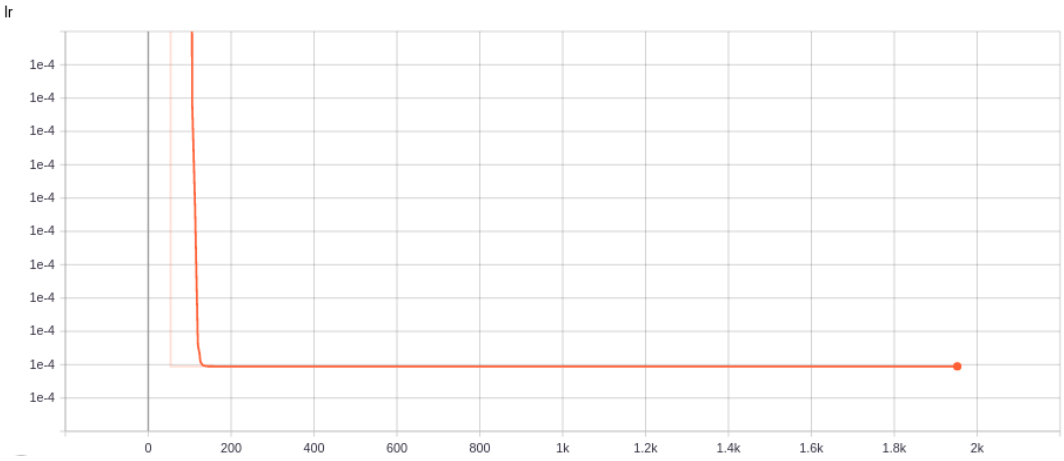
Pada Sub-bab ini akan membahas hasil dari Sub-bab 3.5, bila dijelaskan melalui diagram blok pada Gambar 3.20 maka bagian yang akan dibahas pada Sub-bab ini adalah klasifikasi. Grafik hasil *training* dapat dilihat pada Gambar 4.27, 4.28, 4.29, 4.30, dan Gambar 4.31. Masing-masing dari Gambar tersebut secara berurutan adalah akurasi saat *training*, *loss* saat *training*, *learning rate* saat *training*, akurasi saat evaluasi dan *loss* saat evaluasi.



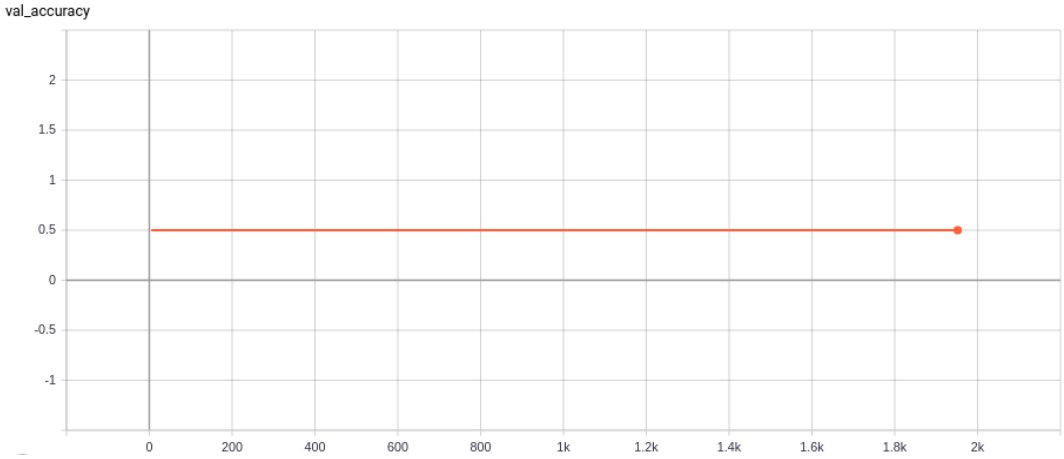
Gambar 4.27: Akurasi saat *training* pada karakter pemain.



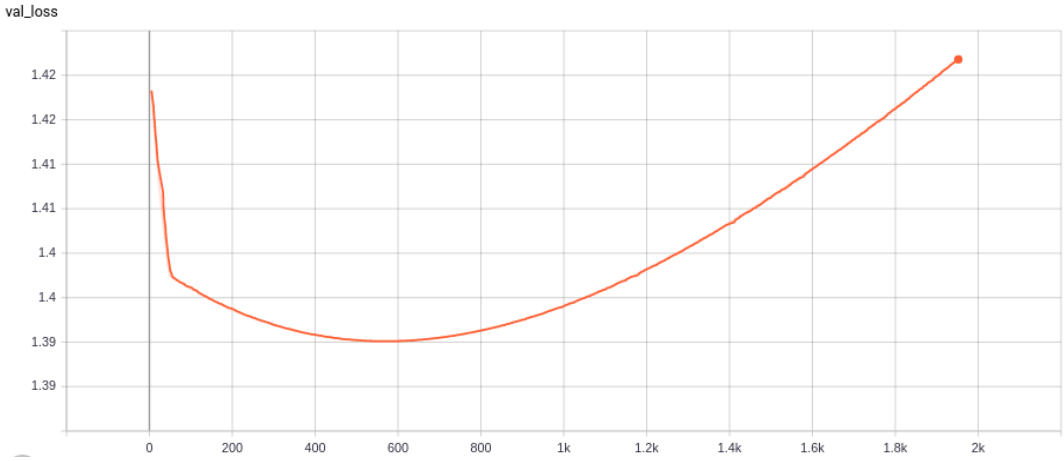
Gambar 4.28: *Loss* saat *training* pada karakter pemain.



Gambar 4.29: *Learning rate* saat *training* pada karakter pemain.



Gambar 4.30: Akurasi pada saat proses evaluasi pada karakter pemain.



Gambar 4.31: *Loss* saat proses evaluasi pada karakter pemain.

Seperti penjelasan tentang proses *training* pada Sub-bab 3.5.4, *training* berlangsung selama 425 *epoch*. Akurasi saat *training* tetap stabil pada 0.1667 dan mengalami *loss* sebesar 1.533, kemudian akurasi saat evaluasi tetap stabil pada 0.5 dan mengalami *loss* saat evaluasisebesar 1.39. Hasil klasifikasi dari data atribut *gameplay* dari karakter pemain yang dijadikan sebagai data *testing* dapat dilihat pada Tabel 5.35 yang kemudian digunakan pada proses evaluasi dan *testing* dengan hasil yang ditampilkan pada Tabel 4.18.

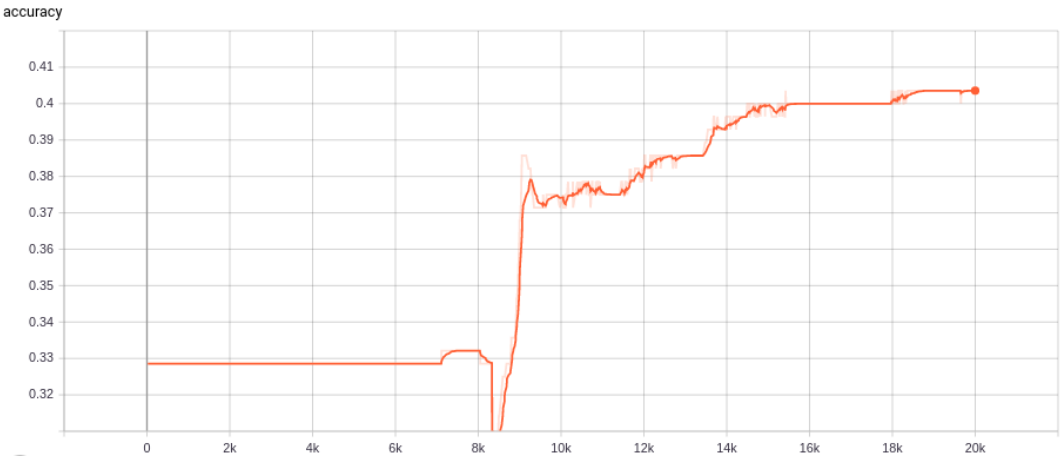
Tabel 4.18: Hasil proses *testing* dengan data *testing* pada karakter pemain.

Type	**Knight**	**Priest**	**Assassin**	**Magician**
Assassin 2	0.122532	0.190597	0.579998	0.106873
Magician 2	0.122532	0.190597	0.579998	0.106873

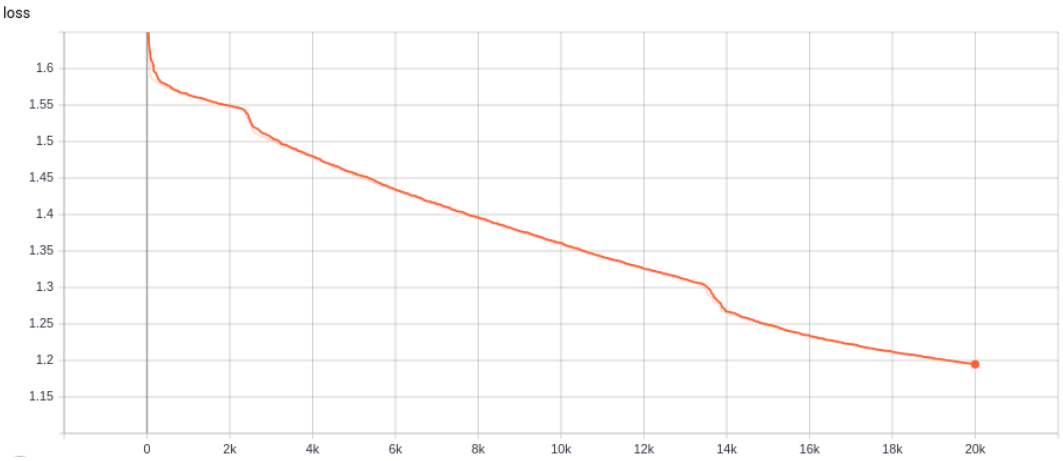
Cara membaca hasil klasifikasi pada Tabel 4.18 sebelumnya sudah dijelaskan pada Sub-bab 3.5.3 dan Sub-bab 3.5.4. Hasil ditampilkan pada Tabel 4.18 terdapat satu karakter yang mengalami salah klasifikasi yang kemudian diberi warna merah, dengan beracuan pada Tabel 5.35 pada bagian kolom *type*. Maka dari 2 karakter tersebut, terdapat 1 karakter yang mengalami salah klasifikasi atau dianggap tidak sesuai dengan tipenya, bila dinyatakan dalam presentase menjadi 50% karakter pemain berhasil terklasifikasi.

4.6 Hasil Klasifikasi Karakter Musuh

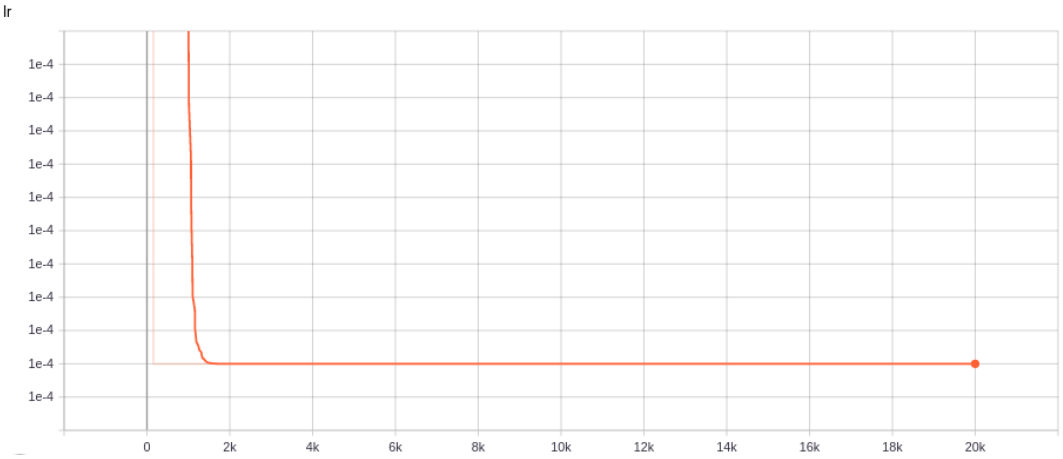
Pada Sub-bab ini akan membahas hasil dari Sub-bab 3.6, bila dijelaskan melalui diagram blok pada Gambar 3.22 maka bagian yang akan dibahas pada Sub-bab ini adalah klasifikasi. Grafik hasil *training* dapat dilihat pada Gambar 4.32, 4.33, 4.34, 4.35, dan Gambar 4.36. Masing-masing dari Gambar tersebut secara berurutan diantaranya adalah akurasi saat *training*, *loss* saat *training*, *learning rate* saat *training*, akurasi saat evaluasu dan *loss* saat evaluasi. Pada proses tersebut dijalankan dengan maksimum 20k *epoch*, dengan optimasi sama seperti yang sudah dilakukan dan dijelaskan sebelumnya pada Sub-bab 3.4.4 dan Sub-bab 3.4.5.



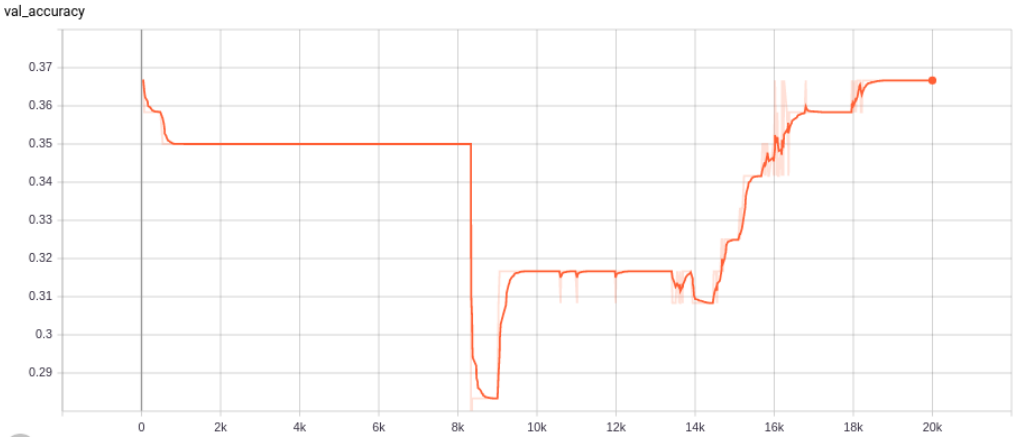
Gambar 4.32: Akurasi saat *training* pada karakter musuh (20k *epoch*).



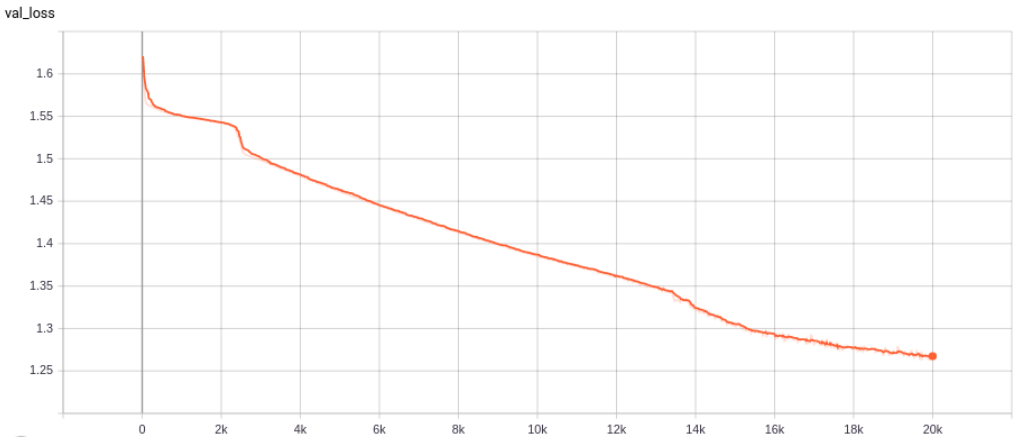
Gambar 4.33: *Loss* saat *training* pada karakter musuh (20k *epoch*).



Gambar 4.34: *Learning rate* saat *training* pada karakter musuh (20k *epoch*).

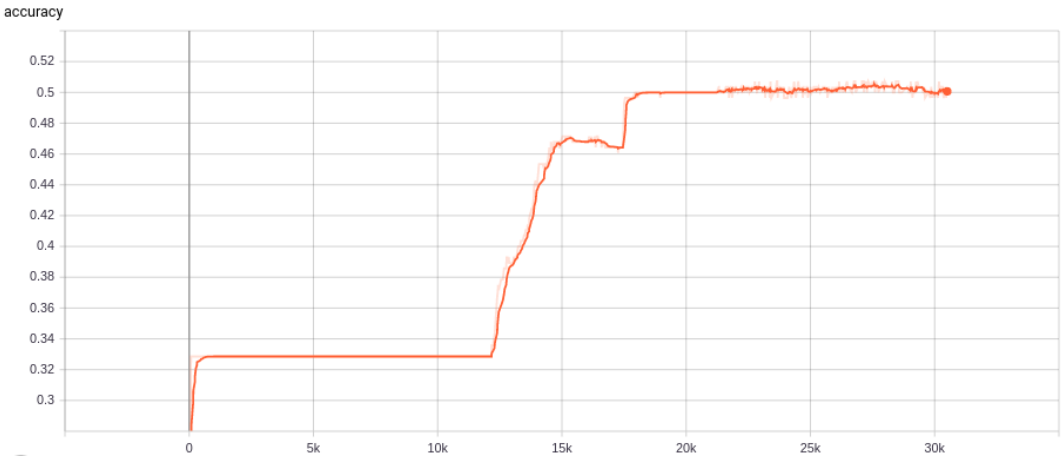


Gambar 4.35: Akurasi saat proses evaluasi pada karakter (20k *epoch*).

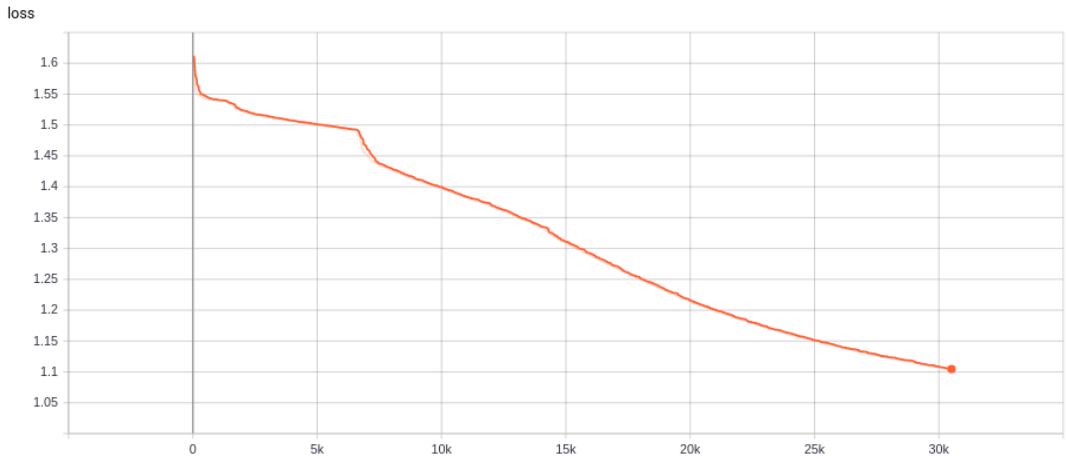


Gambar 4.36: *Loss* proses evaluasi pada karakter pemain (20k *epoch*).

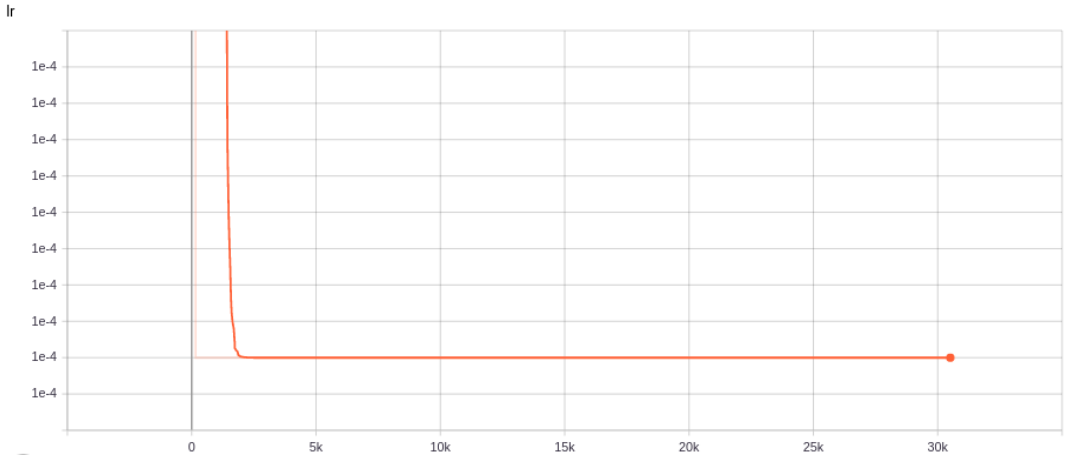
Seperti penjelasan tentang proses *training* pada Sub-bab 3.6.4, *training* berlangsung selama 20k *epoch*. Akurasi saat *training* mampu mencapai 0.4 dan mengalami *loss* sebesar 1.194, kemudian akurasi saat *training* mampu mencapai 0.3667 dan mengalami *loss* saat evaluasi sebesar 1.266. Melihat kondisi ini maka dilakukan peningkatan *epoch* menjadi 40k, hal tersebut bertujuan agar tercapainya kondisi optimum. Maksud dari kondisi tersebut adalah dicapainya nilai *loss* terendah saat dilakukannya evaluasi. Grafik hasil *training* dapat dilihat pada Gambar 4.37, 4.38, 4.39, 4.40, dan Gambar 4.41. Urutan dari gambar tersebut diantaranya adalah akurasi saat *training*, *loss* saat *training*, *learning rate* saat *training*, akurasi saat evaluasi dan *loss* saat evaluasi dengan 40k *epoch*.



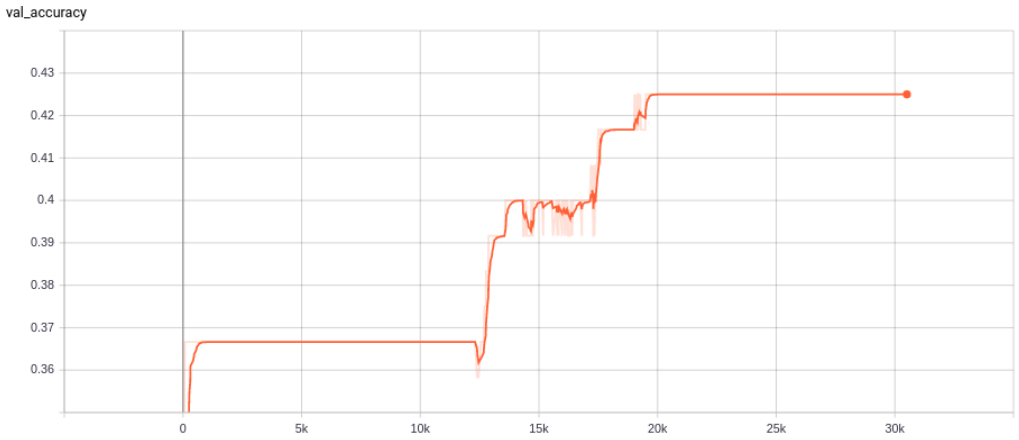
Gambar 4.37: Akurasi saat *training* pada karakter musuh (40k *epoch*).



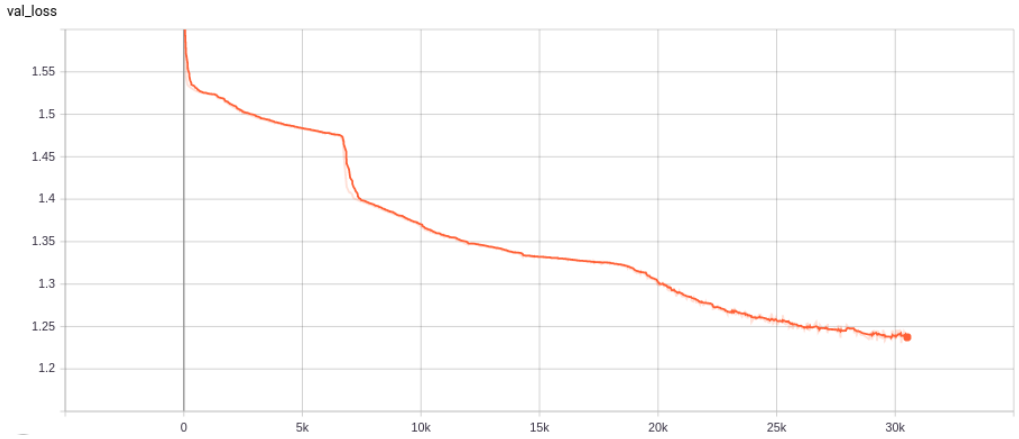
Gambar 4.38: *Loss* saat *training* pada karakter musuh (40k *epoch*).



Gambar 4.39: *Learning rate* saat *training* pada karakter musuh (40k *epoch*).



Gambar 4.40: Akurasi saat evaluasi pada karakter musuh (40k *epoch*).



Gambar 4.41: *Loss* saat evaluasi pada karakter musuh (40k *epoch*).

Seperti penjelasan tentang proses *training* pada Sub-bab 3.6.4, *training* berlangsung selama 30.510 *epoch* setelah dilakukan penambahan 20k *epoch*, yang semula hanya 20k *epoch* menjadi 40k *epoch*. Hal tersebut diasumsikan bahwa kondisi optimum dapat dicapai dengan *epoch* kurang dari 40k dan lebih dari 20k. Akurasi saat *training* mampu mencapai 0.5 dan mengalami *loss* sebesar 1.110, kemudian akurasi saat evaluasi mampu mencapai 0.425 dan mengalami *loss* saat evaluasi sebesar 1.245. Hasil klasifikasi dari data atribut *gameplay* karakter musuh yang dijadikan sebagai data evaluasi dapat dilihat pada Tabel 5.16 sampai dengan Tabel 5.30 untuk musuh dengan urutan ke 281 sampai 400, kemudian data tersebut digunakan pada proses evaluasi dan *testing* dengan hasil yang dapat dilihat pada Tabel 4.19.

Tabel 4.19: Hasil proses *testing* dengan data *testing* pada karakter musuh.

Name	Type	Type 0	Type 1	Type 2	Type 3	Type 4
Enemy 281	2	0.378368	0.168215	0.316104	0.058081	0.079233
Enemy 282	0	0.527585	0.010544	0.046145	0.122267	0.293459
Enemy 283	2	0.371143	0.13154	0.276541	0.092394	0.128382
Enemy 284	1	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 285	4	0.524063	0.010682	0.046597	0.123526	0.295132
Enemy 286	0	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 287	1	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 288	0	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 289	2	0.377502	0.175532	0.322636	0.0527	0.07163
Enemy 290	0	0.37244	0.191845	0.329262	0.045204	0.06125
Enemy 291	1	0.44503	0.158904	0.29768	0.039981	0.058406
Enemy 292	0	0.527579	0.010545	0.046146	0.122269	0.293462
Enemy 293	0	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 294	2	0.375155	0.186315	0.331476	0.045518	0.061537
Enemy 295	0	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 296	0	0.39901	0.177068	0.320346	0.043361	0.060215
Enemy 297	0	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 298	1	0.779063	0.049134	0.121837	0.015684	0.034281
Enemy 299	2	0.374915	0.187168	0.332138	0.044986	0.060793
Enemy 300	3	0.19636	0.031101	0.097399	0.271855	0.403286
Enemy 301	4	0.198122	0.030917	0.097056	0.270797	0.403107
Enemy 302	0	0.360234	0.114637	0.254198	0.112808	0.158123
Enemy 303	2	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 304	0	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 305	1	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 306	0	0.431493	0.01471	0.059049	0.158573	0.336175
Enemy 307	2	0.355833	0.109533	0.246909	0.119611	0.168114

Enemy 308	1	0.247685	0.432672	0.201979	0.047006	0.070658
Enemy 309	0	0.375144	0.187072	0.332027	0.04497	0.060786
Enemy 310	0	0.489814	0.012073	0.051075	0.136044	0.310994
Enemy 311	1	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 312	0	0.196372	0.0311	0.097396	0.271848	0.403284
Enemy 313	2	0.374975	0.186958	0.331975	0.045117	0.060975
Enemy 314	4	0.196605	0.031075	0.097351	0.271708	0.403261
Enemy 315	0	0.759012	0.053907	0.13179	0.017714	0.037577
Enemy 316	2	0.300307	0.069685	0.180658	0.184301	0.26505
Enemy 317	1	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 318	0	0.37831	0.168413	0.316292	0.057946	0.079039
Enemy 319	2	0.374928	0.187163	0.332131	0.044985	0.060792
Enemy 320	3	0.19636	0.031101	0.097399	0.271855	0.403286
Enemy 321	2	0.374915	0.187168	0.332138	0.044986	0.060793
Enemy 322	0	0.527585	0.010544	0.046145	0.122267	0.293459
Enemy 323	2	0.378495	0.162733	0.310932	0.062436	0.085403
Enemy 324	4	0.197093	0.031301	0.097895	0.271315	0.402396
Enemy 325	4	0.525411	0.010629	0.046424	0.123043	0.294493
Enemy 326	3	0.196361	0.031101	0.097399	0.271854	0.403285
Enemy 327	0	0.527585	0.010544	0.046145	0.122267	0.293459
Enemy 328	2	0.37741	0.176108	0.323131	0.052294	0.071056
Enemy 329	4	0.203998	0.033222	0.102619	0.266182	0.393979
Enemy 330	2	0.377303	0.176747	0.323677	0.051846	0.070426
Enemy 331	0	0.374915	0.187168	0.332138	0.044986	0.060793
Enemy 332	1	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 333	0	0.377455	0.175831	0.322893	0.052489	0.071332
Enemy 334	1	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 335	0	0.19636	0.031101	0.097399	0.271855	0.403286
Enemy 336	4	0.196498	0.031087	0.097372	0.271772	0.403272

Enemy 337	0	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 338	4	0.288799	0.023111	0.080821	0.221746	0.385523
Enemy 339	0	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 340	0	0.527585	0.010544	0.046145	0.122267	0.293459
Enemy 341	0	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 342	2	0.339119	0.094142	0.223345	0.142054	0.20134
Enemy 343	0	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 344	3	0.19636	0.031101	0.097399	0.271855	0.403286
Enemy 345	2	0.344267	0.098384	0.230083	0.135569	0.191696
Enemy 346	2	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 347	3	0.19636	0.031101	0.097399	0.271855	0.403286
Enemy 348	0	0.389206	0.175524	0.320846	0.048171	0.066253
Enemy 349	4	0.196971	0.031037	0.09728	0.271487	0.403225
Enemy 350	2	0.376953	0.178691	0.325319	0.050502	0.068535
Enemy 351	4	0.527585	0.010544	0.046145	0.122267	0.293459
Enemy 352	2	0.343683	0.097884	0.229299	0.136321	0.192813
Enemy 353	1	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 354	0	0.375058	0.186662	0.331746	0.045301	0.061233
Enemy 355	0	0.527585	0.010544	0.046145	0.122267	0.293459
Enemy 356	2	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 357	3	0.524283	0.010673	0.046569	0.123447	0.295028
Enemy 358	0	0.19636	0.031101	0.097399	0.271855	0.403286
Enemy 359	4	0.19636	0.031101	0.097399	0.271855	0.403286
Enemy 360	0	0.22214	0.038611	0.115484	0.252301	0.371463
Enemy 361	2	0.376041	0.182882	0.328761	0.047708	0.064608
Enemy 362	2	0.374919	0.187166	0.332135	0.044987	0.060793
Enemy 363	1	0.247685	0.432672	0.201979	0.047006	0.070658
Enemy 364	0	0.19636	0.031101	0.097399	0.271855	0.403286
Enemy 365	2	0.290789	0.064962	0.171607	0.193501	0.279142

Enemy 366	4	0.377982	0.171986	0.319527	0.055262	0.075243
Enemy 367	0	0.374912	0.187175	0.332134	0.044987	0.060793
Enemy 368	3	0.523421	0.010707	0.04668	0.123756	0.295437
Enemy 369	4	0.526284	0.010595	0.046312	0.122731	0.294078
Enemy 370	4	0.527563	0.010545	0.046148	0.122274	0.293469
Enemy 371	0	0.527623	0.010546	0.04615	0.122254	0.293427
Enemy 372	0	0.380983	0.15926	0.307262	0.064246	0.08825
Enemy 373	0	0.375367	0.185541	0.330872	0.046004	0.062218
Enemy 374	1	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 375	1	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 376	0	0.4574	0.013494	0.05545	0.148368	0.325289
Enemy 377	0	0.247685	0.432672	0.201979	0.047006	0.070658
Enemy 378	4	0.201898	0.032631	0.101173	0.267752	0.396547
Enemy 379	4	0.485689	0.012248	0.051624	0.137586	0.312853
Enemy 380	2	0.378341	0.167954	0.315869	0.058299	0.079537
Enemy 381	0	0.247685	0.432672	0.201979	0.047006	0.070658
Enemy 382	2	0.373738	0.137479	0.283757	0.085953	0.119072
Enemy 383	3	0.773596	0.046859	0.119864	0.018757	0.040924
Enemy 384	0	0.46142	0.013312	0.0549	0.146813	0.323555
Enemy 385	3	0.196482	0.031088	0.097375	0.271781	0.403274
Enemy 386	3	0.19636	0.031101	0.097399	0.271855	0.403286
Enemy 387	1	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 388	3	0.196366	0.0311	0.097397	0.271851	0.403285
Enemy 389	0	0.375897	0.183475	0.329236	0.047324	0.064068
Enemy 390	1	0.247687	0.432668	0.201981	0.047006	0.070658
Enemy 391	4	0.196473	0.031132	0.097475	0.271772	0.403148
Enemy 392	1	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 393	4	0.19636	0.031101	0.097399	0.271855	0.403286
Enemy 394	4	0.19636	0.031101	0.097399	0.271855	0.403286

Enemy 395	0	0.374914	0.18717	0.332137	0.044986	0.060793
Enemy 396	0	0.242251	0.026766	0.088846	0.245748	0.39639
Enemy 397	1	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 398	1	0.374915	0.187169	0.332138	0.044986	0.060792
Enemy 399	3	0.19636	0.031101	0.097399	0.271855	0.403286
Enemy 400	3	0.19636	0.031101	0.097399	0.271855	0.403286

Cara membaca hasil klasifikasi pada Tabel 4.18 sebelumnya sudah dijelaskan pada Sub-bab 3.6.3 dan Sub-bab 3.6.4. Hasil ditampilkan pada Tabel 4.19 terdapat 51 karakter yang terklasifikasi dengan benar atau sesuai dengan tipe saat karakter musuh dibuat. Kemudian pada karakter tersebut setiap angka tingkat keyakinan ditandai dengan warna hijau, dengan beracuan pada pada Tabel 5.16 sampai dengan Tabel 5.30 untuk musuh dengan urutan ke 281 sampai 400 pada bagian kolom *type*. Maka dari 120 karakter tersebut, terdapat 51 karakter yang berhasil terklasifikasi, bila dinyatakan dalam presentase maka menjadi 42.5% karakter pemain berhasil terklasifikasi.

BAB 5

PENUTUP

Setelah penerapan metode terhadap masalah yang ingin diselesaikan pada Bab 3 dan dilakukan pengujian terhadap metode tersebut pada Bab 4 maka didapatlah kesimpulan yang akan dijabarkan pada Sub-bab berikut. Kemudian dilanjutkan dengan saran untuk penelitian kedepannya pada Sub-bab selanjutnya.

5.1 Kesimpulan

Pada saat pengerjaan metodologi pada BAB 3 yang dilanjutkan dengan pengujian yang tertuang pada BAB 4, mengharuskan untuk membagi permainan dengan genre RPG ke dalam dua golongan lagi yaitu *single-character* dan *multi-character*. Hal ini dilakukan demi memudahkan pembuatan atribut *gameplay*, *single-character* biasanya digunakan pada permainan RPG dengan genre WRPG, ARPG, SRPG, dan MMORPG, sedangkan *multi-character* biasanya digunakan pada permainan RPG dengan genre TRPG dan JRPG.

Digunakannya metode k-NN, Distribusi Normal, dan Naive Bayes dalam melakukan proses distribusi atribut *gameplay* dalam permainan RPG menjadikan atribut *gameplay* dari pemain dan musuh menjadi terpola dari level terendah ke level tertinggi untuk pemain, mejadi tersebar dengan merata untuk distribusi musuh yang harus dihadapi oleh pemain disetiap levelnya. Hal semacam itu akan memudahkan pengembang permainan dalam mendesain permainan, terlebih lagi dalam pembuatan permainan dengan level yang tinggi, karakter, dan musuh yang berjumlah banyak.

Penggunaan *Neural Network* untuk klasifikasi *hero* pada permainan Dota 2 mampu mencapai 64% *hero* berhasil terklasifikasi. Sedangkan untuk klasifikasi karakter pemain yang dibuat menggunakan metode k-NN dan Naive

Bayes dengan jumlah 8 karakter mampu mencapai 50% pada data *testing*, berarti 50% dari karakter sesuai dengan tipe yang ingin dibuat. Kemudian untuk klasifikasi karakter musuh yang juga dibuat menggunakan metode tersebut dengan jumlah 400 karakter mampu mencapai 42.5% pada data *testing*, berarti 42.5% dari karakter sesuai dengan tipe yang ingin dibuat.

Tinggi dan rendahnya hasil dari klasifikasi sangat bergantung pada pola atribut *gameplay* yang dihasilkan itu sendiri, dari pola atribut *gameplay* tersebut mampu membentuk koerelasi fitur dengan sekumpulan fitur yang berpengaruh terhadap benarnya hasil klasifikasi. Maksud dari berpengaruh disini adalah mampu membentuk pola atribut *gameplay* yang sesuai dengan tipe.

5.2 Saran

Dalam penelitian kedepannya sangat disarankan untuk meneliti tentang bagaimana mensimulasikan pertaarungan secara otomatis pada permainan genre RPG. Hal tersebut bertujuan guna mencoba mensimulasikan nilai yang dihasilkan dari penelitian ini yang berupa atribut *gameplay*.

LAMPIRAN

Tabel 5.1: Hasil keseluruhan data HP dan MP pada pemain.

Levels	HP	MP
1	159	89
2	163	93
3	167	97
4	171	101
5	175	105
6	179	109
7	183	113
8	187	117
9	191	121
10	195	125
11	199	129
12	203	133
13	207	137
14	211	141
15	215	145
16	219	149
17	223	153
18	227	157
19	231	161
20	235	165
21	239	169
22	243	173
23	247	177
24	251	181
25	255	185
26	259	189
27	263	193
28	267	197
29	271	201
30	275	205
31	279	209
32	283	213
33	287	217
34	291	221
35	295	225

Levels	HP	MP
36	299	229
37	303	233
38	307	237
39	311	241
40	315	245
41	319	249
42	323	253
43	327	257
44	331	261
45	335	265
46	339	269
47	343	273
48	347	277
49	351	281
50	355	285
51	359	289
52	363	293
53	367	297
54	371	301
55	375	305
56	379	309
57	383	313
58	387	317
59	391	321
60	395	325
61	399	329
62	403	333
63	407	337
64	411	341
65	415	345
66	419	349
67	423	353
68	427	357
69	431	361
70	435	365

Levels	HP	MP
71	439	369
72	443	373
73	447	377
74	451	381
75	455	385
76	459	389
77	463	393
78	467	397
79	471	401
80	475	405
81	479	409
82	483	413
83	487	417
84	491	421
85	495	425
86	499	429
87	503	433
88	507	437
89	511	441
90	515	445
91	519	449
92	523	453
93	527	457
94	531	461
95	535	465
96	539	469
97	543	473
98	547	477
99	551	481
100	555	485

Tabel 5.2: Hasil keseluruh data atribut *gameplay* pada pemain (Bag. 1).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
1	159	89	0	2	0	0	1	1	1	0	0	0
2	163	93	0	2	0	0	1	2	0	2	0	0
3	167	97	0	2	0	0	1	1	0	1	1	0
4	171	101	0	2	0	0	1	2	2	2	1	0
5	175	105	0	2	0	0	1	0	0	0	0	0
6	179	109	0	2	0	0	1	2	0	0	1	0
7	183	113	0	2	0	0	1	0	0	0	1	0
8	187	117	0	2	0	0	1	0	0	0	0	0
9	191	121	0	2	0	0	1	0	0	2	1	0
10	195	125	0	2	0	0	1	2	0	0	1	0
11	199	129	0	2	0	0	1	0	0	0	0	0
12	203	133	0	2	0	0	1	1	0	0	1	2
13	207	137	0	2	0	0	1	1	2	0	0	0
14	211	141	0	2	0	0	1	0	2	2	1	0
15	215	145	0	2	0	0	1	1	0	0	2	0
16	219	149	0	2	0	0	1	0	0	2	0	0
17	223	153	0	2	0	0	1	2	0	0	0	2
18	227	157	0	2	0	0	1	1	0	0	1	2
19	231	161	0	2	0	0	1	1	0	2	0	0
20	235	165	0	2	0	0	1	1	0	0	0	2
21	239	169	0	2	0	0	1	1	0	2	1	0
22	243	173	0	2	0	0	1	1	0	0	1	1
23	247	177	0	2	0	0	1	0	0	0	1	0
24	251	181	0	2	0	0	1	0	0	2	2	1
25	255	185	0	2	0	0	1	0	1	0	1	0

Tabel 5.3: Hasil keseluruh data atribut *gameplay* pada pemain (Bag. 2).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
26	259	189	0	2	0	0	1	1	0	0	0	2
27	263	193	0	2	0	0	1	1	0	0	1	0
28	267	197	0	2	0	0	1	1	0	0	1	0
29	271	201	0	2	0	0	1	1	0	2	0	1
30	275	205	0	2	0	0	1	1	0	0	0	0
31	279	209	0	2	0	0	1	1	2	0	0	1
32	283	213	0	2	0	0	1	1	0	0	1	1
33	287	217	0	2	0	0	1	1	0	0	1	2
34	291	221	0	2	0	0	1	0	0	0	1	2
35	295	225	0	2	0	0	1	0	0	0	1	1
36	299	229	0	2	0	0	1	1	0	0	0	1
37	303	233	0	2	0	0	1	0	0	2	0	0
38	307	237	0	2	0	0	1	0	0	2	1	0
39	311	241	0	2	0	0	1	0	2	2	2	0
40	315	245	0	2	0	0	1	2	0	0	1	1
41	319	249	0	2	0	0	1	1	0	0	1	0
42	323	253	0	2	0	0	1	2	0	2	0	2
43	327	257	0	2	0	0	1	0	2	2	1	2
44	331	261	0	2	0	0	1	2	1	2	0	0
45	335	265	0	2	0	0	1	1	0	0	0	0
46	339	269	0	2	0	0	1	1	0	2	0	0
47	343	273	0	2	0	0	1	0	0	0	1	0
48	347	277	0	2	0	0	1	0	2	0	1	0
49	351	281	0	2	0	0	1	0	2	0	1	2
50	355	285	0	2	0	0	1	2	0	0	1	1

Tabel 5.4: Hasil keseluruh data atribut *gameplay* pada pemain (Bag. 3).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
51	359	289	0	2	0	0	1	1	0	0	0	0
52	363	293	0	2	0	0	1	0	2	0	1	2
53	367	297	0	2	0	0	1	1	0	2	0	0
54	371	301	0	2	0	0	1	2	0	0	1	0
55	375	305	0	2	0	0	1	2	0	0	0	0
56	379	309	0	2	0	0	1	2	0	0	0	2
57	383	313	0	2	0	0	1	2	0	0	0	0
58	387	317	0	2	0	0	1	0	0	2	0	1
59	391	321	0	2	0	0	1	1	2	0	1	1
60	395	325	0	2	0	0	1	1	0	2	1	0
61	399	329	0	2	0	0	1	0	0	0	0	1
62	403	333	0	2	0	0	1	0	2	2	1	0
63	407	337	0	2	0	0	1	0	2	2	1	0
64	411	341	0	2	0	0	1	0	2	0	1	1
65	415	345	0	2	0	0	1	1	0	0	1	0
66	419	349	0	2	0	0	1	0	0	2	1	2
67	423	353	0	2	0	0	1	1	0	0	1	0
68	427	357	0	2	0	0	1	1	0	2	1	0
69	431	361	0	2	0	0	1	0	0	0	1	0
70	435	365	0	2	0	0	1	2	1	2	1	0
71	439	369	0	2	0	0	1	1	0	0	1	1
72	443	373	0	2	0	0	1	0	0	0	0	0
73	447	377	0	2	0	0	1	0	0	0	1	1
74	451	381	0	2	0	0	1	1	0	0	1	2
75	455	385	0	2	0	0	1	1	0	0	0	0

Tabel 5.5: Hasil keseluruh data atribut *gameplay* pada pemain (Bag. 4).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
76	459	389	0	2	0	0	1	1	0	0	1	0
77	463	393	0	2	0	0	1	0	0	2	1	2
78	467	397	0	2	0	0	1	1	0	2	1	1
79	471	401	0	2	0	0	1	2	0	0	0	1
80	475	405	0	2	0	0	1	1	0	2	1	0
81	479	409	0	2	0	0	1	1	0	0	1	2
82	483	413	0	2	0	0	1	1	0	0	0	0
83	487	417	0	2	0	0	1	0	2	2	1	0
84	491	421	0	2	0	0	1	0	2	0	0	0
85	495	425	0	2	0	0	1	0	0	0	1	0
86	499	429	0	2	0	0	1	0	0	0	0	0
87	503	433	0	2	0	0	1	1	0	2	0	0
88	507	437	0	2	0	0	1	0	2	0	1	0
89	511	441	0	2	0	0	1	0	0	2	0	0
90	515	445	0	2	0	0	1	1	0	0	0	0
91	519	449	0	2	0	0	1	0	0	0	0	2
92	523	453	0	2	0	0	1	0	0	0	1	0
93	527	457	0	2	0	0	1	1	0	0	1	2
94	531	461	0	2	0	0	1	1	2	0	1	2
95	535	465	0	2	0	0	1	0	0	0	0	0
96	539	469	0	2	0	0	1	1	0	0	2	0
97	543	473	0	2	0	0	1	2	0	0	1	2
98	547	477	0	2	0	0	1	0	0	0	0	2
99	551	481	0	2	0	0	1	0	0	2	1	1
100	555	485	0	2	0	0	1	0	0	0	1	0
TOTAL								**74**	**38**	**63**	**65**	**60**

Tabel 5.6: Hasil keseluruhan data HP dan MP karakter pertama pada pemain (*multi-character*).

Levels	HP	MP
1	224	100
2	228	102
3	232	104
4	236	106
5	240	108
6	244	110
7	248	112
8	252	114
9	256	116
10	260	118
11	264	120
12	268	122
13	272	124
14	276	126
15	280	128
16	284	130
17	288	132
18	292	134
19	296	136
20	300	138
21	304	140
22	308	142
23	312	144
24	316	146
25	320	148
26	324	150
27	328	152
28	332	154
29	336	156
30	340	158
31	344	160
32	348	162
33	352	164
34	356	166
35	360	168

Levels	HP	MP
36	364	170
37	368	172
38	372	174
39	376	176
40	380	178
41	384	180
42	388	182
43	392	184
44	396	186
45	400	188
46	404	190
47	408	192
48	412	194
49	416	196
50	420	198
51	424	200
52	428	202
53	432	204
54	436	206
55	440	208
56	444	210
57	448	212
58	452	214
59	456	216
60	460	218
61	464	220
62	468	222
63	472	224
64	476	226
65	480	228
66	484	230
67	488	232
68	492	234
69	496	236
70	500	238

Levels	HP	MP
71	504	240
72	508	242
73	512	244
74	516	246
75	520	248
76	524	250
77	528	252
78	532	254
79	536	256
80	540	258
81	544	260
82	548	262
83	552	264
84	556	266
85	560	268
86	564	270
87	568	272
88	572	274
89	576	276
90	580	278
91	584	280
92	588	282
93	592	284
94	596	286
95	600	288
96	604	290
97	608	292
98	612	294
99	616	296
100	620	298

Tabel 5.7: Hasil keseluruh data atribut *gameplay* karakter pertama (*multi-character*). (Bag. 1).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
1	224	100	0	0	2	1	0	2	0	0	0	2
2	228	102	0	0	2	1	0	0	0	0	0	1
3	232	104	0	0	2	1	0	1	0	1	0	2
4	236	106	0	0	2	1	0	2	0	0	0	0
5	240	108	0	0	2	1	0	0	2	1	1	0
6	244	110	0	0	2	1	0	1	2	2	0	0
7	248	112	0	0	2	1	0	1	0	1	1	0
8	252	114	0	0	2	1	0	1	0	2	1	0
9	256	116	0	0	2	1	0	1	0	2	1	1
10	260	118	0	0	2	1	0	1	0	0	0	0
11	264	120	0	0	2	1	0	1	0	0	0	2
12	268	122	0	0	2	1	0	2	0	1	0	0
13	272	124	0	0	2	1	0	1	0	1	1	0
14	276	126	0	0	2	1	0	1	0	1	0	0
15	280	128	0	0	2	1	0	1	2	0	0	2
16	284	130	0	0	2	1	0	1	2	0	0	0
17	288	132	0	0	2	1	0	1	0	0	0	2
18	292	134	0	0	2	1	0	1	0	1	0	0
19	296	136	0	0	2	1	0	0	0	0	2	0
20	300	138	0	0	2	1	0	1	0	1	0	0
21	304	140	0	0	2	1	0	2	2	2	1	0
22	308	142	0	0	2	1	0	1	2	1	0	0
23	312	144	0	0	2	1	0	0	0	2	0	0
24	316	146	0	0	2	1	0	1	0	1	0	1
25	320	148	0	0	2	1	0	1	2	1	1	0

Tabel 5.8: Hasil keseluruh data atribut *gameplay* karakter pertama (*multi-character*). (Bag. 2).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
26	324	150	0	0	2	1	0	0	0	2	0	1
27	328	152	0	0	2	1	0	1	0	0	1	0
28	332	154	0	0	2	1	0	1	0	0	2	1
29	336	156	0	0	2	1	0	1	0	0	1	0
30	340	158	0	0	2	1	0	0	0	0	2	2
31	344	160	0	0	2	1	0	1	0	1	1	0
32	348	162	0	0	2	1	0	1	0	1	1	0
33	352	164	0	0	2	1	0	1	0	1	0	0
34	356	166	0	0	2	1	0	0	0	1	1	0
35	360	168	0	0	2	1	0	0	0	1	0	2
36	364	170	0	0	2	1	0	0	0	0	0	0
37	368	172	0	0	2	1	0	2	0	1	1	2
38	372	174	0	0	2	1	0	1	0	1	0	0
39	376	176	0	0	2	1	0	1	0	2	1	1
40	380	178	0	0	2	1	0	1	0	0	1	0
41	384	180	0	0	2	1	0	1	0	2	0	1
42	388	182	0	0	2	1	0	0	0	1	1	2
43	392	184	0	0	2	1	0	0	0	0	0	0
44	396	186	0	0	2	1	0	0	0	1	0	1
45	400	188	0	0	2	1	0	1	0	0	0	2
46	404	190	0	0	2	1	0	0	2	1	0	0
47	408	192	0	0	2	1	0	1	0	1	1	1
48	412	194	0	0	2	1	0	1	0	0	0	1
49	416	196	0	0	2	1	0	0	0	1	1	0
50	420	198	0	0	2	1	0	1	0	0	0	1

Tabel 5.9: Hasil keseluruh data atribut *gameplay* karakter pertama (*multi-character*). (Bag. 3).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
51	424	200	0	0	2	1	0	0	0	0	0	2
52	428	202	0	0	2	1	0	1	0	1	1	0
53	432	204	0	0	2	1	0	0	0	0	1	1
54	436	206	0	0	2	1	0	1	2	1	0	0
55	440	208	0	0	2	1	0	2	0	0	0	0
56	444	210	0	0	2	1	0	1	0	2	1	0
57	448	212	0	0	2	1	0	1	0	0	0	1
58	452	214	0	0	2	1	0	0	0	1	0	0
59	456	216	0	0	2	1	0	1	0	1	1	0
60	460	218	0	0	2	1	0	1	0	1	0	1
61	464	220	0	0	2	1	0	0	0	1	0	0
62	468	222	0	0	2	1	0	2	2	1	0	1
63	472	224	0	0	2	1	0	1	2	0	0	1
64	476	226	0	0	2	1	0	1	0	1	1	1
65	480	228	0	0	2	1	0	1	0	2	0	1
66	484	230	0	0	2	1	0	0	0	1	0	0
67	488	232	0	0	2	1	0	0	0	1	2	1
68	492	234	0	0	2	1	0	0	0	1	1	0
69	496	236	0	0	2	1	0	1	2	0	0	1
70	500	238	0	0	2	1	0	2	2	2	0	0
71	504	240	0	0	2	1	0	1	0	1	0	1
72	508	242	0	0	2	1	0	1	2	2	0	1
73	512	244	0	0	2	1	0	2	0	1	0	2
74	516	246	0	0	2	1	0	1	2	1	1	2
75	520	248	0	0	2	1	0	1	0	1	1	0

Tabel 5.10: Hasil keseluruh data atribut *gameplay* karakter pertama (*multi-character*). (Bag. 4).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
76	459	389	0	2	0	0	1	1	0	0	1	0
77	463	393	0	2	0	0	1	0	0	2	1	2
78	467	397	0	2	0	0	1	1	0	2	1	1
79	471	401	0	2	0	0	1	2	0	0	0	1
80	475	405	0	2	0	0	1	1	0	2	1	0
81	479	409	0	2	0	0	1	1	0	0	1	2
82	483	413	0	2	0	0	1	1	0	0	0	0
83	487	417	0	2	0	0	1	0	2	2	1	0
84	491	421	0	2	0	0	1	0	2	0	0	0
85	495	425	0	2	0	0	1	0	0	0	1	0
86	499	429	0	2	0	0	1	0	0	0	0	0
87	503	433	0	2	0	0	1	1	0	2	0	0
88	507	437	0	2	0	0	1	0	2	0	1	0
89	511	441	0	2	0	0	1	0	0	2	0	0
90	515	445	0	2	0	0	1	1	0	0	0	0
91	519	449	0	2	0	0	1	0	0	0	0	2
92	523	453	0	2	0	0	1	0	0	0	1	0
93	527	457	0	2	0	0	1	1	0	0	1	2
94	531	461	0	2	0	0	1	1	2	0	1	2
95	535	465	0	2	0	0	1	0	0	0	0	0
96	539	469	0	2	0	0	1	1	0	0	2	0
97	543	473	0	2	0	0	1	2	0	0	1	2
98	547	477	0	2	0	0	1	0	0	0	0	2
99	551	481	0	2	0	0	1	0	0	2	1	1
100	555	485	0	2	0	0	1	0	0	0	1	0
TOTAL								**88**	**32**	**81**	**43**	**56**

Tabel 5.11: Hasil keseluruhan data HP dan MP karakter kedua pada pemain (*multi-character*).

Levels	HP	MP
1	223	154
2	226	158
3	229	162
4	232	166
5	235	170
6	238	174
7	241	178
8	244	182
9	247	186
10	250	190
11	253	194
12	256	198
13	259	202
14	262	206
15	265	210
16	268	214
17	271	218
18	274	222
19	277	226
20	280	230
21	283	234
22	286	238
23	289	242
24	292	246
25	295	250
26	298	254
27	301	258
28	304	262
29	307	266
30	310	270
31	313	274
32	316	278
33	319	282
34	322	286
35	325	290

Levels	HP	MP
36	328	294
37	331	298
38	334	302
39	337	306
40	340	310
41	343	314
42	346	318
43	349	322
44	352	326
45	355	330
46	358	334
47	361	338
48	364	342
49	367	346
50	370	350
51	373	354
52	376	358
53	379	362
54	382	366
55	385	370
56	388	374
57	391	378
58	394	382
59	397	386
60	400	390
61	403	394
62	406	398
63	409	402
64	412	406
65	415	410
66	418	414
67	421	418
68	424	422
69	427	426
70	430	430

Levels	HP	MP
71	433	434
72	436	438
73	439	442
74	442	446
75	445	450
76	448	454
77	451	458
78	454	462
79	457	466
80	460	470
81	463	474
82	466	478
83	469	482
84	472	486
85	475	490
86	478	494
87	481	498
88	484	502
89	487	506
90	490	510
91	493	514
92	496	518
93	499	522
94	502	526
95	505	530
96	508	534
97	511	538
98	514	542
99	517	546
100	520	550

Tabel 5.12: Hasil keseluruh data atribut *gameplay* karakter kedua (*multi-character*). (Bag. 1).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
1	223	154	0	1	0	0	2	0	0	2	0	1
2	226	158	0	1	0	0	2	0	1	1	0	0
3	229	162	0	1	0	0	2	1	2	1	0	1
4	232	166	0	1	0	0	2	1	1	2	0	0
5	235	170	0	1	0	0	2	1	1	0	0	1
6	238	174	0	1	0	0	2	1	1	0	0	0
7	241	178	0	1	0	0	2	1	1	1	0	1
8	244	182	0	1	0	0	2	1	1	0	0	1
9	247	186	0	1	0	0	2	1	2	1	0	0
10	250	190	0	1	0	0	2	0	0	0	0	0
11	253	194	0	1	0	0	2	1	0	1	2	1
12	256	198	0	1	0	0	2	0	1	1	2	1
13	259	202	0	1	0	0	2	0	1	0	0	1
14	262	206	0	1	0	0	2	0	0	1	0	0
15	265	210	0	1	0	0	2	0	2	0	0	0
16	268	214	0	1	0	0	2	1	0	1	0	1
17	271	218	0	1	0	0	2	0	2	0	0	1
18	274	222	0	1	0	0	2	0	0	0	0	0
19	277	226	0	1	0	0	2	0	0	2	0	2
20	280	230	0	1	0	0	2	0	1	1	0	1
21	283	234	0	1	0	0	2	1	0	1	0	1
22	286	238	0	1	0	0	2	1	0	0	0	0
23	289	242	0	1	0	0	2	1	0	0	2	1
24	292	246	0	1	0	0	2	1	1	0	0	0
25	295	250	0	1	0	0	2	0	2	1	0	0

Tabel 5.13: Hasil keseluruh data atribut *gameplay* karakter kedua (*multi-character*). (Bag. 2).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
26	298	254	0	1	0	0	2	2	0	1	2	0
27	301	258	0	1	0	0	2	0	1	0	0	0
28	304	262	0	1	0	0	2	0	1	0	0	1
29	307	266	0	1	0	0	2	0	0	2	0	1
30	310	270	0	1	0	0	2	2	1	1	1	1
31	313	274	0	1	0	0	2	0	0	0	0	1
32	316	278	0	1	0	0	2	1	0	0	0	1
33	319	282	0	1	0	0	2	0	0	0	0	1
34	322	286	0	1	0	0	2	0	1	0	2	1
35	325	290	0	1	0	0	2	1	1	1	0	0
36	328	294	0	1	0	0	2	0	0	0	2	1
37	331	298	0	1	0	0	2	0	0	0	2	1
38	334	302	0	1	0	0	2	1	0	0	0	1
39	337	306	0	1	0	0	2	2	1	1	2	0
40	340	310	0	1	0	0	2	0	2	1	1	0
41	343	314	0	1	0	0	2	0	2	0	0	1
42	346	318	0	1	0	0	2	0	0	0	2	0
43	349	322	0	1	0	0	2	0	0	1	0	0
44	352	326	0	1	0	0	2	1	1	1	1	1
45	355	330	0	1	0	0	2	1	1	2	0	0
46	358	334	0	1	0	0	2	1	0	0	2	0
47	361	338	0	1	0	0	2	0	1	0	0	0
48	364	342	0	1	0	0	2	1	0	2	0	1
49	367	346	0	1	0	0	2	0	0	1	0	0
50	370	350	0	1	0	0	2	2	2	1	2	0

Tabel 5.14: Hasil keseluruh data atribut *gameplay* karakter kedua (*multi-character*). (Bag. 3).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
51	373	354	0	1	0	0	2	1	1	0	2	1
52	376	358	0	1	0	0	2	1	1	1	0	0
53	379	362	0	1	0	0	2	0	1	1	0	1
54	382	366	0	1	0	0	2	0	0	0	0	0
55	385	370	0	1	0	0	2	1	1	2	2	0
56	388	374	0	1	0	0	2	0	0	0	0	0
57	391	378	0	1	0	0	2	0	2	0	0	0
58	394	382	0	1	0	0	2	0	0	0	0	1
59	397	386	0	1	0	0	2	0	2	0	0	0
60	400	390	0	1	0	0	2	0	0	1	0	1
61	403	394	0	1	0	0	2	2	2	0	2	0
62	406	398	0	1	0	0	2	1	0	0	1	1
63	409	402	0	1	0	0	2	0	0	2	2	1
64	412	406	0	1	0	0	2	0	0	0	0	1
65	415	410	0	1	0	0	2	0	0	0	0	1
66	418	414	0	1	0	0	2	0	0	0	2	1
67	421	418	0	1	0	0	2	0	0	2	2	1
68	424	422	0	1	0	0	2	0	0	0	0	0
69	427	426	0	1	0	0	2	0	2	1	0	1
70	430	430	0	1	0	0	2	0	0	1	0	0
71	433	434	0	1	0	0	2	2	0	1	0	1
72	436	438	0	1	0	0	2	0	1	1	0	1
73	439	442	0	1	0	0	2	1	1	0	2	0
74	442	446	0	1	0	0	2	0	0	1	2	0
75	445	450	0	1	0	0	2	1	2	0	0	1

Tabel 5.15: Hasil keseluruh data atribut *gameplay* karakter kedua (*multi-character*). (Bag. 4).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
76	448	454	0	1	0	0	2	0	0	2	0	1
77	451	458	0	1	0	0	2	1	0	1	0	0
78	454	462	0	1	0	0	2	0	0	0	0	0
79	457	466	0	1	0	0	2	2	1	2	2	1
80	460	470	0	1	0	0	2	2	1	0	2	0
81	463	474	0	1	0	0	2	0	0	1	2	0
82	466	478	0	1	0	0	2	2	1	2	0	0
83	469	482	0	1	0	0	2	0	1	1	0	1
84	472	486	0	1	0	0	2	0	1	0	1	1
85	475	490	0	1	0	0	2	1	0	0	0	2
86	478	494	0	1	0	0	2	2	0	1	0	1
87	481	498	0	1	0	0	2	0	0	0	0	0
88	484	502	0	1	0	0	2	1	1	0	0	1
89	487	506	0	1	0	0	2	1	2	0	0	0
90	490	510	0	1	0	0	2	2	0	0	2	1
91	493	514	0	1	0	0	2	2	2	0	0	0
92	496	518	0	1	0	0	2	1	0	0	0	1
93	499	522	0	1	0	0	2	1	0	0	0	0
94	502	526	0	1	0	0	2	2	0	0	2	1
95	505	530	0	1	0	0	2	0	1	0	1	1
96	508	534	0	1	0	0	2	1	2	0	0	1
97	511	538	0	1	0	0	2	0	0	0	2	1
98	514	542	0	1	0	0	2	0	1	1	0	1
99	517	546	0	1	0	0	2	1	1	0	1	0
100	520	550	0	1	0	0	2	0	2	0	1	0
TOTAL								**60**	**68**	**57**	**58**	**57**

Tabel 5.16: Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 1).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 1	1	44	53	2	0	1	0	1	0	2	2	6	2	2
Enemy 2	1	41	21	0	0	0	0	1	0	2	2	5	2	2
Enemy 3	1	43	63	0	1	0	0	0	0	2	5	2	2	2
Enemy 4	1	42	52	0	1	0	0	2	0	2	3	2	2	2
Enemy 5	1	44	50	0	0	0	0	0	0	2	7	2	2	2
Enemy 6	1	76	23	0	0	0	0	1	0	2	2	5	2	2
Enemy 7	2	49	279	1	0	1	1	1	0	2	2	9	2	2
Enemy 8	2	41	53	0	0	1	1	0	1	2	10	2	3	2
Enemy 9	2	45	50	2	0	1	0	0	1	2	3	5	3	2
Enemy 10	2	42	24	4	2	0	0	2	0	2	2	2	2	2
Enemy 11	2	87	27	0	0	0	2	1	0	2	3	12	3	2
Enemy 12	2	46	48	2	2	0	0	0	0	2	2	5	2	2
Enemy 13	3	46	46	2	0	0	0	0	0	2	2	3	2	2
Enemy 14	3	54	25	3	1	0	1	0	0	3	7	3	2	2
Enemy 15	3	44	63	0	0	0	0	0	1	3	3	2	2	2
Enemy 16	4	99	21	0	0	0	0	2	0	3	4	6	4	3
Enemy 17	4	52	60	2	0	1	1	0	0	3	4	6	4	2
Enemy 18	4	44	37	4	1	1	1	0	0	3	4	3	4	2
Enemy 19	4	40	41	2	0	1	2	0	0	2	4	11	4	3
Enemy 20	5	65	36	3	0	0	1	1	0	4	7	2	3	4
Enemy 21	5	57	215	1	1	0	1	1	0	2	3	12	4	3
Enemy 22	5	46	149	1	1	1	0	1	0	2	2	7	2	4
Enemy 23	5	86	41	0	0	1	0	0	0	4	2	12	2	3
Enemy 24	5	40	22	4	2	0	0	2	0	2	3	2	2	3
Enemy 25	5	50	104	0	0	1	1	1	0	4	3	2	4	3

Tabel 5.17: Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 2).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 26	5	60	62	2	1	0	0	1	0	3	2	3	3	3
Enemy 27	5	61	274	1	1	0	1	1	0	4	4	6	3	2
Enemy 28	6	57	59	2	1	1	0	0	1	2	3	7	5	2
Enemy 29	6	51	36	0	0	0	2	2	0	3	2	4	5	2
Enemy 30	6	41	22	4	1	0	0	1	1	3	6	3	2	2
Enemy 31	6	57	233	1	0	0	0	0	2	2	4	4	5	4
Enemy 32	6	62	24	4	0	2	0	0	1	3	11	2	2	3
Enemy 33	6	99	40	0	0	0	0	2	0	3	2	12	3	3
Enemy 34	6	94	36	0	0	0	0	1	0	4	2	3	2	2
Enemy 35	6	59	24	4	0	2	0	0	0	4	3	2	5	2
Enemy 36	7	57	75	0	0	0	1	0	1	5	10	4	3	4
Enemy 37	7	63	26	0	0	0	2	0	1	2	5	14	4	5
Enemy 38	7	63	330	1	1	0	0	0	1	2	3	7	4	4
Enemy 39	8	51	59	0	0	0	1	2	0	2	12	5	5	4
Enemy 40	8	78	50	3	2	0	0	2	0	2	10	4	3	5
Enemy 41	8	72	25	0	0	0	1	1	0	4	5	18	2	2
Enemy 42	8	66	73	2	0	0	1	1	1	3	2	3	5	3
Enemy 43	8	58	51	4	2	0	2	0	0	5	4	2	5	5
Enemy 44	8	75	50	0	1	0	1	0	0	3	5	14	4	2
Enemy 45	9	75	79	2	0	0	0	0	0	4	3	10	4	2
Enemy 46	9	49	50	0	1	0	0	0	1	5	7	2	6	3
Enemy 47	9	54	47	0	2	0	1	0	0	2	3	6	2	5
Enemy 48	9	64	291	1	1	1	1	0	0	6	7	10	4	4
Enemy 49	9	81	28	0	0	0	0	0	1	6	2	3	4	4
Enemy 50	9	53	57	0	0	1	0	0	2	6	8	4	3	2

Tabel 5.18: Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 3).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 51	9	69	27	3	1	0	0	0	1	2	16	4	3	6
Enemy 52	9	43	25	4	0	0	0	1	1	6	7	2	6	2
Enemy 53	9	74	57	4	1	0	1	1	0	3	7	2	3	6
Enemy 54	10	74	62	3	0	0	1	1	0	4	8	4	7	5
Enemy 55	10	80	52	4	1	0	0	0	1	5	5	4	5	2
Enemy 56	11	53	49	4	0	0	1	1	1	3	8	4	4	5
Enemy 57	11	71	78	2	0	0	0	0	0	6	8	2	4	4
Enemy 58	11	67	64	4	0	1	1	0	0	5	7	3	8	6
Enemy 59	11	65	77	0	1	0	0	0	0	7	7	2	3	2
Enemy 60	11	64	292	1	0	0	1	0	0	4	8	12	6	3
Enemy 61	11	56	88	0	0	1	1	0	1	7	5	3	2	4
Enemy 62	11	95	98	0	2	0	1	0	0	7	4	2	7	6
Enemy 63	12	80	56	3	0	0	2	0	0	3	8	2	3	2
Enemy 64	12	68	48	4	1	0	0	0	1	4	5	6	4	6
Enemy 65	12	50	77	0	0	0	0	1	2	2	9	3	6	4
Enemy 66	12	66	64	0	1	0	1	0	1	4	8	25	3	4
Enemy 67	12	90	59	4	0	0	2	0	0	3	4	3	4	5
Enemy 68	12	49	53	2	0	0	0	0	0	2	7	4	3	6
Enemy 69	12	60	71	2	1	1	0	0	0	4	8	11	7	4
Enemy 70	12	50	66	0	0	0	0	0	0	6	9	5	2	3
Enemy 71	13	64	57	3	1	0	0	1	0	6	12	4	3	6
Enemy 72	13	96	51	4	0	1	0	0	0	7	11	6	8	3
Enemy 73	13	98	105	2	0	1	0	0	1	8	2	7	7	3
Enemy 74	14	54	54	0	0	0	0	0	0	9	7	23	7	3
Enemy 75	14	132	44	0	1	1	0	0	0	2	5	9	2	2

Tabel 5.19: Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 4).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 76	14	105	41	3	1	0	0	0	1	8	19	5	8	7
Enemy 77	15	115	195	0	0	0	1	0	0	6	11	7	6	4
Enemy 78	15	102	107	2	0	0	1	0	1	3	8	6	2	7
Enemy 79	15	77	86	2	0	0	0	0	0	7	8	6	5	8
Enemy 80	15	101	67	4	2	0	2	0	0	8	6	2	6	2
Enemy 81	15	74	73	0	0	2	0	0	0	8	6	9	6	5
Enemy 82	15	260	82	3	0	0	0	0	0	2	10	7	8	4
Enemy 83	16	116	46	4	0	0	0	0	0	4	7	4	10	2
Enemy 84	17	126	40	0	0	0	2	0	0	4	12	20	11	4
Enemy 85	17	95	47	3	0	0	1	0	2	8	8	7	10	2
Enemy 86	17	104	20	0	0	1	1	0	0	5	12	22	3	8
Enemy 87	17	94	50	0	0	0	1	0	0	6	11	11	3	2
Enemy 88	17	57	100	0	1	0	1	0	1	8	3	2	6	8
Enemy 89	17	100	72	0	0	2	0	0	2	9	10	10	6	4
Enemy 90	17	73	69	0	0	0	0	0	0	8	7	17	8	9
Enemy 91	17	117	121	2	0	1	0	1	0	10	5	7	8	8
Enemy 92	17	63	63	2	0	0	2	0	0	2	12	9	2	4
Enemy 93	18	82	39	3	0	0	1	0	0	4	15	6	6	10
Enemy 94	18	127	89	3	0	0	0	2	0	6	8	3	6	4
Enemy 95	18	109	78	4	0	2	1	0	0	10	2	3	8	2
Enemy 96	18	112	300	1	1	0	0	1	0	10	12	20	3	3
Enemy 97	18	88	33	3	0	0	1	1	1	5	16	2	2	6
Enemy 98	18	90	101	0	1	0	1	0	1	6	8	2	11	2
Enemy 99	19	109	263	1	1	0	1	1	0	8	10	12	8	8
Enemy 100	19	132	137	2	0	0	0	0	1	9	5	9	4	2

Tabel 5.20: Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 5).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 101	19	56	49	3	0	1	0	1	1	9	14	3	8	5
Enemy 102	19	89	80	0	0	1	0	1	0	4	10	28	11	7
Enemy 103	19	50	34	4	0	2	0	2	0	3	8	3	3	5
Enemy 104	19	71	75	0	0	1	0	1	1	7	4	2	4	7
Enemy 105	19	84	92	0	2	0	0	2	0	7	7	3	6	2
Enemy 106	19	354	90	3	0	0	0	0	2	10	10	4	4	10
Enemy 107	20	126	128	2	0	2	0	0	2	6	4	5	8	7
Enemy 108	20	100	108	2	0	2	1	0	0	6	7	9	9	9
Enemy 109	20	54	51	0	0	0	1	0	1	2	6	9	10	6
Enemy 110	21	73	88	0	0	0	0	0	0	8	10	7	3	4
Enemy 111	21	110	101	4	1	0	0	1	1	8	3	9	13	4
Enemy 112	21	105	33	4	0	0	2	0	2	10	13	10	9	10
Enemy 113	21	54	120	0	0	0	1	1	0	10	13	3	12	3
Enemy 114	21	56	115	0	1	0	0	1	1	8	10	3	13	6
Enemy 115	21	60	59	4	1	0	0	1	1	3	14	9	2	3
Enemy 116	22	51	82	0	0	0	0	0	2	4	7	4	13	9
Enemy 117	22	149	261	0	2	0	0	2	0	2	18	10	5	5
Enemy 118	22	115	120	2	1	0	0	0	1	8	6	6	11	4
Enemy 119	22	148	236	0	0	0	0	1	1	5	9	6	8	2
Enemy 120	22	81	87	2	0	0	0	0	1	10	14	7	13	8
Enemy 121	22	101	103	2	0	0	1	0	0	10	10	5	9	4
Enemy 122	23	124	333	1	0	2	0	1	0	4	4	18	8	10
Enemy 123	23	105	153	0	2	0	0	1	0	12	14	4	4	2
Enemy 124	23	138	66	4	1	0	1	0	0	9	12	11	13	6
Enemy 125	23	141	36	3	0	1	1	1	0	11	20	11	8	4

Tabel 5.21: Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 6).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 126	23	132	134	2	0	1	0	1	1	5	6	11	5	9
Enemy 127	23	159	112	0	0	0	1	0	0	11	5	7	8	4
Enemy 128	23	94	36	4	2	0	0	1	0	8	2	7	13	4
Enemy 129	24	139	236	0	1	0	0	1	0	4	15	7	9	9
Enemy 130	24	126	48	0	0	0	0	0	0	8	16	26	8	3
Enemy 131	24	57	123	1	1	0	0	1	1	4	6	7	10	3
Enemy 132	24	106	105	0	0	1	0	2	0	8	10	27	5	10
Enemy 133	24	131	133	1	1	0	0	1	1	8	2	13	11	9
Enemy 134	25	102	102	2	0	1	0	1	1	9	4	8	5	9
Enemy 135	25	78	126	0	1	1	0	0	0	4	7	5	6	7
Enemy 136	25	140	131	4	1	1	0	0	0	13	7	9	4	3
Enemy 137	26	72	72	2	0	0	1	1	0	3	5	5	16	2
Enemy 138	26	103	102	4	0	2	0	2	0	11	5	4	15	12
Enemy 139	26	81	73	4	0	1	1	0	1	11	10	7	16	4
Enemy 140	26	75	257	1	1	2	0	0	0	11	14	22	7	10
Enemy 141	27	138	132	4	0	2	0	0	1	6	5	12	16	11
Enemy 142	27	152	69	0	1	0	1	0	0	4	8	24	13	8
Enemy 143	27	192	57	0	0	0	0	0	0	8	18	37	4	2
Enemy 144	28	85	94	2	0	2	0	1	0	15	15	3	4	4
Enemy 145	28	48	84	0	0	0	2	2	0	15	15	3	6	13
Enemy 146	28	98	83	4	1	0	2	0	0	4	12	10	16	11
Enemy 147	28	151	159	2	0	1	1	1	0	16	16	4	15	14
Enemy 148	28	126	22	4	0	0	2	0	0	7	3	13	2	11
Enemy 149	28	130	291	1	1	1	0	0	1	12	4	17	4	9
Enemy 150	29	48	29	0	0	0	2	2	0	7	18	43	11	4

Tabel 5.22: Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 7).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 151	29	143	136	4	0	0	0	1	0	13	9	12	7	2
Enemy 152	29	127	118	0	0	0	0	1	1	7	4	5	13	12
Enemy 153	29	95	267	1	0	0	1	0	1	12	19	21	17	5
Enemy 154	29	124	130	2	1	1	0	0	0	9	14	9	6	15
Enemy 155	30	47	124	2	1	0	1	0	0	9	3	8	8	2
Enemy 156	30	113	210	0	0	1	2	0	0	6	13	11	11	13
Enemy 157	30	121	121	4	0	1	1	0	1	13	18	4	12	7
Enemy 158	30	175	354	0	0	1	1	0	0	5	15	6	3	14
Enemy 159	30	151	162	2	0	0	1	0	0	13	5	11	16	9
Enemy 160	30	148	163	0	0	0	0	0	0	13	11	9	8	12
Enemy 161	30	158	152	4	0	0	0	2	0	10	8	13	15	8
Enemy 162	30	163	168	2	0	1	0	1	1	13	18	8	7	8
Enemy 163	31	59	70	0	1	1	0	0	0	2	16	11	9	2
Enemy 164	31	180	188	2	2	0	0	0	2	10	15	6	13	7
Enemy 165	31	125	68	4	0	1	1	0	1	16	17	3	9	14
Enemy 166	31	159	151	3	0	0	0	0	0	10	9	5	10	4
Enemy 167	31	189	342	1	0	0	1	1	0	10	20	25	5	9
Enemy 168	32	102	162	2	1	0	1	0	0	14	10	8	4	5
Enemy 169	32	85	138	0	1	0	0	1	1	10	15	2	13	12
Enemy 170	33	42	49	2	0	2	0	0	0	7	15	5	13	11
Enemy 171	33	117	111	4	0	0	1	0	2	15	20	13	13	16
Enemy 172	33	230	69	0	2	0	0	2	0	15	15	20	13	11
Enemy 173	33	108	187	0	2	0	0	2	0	8	17	15	11	16
Enemy 174	33	70	74	2	2	0	0	0	1	12	13	14	14	8
Enemy 175	33	40	96	0	1	0	0	1	0	5	14	11	12	16

Tabel 5.23: Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 8).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 176	33	124	170	2	2	0	1	0	0	18	10	5	16	8
Enemy 177	33	128	232	1	1	0	1	0	1	6	18	24	14	6
Enemy 178	35	271	90	3	0	0	1	0	2	14	17	15	10	10
Enemy 179	35	167	164	4	0	1	0	1	1	4	9	3	10	4
Enemy 180	35	124	96	3	1	1	1	0	0	16	22	9	7	12
Enemy 181	35	139	100	4	2	0	0	0	1	18	11	14	10	6
Enemy 182	35	144	148	2	0	0	2	2	0	17	21	11	4	3
Enemy 183	36	57	166	0	2	0	0	0	2	18	14	3	2	17
Enemy 184	36	138	134	4	0	0	0	1	1	9	23	14	5	15
Enemy 185	36	133	135	0	0	1	0	0	0	19	7	2	5	3
Enemy 186	37	189	181	4	1	1	1	0	0	16	8	2	3	5
Enemy 187	37	191	115	0	0	0	2	0	0	13	20	44	4	17
Enemy 188	37	200	193	0	1	2	0	0	0	3	22	25	9	8
Enemy 189	37	67	102	0	1	2	0	0	0	8	15	12	2	12
Enemy 190	38	189	161	0	0	2	0	0	0	16	13	14	3	12
Enemy 191	38	50	145	2	0	0	1	1	1	20	9	4	8	3
Enemy 192	38	382	197	3	0	0	0	2	1	19	10	4	19	3
Enemy 193	39	116	259	1	0	1	0	1	0	5	7	14	11	9
Enemy 194	39	139	216	0	0	0	0	0	1	12	19	12	9	18
Enemy 195	39	53	128	0	2	0	1	0	0	3	11	10	16	10
Enemy 196	39	209	133	3	1	1	0	0	1	10	20	6	23	14
Enemy 197	39	124	70	4	0	0	1	0	0	14	13	9	19	19
Enemy 198	39	163	161	0	1	0	0	0	1	11	14	22	12	15
Enemy 199	39	162	171	1	0	1	0	0	1	19	14	20	6	17
Enemy 200	39	149	202	0	0	0	1	0	2	9	20	16	16	7

Tabel 5.24: Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 9).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 201	39	79	51	0	0	0	1	0	0	15	11	13	19	14
Enemy 202	39	184	116	4	1	0	0	1	0	3	19	13	2	14
Enemy 203	40	101	216	1	0	0	0	0	2	12	25	26	11	2
Enemy 204	40	226	27	3	0	0	1	1	0	14	16	3	5	11
Enemy 205	40	73	135	2	0	0	1	0	1	8	11	10	20	12
Enemy 206	40	397	208	3	0	1	0	0	1	11	15	14	20	16
Enemy 207	40	102	100	4	0	0	0	0	0	17	7	3	2	11
Enemy 208	40	168	21	4	2	0	0	0	0	3	3	16	6	14
Enemy 209	41	214	413	0	0	0	0	1	1	2	20	11	9	7
Enemy 210	41	124	210	0	0	2	0	0	1	9	23	8	5	7
Enemy 211	41	170	314	0	0	2	0	0	1	10	20	8	12	12
Enemy 212	42	64	164	2	0	1	1	0	0	20	16	2	8	4
Enemy 213	42	213	211	0	0	1	0	1	1	15	26	51	21	18
Enemy 214	42	192	254	0	1	1	1	0	0	12	19	17	16	14
Enemy 215	42	40	211	2	1	1	0	0	0	9	19	4	9	8
Enemy 216	43	206	294	0	1	1	0	0	1	3	17	16	3	8
Enemy 217	43	237	245	2	1	0	2	0	0	11	4	18	8	13
Enemy 218	43	76	34	0	0	0	1	2	0	8	10	29	14	17
Enemy 219	43	165	93	4	0	0	1	0	0	12	19	16	14	21
Enemy 220	43	94	171	2	2	0	1	0	0	15	22	2	16	21
Enemy 221	44	179	128	4	1	0	0	1	1	4	3	8	14	2
Enemy 222	44	235	236	2	0	0	1	1	0	6	21	7	15	18
Enemy 223	44	134	94	0	0	0	1	0	1	20	6	8	10	21
Enemy 224	45	193	140	0	0	1	1	0	0	2	14	16	7	4
Enemy 225	45	71	184	2	0	0	0	2	2	21	25	9	7	13

Tabel 5.25: Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 10).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 226	45	204	198	0	0	0	1	0	0	14	10	16	26	22
Enemy 227	45	136	136	0	0	0	0	0	2	5	7	11	13	3
Enemy 228	45	247	396	1	1	0	2	0	0	21	4	19	10	8
Enemy 229	45	78	357	1	0	0	1	0	1	7	4	15	21	11
Enemy 230	45	191	57	4	0	0	1	0	0	8	23	15	11	3
Enemy 231	45	173	170	4	1	0	0	0	0	3	5	14	17	10
Enemy 232	46	209	209	0	0	0	2	0	1	2	20	28	19	18
Enemy 233	46	137	244	0	1	0	0	0	0	18	20	19	2	21
Enemy 234	46	152	235	0	1	0	1	0	1	2	24	9	16	3
Enemy 235	46	300	68	0	1	1	0	0	1	19	7	17	16	20
Enemy 236	46	74	40	3	1	0	1	0	0	17	15	2	2	10
Enemy 237	46	156	186	0	0	2	2	0	0	14	16	3	2	14
Enemy 238	47	227	376	1	0	0	0	1	0	18	10	10	8	13
Enemy 239	47	74	49	4	1	0	1	1	0	20	10	3	2	9
Enemy 240	47	103	106	2	0	0	2	0	2	7	19	11	26	13
Enemy 241	47	157	65	0	0	1	1	0	0	11	10	19	17	3
Enemy 242	47	265	148	0	0	1	0	0	1	10	29	58	21	23
Enemy 243	47	215	221	2	0	0	2	0	0	13	25	7	3	19
Enemy 244	48	307	238	0	0	0	0	2	2	21	3	5	28	15
Enemy 245	48	286	450	1	1	0	0	2	0	25	9	12	17	19
Enemy 246	48	130	338	1	1	1	0	1	0	11	6	15	21	3
Enemy 247	49	109	83	4	0	0	0	1	0	16	3	10	2	5
Enemy 248	49	218	344	0	0	0	1	1	0	5	14	2	4	15
Enemy 249	49	203	212	1	1	1	0	1	0	19	21	22	13	8
Enemy 250	49	290	181	3	0	0	0	0	2	20	24	2	24	9

Tabel 5.26: Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 11).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 276	56	314	196	0	0	1	0	1	0	15	25	50	30	24
Enemy 277	57	312	69	0	0	0	2	1	0	11	19	23	26	10
Enemy 278	57	185	180	4	0	2	0	0	2	21	6	23	26	26
Enemy 279	57	461	263	3	0	0	0	0	0	13	35	19	12	3
Enemy 280	57	242	324	0	1	0	1	0	0	4	28	22	33	3
Enemy 281	57	67	90	1	2	0	0	0	2	7	34	43	21	10
Enemy 282	57	166	267	0	1	0	0	0	2	9	14	11	23	10
Enemy 283	57	266	267	2	0	0	1	0	2	26	8	2	29	24
Enemy 284	57	283	165	3	0	0	0	0	2	9	29	24	2	18
Enemy 285	57	124	131	2	0	1	1	0	1	3	18	5	4	21
Enemy 286	58	54	31	4	0	1	1	1	0	22	25	11	25	15
Enemy 287	58	314	97	4	1	1	0	0	1	20	22	19	19	5
Enemy 288	58	149	30	0	0	0	2	1	0	4	15	27	21	5
Enemy 289	58	311	184	3	0	0	0	1	0	11	25	2	5	12
Enemy 290	59	256	256	0	0	0	0	0	0	25	23	46	28	9
Enemy 291	59	71	199	0	1	0	0	1	0	21	35	8	28	23
Enemy 292	59	59	300	0	0	0	0	0	0	20	25	16	10	28
Enemy 293	59	215	206	3	0	0	2	0	2	29	19	7	8	20
Enemy 294	59	303	300	0	0	0	0	0	0	31	5	24	6	19
Enemy 295	59	112	117	2	0	1	0	1	0	8	23	9	8	11
Enemy 296	60	343	594	1	0	0	0	2	0	17	37	39	27	11
Enemy 297	60	159	154	4	0	0	0	1	1	9	8	4	6	14
Enemy 298	60	52	56	2	0	0	1	1	1	2	33	7	17	5
Enemy 299	61	347	351	0	1	0	0	1	1	28	38	5	20	5
Enemy 300	61	144	172	0	0	0	2	0	0	29	27	21	10	24

Tabel 5.27: Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 12).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 301	61	284	252	0	0	0	1	0	1	7	38	67	12	6
Enemy 302	61	306	310	2	0	0	0	0	0	7	6	10	19	20
Enemy 303	61	121	99	4	0	0	0	0	1	12	34	7	22	12
Enemy 304	61	274	272	4	0	1	0	0	1	28	29	26	5	20
Enemy 305	62	215	208	0	1	0	0	1	1	16	32	47	10	14
Enemy 306	62	92	72	0	1	0	0	0	0	5	4	11	6	26
Enemy 307	62	249	251	2	0	0	1	0	0	29	35	7	10	28
Enemy 308	62	283	281	4	0	0	1	0	1	17	13	12	31	22
Enemy 309	62	186	422	1	0	2	0	0	1	11	8	25	30	24
Enemy 310	62	296	59	3	1	0	1	0	1	26	22	10	34	13
Enemy 311	62	75	46	4	0	0	0	1	0	27	6	15	7	12
Enemy 312	63	324	43	4	1	0	0	1	1	32	26	23	21	21
Enemy 313	63	400	265	0	1	1	1	0	0	6	31	54	21	30
Enemy 314	63	277	180	0	0	0	0	1	0	9	5	13	33	3
Enemy 315	63	46	118	0	1	1	1	0	0	28	16	16	32	25
Enemy 316	63	206	206	2	1	0	0	1	1	33	15	7	24	26
Enemy 317	63	102	32	4	1	0	1	1	0	8	18	7	7	12
Enemy 318	64	299	43	4	0	0	2	1	0	11	9	24	22	11
Enemy 319	64	132	139	2	0	0	1	1	0	4	7	18	29	30
Enemy 320	64	323	314	4	0	0	0	0	0	20	21	11	32	28
Enemy 321	64	306	297	4	0	0	0	1	1	11	17	7	31	11
Enemy 322	64	230	444	1	2	0	0	0	2	16	5	20	31	29
Enemy 323	65	332	112	0	0	1	1	0	1	22	17	32	29	11
Enemy 324	65	315	303	0	1	0	0	0	1	2	3	14	7	26
Enemy 325	65	250	301	2	0	0	0	1	0	33	28	6	7	19

Tabel 5.28: Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 13).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 326	66	250	253	2	0	1	0	2	0	28	10	8	4	17
Enemy 327	66	293	318	2	0	0	1	0	1	2	25	7	19	15
Enemy 328	66	268	318	0	0	0	0	1	0	23	28	28	30	14
Enemy 329	66	284	137	0	0	0	0	1	0	2	41	84	22	23
Enemy 330	67	179	275	1	0	1	0	0	0	7	32	41	38	4
Enemy 331	67	193	240	2	0	0	1	1	0	30	12	13	2	14
Enemy 332	67	410	208	0	0	0	2	0	2	2	13	21	4	10
Enemy 333	67	580	286	3	0	0	0	0	0	31	38	11	16	29
Enemy 334	68	187	179	3	0	1	0	1	0	14	23	18	24	23
Enemy 335	68	296	595	1	1	2	0	0	0	14	34	34	13	16
Enemy 336	68	243	358	1	1	1	0	0	1	21	38	43	24	4
Enemy 337	69	198	192	4	0	0	0	1	0	14	9	9	36	15
Enemy 338	69	104	337	2	1	0	0	0	0	19	19	27	32	22
Enemy 339	69	301	341	0	0	0	0	1	1	31	15	12	39	20
Enemy 340	70	250	244	4	0	0	0	0	0	8	43	4	15	18
Enemy 341	70	307	298	0	0	0	1	0	1	34	7	10	4	9
Enemy 342	70	362	61	4	0	1	2	0	0	4	10	27	23	22
Enemy 343	70	55	333	0	0	0	0	0	0	5	10	2	25	23
Enemy 344	70	356	171	3	0	0	0	0	0	5	21	8	34	13
Enemy 345	70	273	264	0	0	0	2	0	0	8	43	67	2	4
Enemy 346	70	280	231	4	0	0	0	0	0	23	31	24	22	27
Enemy 347	70	222	137	3	0	1	0	0	0	3	41	7	2	12
Enemy 348	71	224	220	0	1	1	0	0	0	30	42	45	3	2
Enemy 349	71	141	186	2	0	1	0	1	1	24	9	27	15	5
Enemy 350	71	310	309	4	0	0	1	1	0	16	23	5	36	26

Tabel 5.29: Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 14).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 351	71	40	350	0	1	0	1	0	1	4	40	7	22	29
Enemy 352	71	240	279	0	0	1	1	0	0	14	29	27	26	23
Enemy 353	71	226	226	2	0	2	0	2	0	2	28	3	19	3
Enemy 354	71	409	120	0	2	0	0	1	0	34	43	54	27	11
Enemy 355	71	83	281	2	0	0	1	1	1	37	39	7	11	14
Enemy 356	71	404	407	2	1	0	1	0	0	20	34	2	21	30
Enemy 357	71	245	236	4	1	0	1	1	0	23	5	16	7	2
Enemy 358	71	153	365	1	1	0	0	0	0	20	8	17	17	16
Enemy 359	72	138	31	4	1	0	1	1	0	33	11	29	37	22
Enemy 360	72	433	158	0	1	1	0	0	1	15	42	81	9	33
Enemy 361	72	41	139	2	0	0	0	1	1	22	35	4	37	12
Enemy 362	72	268	268	2	1	0	1	1	0	2	14	30	22	28
Enemy 363	72	354	58	4	1	0	0	0	1	23	9	27	14	19
Enemy 364	72	195	346	0	1	0	1	1	0	4	32	29	6	30
Enemy 365	72	220	25	4	0	2	2	0	0	2	6	28	26	27
Enemy 366	72	152	155	2	0	1	1	0	0	3	33	3	30	25
Enemy 367	72	341	252	4	0	0	0	1	1	15	4	29	7	33
Enemy 368	73	138	132	4	0	0	0	2	2	17	12	7	18	29
Enemy 369	73	229	346	2	0	0	0	0	1	14	19	9	5	8
Enemy 370	73	217	387	0	1	0	0	0	0	33	42	4	6	22
Enemy 371	73	257	260	2	0	1	0	0	0	33	25	8	5	23
Enemy 372	74	61	127	0	1	0	1	0	0	36	46	8	33	35
Enemy 373	74	332	620	0	1	0	0	1	0	33	30	23	16	24
Enemy 374	74	194	155	0	0	0	1	0	2	26	19	26	29	12
Enemy 375	74	345	337	0	2	0	2	0	0	18	6	11	38	17

Tabel 5.30: Hasil keseluruhan data atribut *gameplay* pada musuh (Bag. 15).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 376	74	295	289	0	0	1	0	0	1	15	9	12	18	35
Enemy 377	75	268	501	1	2	0	0	2	0	7	35	38	12	26
Enemy 378	75	323	228	0	1	0	0	0	1	9	25	32	11	6
Enemy 379	75	374	366	4	0	0	0	1	0	19	28	23	42	33
Enemy 380	75	371	34	0	0	0	1	0	1	26	10	18	30	29
Enemy 381	75	60	58	4	1	0	1	1	0	9	35	27	29	24
Enemy 382	75	178	223	0	0	1	0	0	0	38	22	7	15	26
Enemy 383	76	294	289	4	1	1	0	1	0	4	5	30	38	15
Enemy 384	76	222	62	4	0	1	0	0	1	9	41	12	39	11
Enemy 385	76	579	303	3	2	0	2	0	0	11	38	24	12	7
Enemy 386	76	197	290	0	0	0	0	0	1	12	18	5	28	36
Enemy 387	76	154	292	1	0	1	0	0	0	33	39	40	22	4
Enemy 388	76	177	162	0	1	1	1	0	0	24	28	31	36	29
Enemy 389	76	433	268	0	1	0	0	1	1	20	39	65	15	13
Enemy 390	77	300	400	1	0	1	1	0	0	17	18	26	21	9
Enemy 391	77	427	113	0	0	0	1	0	0	10	39	53	19	11
Enemy 392	77	222	40	3	1	1	0	0	1	29	35	30	13	3
Enemy 393	77	467	296	3	0	2	0	0	0	21	44	14	31	4
Enemy 394	77	350	359	2	0	1	1	0	1	2	7	14	6	12
Enemy 395	77	475	358	0	1	0	0	1	0	3	36	43	4	23
Enemy 396	78	300	199	3	0	2	0	1	0	27	29	12	23	5
Enemy 397	78	267	441	0	0	1	0	0	1	37	29	17	12	25
Enemy 398	78	146	201	0	0	0	0	0	1	31	23	19	11	10
Enemy 399	78	75	261	2	0	0	1	0	0	22	30	2	43	15
Enemy 400	78	63	337	0	0	1	0	0	0	24	41	33	20	34

Tabel 5.31: Dataset atribut *gameplay hero* Dota 2 untuk *training* (Bag. 1).

name	type	baseStr	strGrowth	maxStr	baseAgi	agiGrowth	maxAgi	baseInt	intGrowth	maxInt	totalBaseAttr	totalAttrGrowth	totalMaxAttr	moveSpeed	baseArmor	minDmg	maxDmg	range	baseAttackTime	attackPoint	attackBackswing	turnRate	regeneration
Abaddon	0	23	2.6	85.4	17	1.5	53	21	2	69	61	6.1	207.4	310	1.43	55	65	150	1.7	0.56	0.41	0.6	1.5
Ancient Apparition	2	18	1.7	58.8	20	2.2	72.8	25	2.6	87.4	63	6.5	219	295	1.86	44	54	675	1.7	0.45	0.3	0.6	1.5
Anti-Mage	1	22	1.5	58	22	2.8	89.2	15	1.8	58.2	59	6.1	205.4	310	2.14	49	53	150	1.45	0.3	0.6	0.5	1.5
Axe	0	25	2.8	92.2	20	2.2	72.8	18	1.6	56.4	63	6.6	221.4	290	1.86	49	53	150	1.7	0.5	0.5	0.6	4.25
Bane	2	23	2.4	80.6	23	2.4	80.6	23	2.4	80.6	69	7.2	241.8	310	4.29	60	66	400	1.7	0.3	0.7	0.6	1.5
Batrider	2	23	2.7	87.8	15	1.5	51	24	2.5	84	62	6.7	222.8	290	2.14	38	42	375	1.7	0.3	0.54	1	3.25
Beastmaster	0	23	2.5	83	18	1.6	56.4	16	1.9	61.6	57	6	201	310	4.57	64	68	150	1.7	0.3	0.7	0.5	1.5
Bounty Hunter	1	17	2.1	67.4	21	3	93	19	2	67	57	7.1	227.4	315	6	45	59	150	1.7	0.59	0.59	0.6	2
Bristleback	0	22	2.2	74.8	17	1.8	60.2	14	2.8	81.2	53	6.8	216.2	290	3.43	44	54	150	1.8	0.3	0.3	1	1.5
Broodmother	1	17	2.8	84.2	18	2.2	70.8	18	2	66	53	7	221	270	2.57	44	50	150	1.7	0.4	0.5	0.5	1.5
Centaur Warrunner	0	23	4.3	126.2	15	1.6	53.4	15	1.6	53.4	53	7.5	233	300	2.14	55	57	150	1.7	0.3	0.3	0.5	1.5
Chaos Knight	0	22	3.2	98.8	14	2.1	64.4	16	1.2	44.8	52	6.5	208	325	4	51	81	150	1.7	0.5	0.5	0.5	1.5
Chen	2	23	1.8	66.2	15	2.1	65.4	21	2.8	88.2	59	6.7	219.8	300	1.14	43	53	650	1.7	0.5	0.5	0.6	1.5
Clinkz	1	15	1.9	60.6	22	2.6	84.4	16	1.55	53.2	53	6.05	198.2	295	2.14	37	43	640	1.7	0.7	0.3	0.5	1.5
Clockwerk	0	26	3.2	102.8	13	2.3	68.2	17	1.3	48.2	56	6.8	219.2	315	1.86	54	56	150	1.7	0.33	0.64	0.6	1.5
Dark Seer	2	22	2.6	84.4	12	1.2	40.8	23	2.7	87.8	57	6.5	213	295	6.71	54	60	150	1.7	0.59	0.58	0.6	1.5
Dark Willow	2	18	1.8	61.2	18	1.6	56.4	20	3	92	56	6.4	209.6	295	1.57	45	53	475	1.3	0.3	0	0.7	1.5
Dazzle	2	16	2.15	67.6	21	1.7	61.8	27	3.4	108.6	64	7.25	238	305	2	41	59	550	1.7	0.3	0.3	0.6	1.5
Death Prophet	2	17	2.6	79.4	14	1.4	47.6	23	3	95	54	7	222	310	3	47	59	600	1.7	0.56	0.51	0.5	2
Disruptor	2	19	2.2	71.8	15	1.4	48.6	22	2.5	82	56	6.1	202.4	300	1.14	49	53	600	1.7	0.4	0.5	0.5	1.5
Doom	0	26	3.5	110	11	0.9	32.6	13	2.1	63.4	50	6.5	206	290	0.57	53	69	150	2	0.5	0.7	0.5	1.5
Dragon Knight	0	19	3.1	93.4	19	2.2	71.8	15	1.7	55.8	53	7	221	290	3.71	50	56	150	1.7	0.5	0.5	0.6	1.5
Drow Ranger	1	17	1.9	62.6	26	1.9	71.6	15	1.4	48.6	58	5.2	182.8	285	0.71	40	51	625	1.7	0.7	0.3	0.7	1.5
Earth Spirit	2	21	3.2	97.8	17	1.5	53	18	2.1	68.4	56	6.8	219.2	290	3.43	46	56	150	1.7	0.35	0.65	0.6	1.5
Earthshaker	2	22	3.2	98.8	12	1.4	45.6	16	1.8	59.2	50	6.4	203.6	310	2.71	46	56	150	1.7	0.467	0.863	0.9	2
Elder Titan	2	24	2.6	86.4	14	1.8	57.2	23	1.6	61.4	61	6	205	315	3	47	57	150	1.7	0.35	0.97	0.5	1.5
Ember Spirit	1	19	2.1	69.4	22	1.8	65.2	20	1.8	63.2	61	5.7	197.8	305	1.14	52	56	150	1.7	0.4	0.3	0.6	2
Enchantress	2	16	1.3	47.2	19	1.8	62.2	21	3.1	95.4	56	6.2	204.8	340	0.71	52	62	550	1.7	0.3	0.7	0.5	1.5
Enigma	2	17	2.4	74.6	14	1	38	20	3.4	101.6	51	6.8	214.2	300	4	42	48	500	1.7	0.4	0.77	0.5	1.5
Faceless Void	1	23	2.1	73.4	23	2.8	90.2	15	1.5	51	61	6.4	214.6	295	3.29	56	62	150	1.7	0.5	0.56	1	2
Gyrocopter	1	18	2.1	68.4	24	2.8	91.2	19	2.1	69.4	61	7	229	320	4.43	37	47	365	1.7	0.2	0.97	0.6	1.5
Huskar	0	21	2.7	85.8	15	1.4	48.6	18	1.5	54	54	5.6	188.4	300	1.14	42	51	400	1.6	0.4	0.5	0.5	1.5
Juggernaut	1	20	2.2	72.8	26	2.4	83.6	14	1.4	47.6	60	6	204	295	3.71	48	52	150	1.4	0.33	0.84	0.6	2
Keeper of the Light	2	14	2.1	61.4	15	1.6	53.4	25	2.8	92.2	54	6.5	210	335	1.14	43	50	600	1.7	0.3	0.85	0.5	1.5
Kunkka	0	24	3.3	103.2	14	1.3	45.2	18	1.5	54	56	6.1	202.4	300	4	50	60	150	1.7	0.4	0.3	0.6	1.5
Legion Commander	0	26	2.9	95.6	18	1.7	58.8	20	2.2	72.8	64	6.8	227.2	320	2.57	61	65	150	1.7	0.46	0.64	0.5	1.5
Lich	2	18	1.85	62.4	15	2	63	18	3.25	96	51	7.1	221.4	315	1.14	42	51	550	1.7	0.46	0.54	0.5	1.5
Lifestealer	0	25	3.1	99.4	18	1.9	63.6	15	1.75	57	58	6.75	220	315	1.57	52	62	150	1.85	0.29	0.44	1	1.5
Lina	2	18	1.8	61.2	16	1.5	52	27	3.2	103.8	61	6.5	217	295	1.29	40	58	670	1.6	0.75	0.78	0.5	1.5
Lion	2	16	2	64	15	1.5	51	20	3	92	51	6.5	207	290	1.14	47	53	600	1.7	0.43	0.74	0.5	1.5
Lone Druid	1	17	2.4	74.6	24	2.7	88.8	13	1.4	46.6	54	6.5	210	320	3.43	42	46	550	1.7	0.33	0.53	0.5	1.75
Luna	1	15	2.5	75	18	3.3	97.2	16	1.85	60.4	49	7.65	232.6	330	2.57	38	44	330	1.7	0.46	0.54	0.6	1.5
Lycan	0	25	3.3	104.2	16	1	40	17	1.55	54.2	58	5.85	198.4	305	3.29	61	66	150	1.7	0.55	0.55	0.5	1.5
Magnus	0	21	3.2	97.8	15	2.5	75	19	1.65	58.6	55	7.35	231.4	310	4.14	53	65	150	1.8	0.5	0.84	0.8	2
Medusa	1	14	1.95	60.8	20	2.5	80	19	2.1	69.4	53	6.55	210.2	285	1.86	44	50	600	1.7	0.5	0.6	0.5	1.5
Meepo	1	23	1.6	61.4	23	2.2	75.8	20	1.6	58.4	66	5.4	195.6	310	2.29	43	49	150	1.7	0.38	0.6	0.65	1.5
Mirana	1	17	2.15	68.6	20	3.6	106.4	17	1.65	56.6	54	7.4	231.6	295	1.86	41	52	630	1.7	0.3	0.7	0.5	1.5
Monkey King	1	18	2.8	85.2	22	3.2	98.8	20	1.8	63.2	60	7.8	247.2	300	0.14	52	58	300	1.7	0.45	0.2	0.6	3
Morphling	1	19	2.3	74.2	24	3.7	112.8	17	1.1	43.4	60	7.1	230.4	280	1.43	33	42	350	1.5	0.5	0.5	0.6	1.5

Tabel 5.32: Dataset atribut *gameplay hero* Dota 2 untuk *training* (Bag. 2).

name	type	baseStr	strGrowth	maxStr	baseAgi	agiGrowth	maxAgi	baseInt	intGrowth	maxInt	totalBaseAttr	totalAttrGrowth	totalMaxAttr	moveSpeed	baseArmor	minDmg	maxDmg	range	baseAttackTime	attackPoint	attackBackswing	turnRate	regeneration
Naga Siren	1	21	2.8	88.2	21	2.75	87	21	2	69	63	7.55	244.2	315	6	44	46	150	1.7	0.5	0.5	0.5	3
Nature's Prophet	2	19	2.1	69.4	18	1.9	63.6	25	2.9	94.6	62	6.9	227.6	290	3.57	55	69	600	1.7	0.4	0.77	0.6	1.5
Necrophos	2	16	2.3	71.2	15	1.2	43.8	22	2.5	82	53	6	197	285	3.14	44	48	550	1.7	0.53	0.47	0.5	1.5
Night Stalker	0	23	3.1	97.4	18	2.25	72	13	1.6	51.4	54	6.95	220.8	295	5.57	61	65	150	1.7	0.55	0.55	0.5	3.25
Nyx Assassin	1	18	2.3	73.2	19	2.2	71.8	18	2.1	68.4	55	6.6	213.4	295	3.71	46	50	150	1.7	0.46	0.54	0.5	4.75
Ogre Magi	2	23	3.5	107	14	1.55	51.2	17	2	65	54	7.05	223.2	285	8	58	64	150	1.7	0.3	0.3	0.6	3.25
Omniknight	0	22	3.1	96.4	15	1.75	57	17	1.8	60.2	54	6.65	213.6	305	5.14	53	63	150	1.7	0.433	0.567	0.6	1.5
Oracle	2	18	2.2	70.8	15	1.7	55.8	23	2.9	92.6	56	6.8	219.2	305	2.14	39	45	620	1.4	0.3	0.7	0.5	1.5
Pangolier	1	16	2.5	76	18	2.8	85.2	16	1.9	61.6	50	7.2	222.8	305	3.57	47	53	150	1.7	0.33	0	1	1.5
Phantom Assassin	1	20	2.15	71.6	23	3.15	98.6	15	1.4	48.6	58	6.7	218.8	305	4.29	46	48	150	1.7	0.3	0.7	0.6	1.5
Puck	2	15	2	63	22	1.7	62.8	25	2.4	82.6	62	6.1	208.4	295	1.14	53	64	550	1.7	0.5	0.8	0.5	1.5
Pudge	0	25	3.5	109	14	1.5	50	14	1.5	50	53	6.5	209	280	1	52	58	150	1.7	0.5	1.17	0.7	1.5
Pugna	2	17	1.5	53	16	1	40	26	4.5	134	59	7	227	335	1.29	45	53	630	1.7	0.5	0.5	0.5	1.5
Queen of Pain	2	16	2	64	18	2	66	24	2.5	84	58	6.5	214	295	1.57	45	53	550	1.5	0.56	0.41	0.5	1.5
Razor	1	21	2.6	83.4	22	1.8	65.2	21	1.8	64.2	64	6.2	212.8	285	2.14	45	47	475	1.7	0.3	0.7	0.5	1.5
Riki	1	17	1.9	62.6	34	2.2	86.8	14	1.3	45.2	65	5.4	194.6	275	4.86	38	42	150	1.7	0.3	0.3	0.6	1.5
Rubick	2	19	1.8	62.2	14	1.6	52.4	27	2.4	84.6	60	5.8	199.2	290	1	44	54	550	1.7	0.4	0.77	0.7	1.5
Sand King	0	22	2.9	91.6	19	2.1	69.4	16	1.8	59.2	57	6.8	220.2	295	2.71	47	63	150	1.7	0.53	0.47	0.5	1.5
Shadow Demon	2	21	2.2	73.8	18	2.2	70.8	23	2.7	87.8	62	7.1	232.4	295	2.57	50	54	500	1.7	0.35	0.5	0.6	1.5
Shadow Fiend	1	15	2.3	70.2	20	2.9	89.6	18	2	66	53	7.2	225.8	310	0.86	35	41	500	1.7	0.5	0.54	1	1.75
Shadow Shaman	2	21	2.1	71.4	16	1.6	54.4	21	3	93	58	6.7	218.8	285	2.29	71	78	400	1.7	0.3	0.5	0.5	1.5
Silencer	2	17	2.5	77	22	3	94	27	2.5	87	66	8	258	295	2.14	43	57	600	1.7	0.5	0.5	0.6	1.5
Skywrath Mage	2	19	1.8	62.2	13	0.8	32.2	27	3.6	113.4	59	6.2	207.8	330	-0.14	39	49	600	1.7	0.4	0.78	0.5	1.5
Slardar	0	21	3.1	95.4	17	2.4	74.6	15	1.5	51	53	7	221	290	5.43	51	59	150	1.7	0.36	0.64	0.5	1.5
Slark	1	20	1.9	65.6	21	1.5	57	16	1.7	56.8	57	5.1	179.4	295	2	54	62	150	1.7	0.5	0.3	0.5	1.5
Sniper	1	16	2	64	21	2.7	85.8	15	2.6	77.4	52	7.3	227.2	285	2	36	42	550	1.7	0.17	0.7	0.7	1.5
Spectre	1	20	2.3	75.2	23	1.8	66.2	16	1.9	61.6	59	6	203	285	3.29	46	50	150	1.7	0.3	0.7	0.5	1.5
Spirit Breaker	0	29	2.7	93.8	17	1.7	57.8	14	1.8	57.2	60	6.2	208.8	285	5.43	60	70	150	1.9	0.6	0.3	0.5	2
Storm Spirit	2	19	1.8	62.2	22	1.8	65.2	24	3	96	65	6.6	223.4	285	5.14	46	56	480	1.7	0.5	0.3	0.8	1.5
Sven	0	23	3	95	21	2	69	16	1.3	47.2	60	6.3	211.2	290	5	61	66	150	1.8	0.4	0.3	0.6	1.5
Techies	2	17	2.3	72.2	14	1.3	45.2	22	2.9	91.6	53	6.5	209	270	7	29	31	700	1.7	0.5	0.5	0.5	1.5
Templar Assassin	1	18	2.4	75.6	23	2.3	78.2	20	2	68	61	6.7	221.8	300	4.29	53	59	140	1.7	0.3	0.5	0.7	1.5
Terrorblade	1	15	1.7	55.8	22	3.2	98.8	19	1.75	61	56	6.65	215.6	310	10.14	48	54	150	1.5	0.3	0.6	0.5	3.25
Tidehunter	0	22	3.3	101.2	15	1.5	51	16	1.7	56.8	53	6.5	209	305	3.14	47	53	150	1.7	0.6	0.56	0.5	1.5
Timbersaw	0	21	2.1	71.4	16	1.3	47.2	21	2.4	78.6	58	5.8	197.2	290	0.29	47	51	150	1.7	0.36	0.64	0.6	1.5
Tinker	2	17	2.3	72.2	13	1.2	41.8	30	2.2	82.8	60	5.7	196.8	305	3.86	52	58	500	1.7	0.35	0.65	0.6	1.5
Tiny	0	26	3.3	105.2	0	0	0	17	1.6	55.4	43	4.9	160.6	285	0	70	76	150	1.7	0.4	1	0.5	2.5
Treant Protector	0	25	3.6	111.4	15	2	63	17	1.8	60.2	57	7.4	234.6	270	1.14	87	95	150	1.9	0.6	0.4	0.5	1.5
Troll Warlord	1	20	2.5	80	21	2.5	81	13	1	37	54	6	198	295	2	38	56	500	1.7	0.3	0.3	0.5	1.5
Underlord	0	25	2.9	94.6	12	1.3	43.2	17	2.6	79.4	54	6.8	217.2	290	3.71	62	68	150	1.7	0.45	0.7	0.6	1.5
Ursa	1	23	3	95	18	2.1	68.4	16	1.5	52	57	6.6	215.4	305	5.57	42	46	150	1.7	0.3	0.3	0.5	2
Vengeful Spirit	1	18	2.9	87.6	27	3.3	106.2	13	1.5	49	58	7.7	242.8	295	3.86	39	53	400	1.7	0.33	0.64	0.6	1.5
Venomancer	1	18	1.9	63.6	22	2.8	89.2	17	1.8	60.2	57	6.5	213	270	3.14	38	40	450	1.7	0.3	0.7	0.5	1.5
Viper	1	20	2.4	77.6	21	2.9	90.6	15	1.8	58.2	56	7.1	226.4	275	4.5	44	46	575	1.7	0.33	1	0.5	1.5
Visage	2	22	2.7	86.8	11	1.3	42.2	24	2.5	84	57	6.5	213	285	-0.43	45	55	600	1.7	0.46	0.54	0.5	1.5
Warlock	2	22	2.8	89.2	10	1	34	24	2.7	88.8	56	6.5	212	295	2.43	46	56	600	1.7	0.3	0.3	0.5	1.5
Windranger	2	15	2.8	82.2	17	1.4	50.6	22	2.6	84.4	54	6.8	217.2	295	1.43	44	56	600	1.5	0.4	0.3	0.6	1.5
Witch Doctor	2	16	2.1	66.4	13	1.4	46.6	24	2.9	93.6	53	6.4	206.6	305	0.86	51	61	600	1.7	0.4	0.5	0.5	1.5
Wraith King	0	22	2.8	89.2	18	1.7	58.8	18	1.6	56.4	58	6.1	204.4	300	2.57	61	63	150	1.7	0.56	0.44	0.5	1.5
Zeus	2	19	2.6	81.4	11	1.2	39.8	22	2.7	86.8	52	6.5	208	300	1.57	43	51	350	1.7	0.633	0.366	0.6	1.5

Tabel 5.33: Dataset atribut *gameplay hero* Dota 2 untuk *testing*.

name	type	baseStr	strGrowth	maxStr	baseAgi	agiGrowth	maxAgi	baseInt	intGrowth	maxInt	totalBaseAttr	totalAttrGrowth	totalMaxAttr	moveSpeed	baseArmor	minDmg	maxDmg	range	baseAttackTime	attackPoint	attackBackswing	turnRate	regeneration
ArcWarden	1	24	3	96	15	1.8	58.2	24	2.6	86.4	63	7.4	240.6	280	0.14	44	54	625	1.7	0.3	0.7	0.6	1.5
Jakiro	2	25	2.6	87.4	10	1.2	38.8	28	2.8	95.2	63	6.6	221.4	290	2.43	53	61	400	1.7	0.4	0.5	0.5	1.5
WinterWyvern	2	24	2.4	81.6	16	1.9	61.6	25	3.1	99.4	65	7.4	242.6	285	1.29	38	45	425	1.7	0.25	0.8	0.5	1.5
Bloodseeker	1	23	2.7	87.8	24	3	96	18	1.7	58.8	65	7.4	242.6	285	3.43	57	63	150	1.7	0.43	0.74	0.5	1.5
Phoenix	0	19	3.2	95.8	12	1.3	43.2	18	1.8	61.2	49	6.3	200.2	285	-0.29	45	55	500	1.7	0.35	0.633	1	1.5
Io	0	17	2.2	69.8	14	1.6	52.4	23	1.7	63.8	54	5.5	186	295	0	43	52	575	1.7	0.15	0.4	0.7	1.5
Leshrac	2	16	1.8	59.2	23	1.7	63.8	26	3	98	65	6.5	221	325	3.29	41	45	600	1.7	0.4	0.77	0.5	1.5
OutworldDevourer	2	19	2.6	81.4	24	2	72	26	2.7	90.8	69	7.3	244.2	315	3.93	40	55	450	1.7	0.46	0.54	0.5	1.5
Brewmaster	0	23	3.2	99.8	22	1.95	68.8	14	1.25	44	59	6.4	212.6	300	2.14	52	59	150	1.7	0.35	0.65	0.6	2
Tusk	0	23	2.6	85.4	23	2.1	73.4	18	1.7	58.8	64	6.4	217.6	300	3.29	50	54	150	1.7	0.36	0.64	0.7	1.5
Weaver	1	15	1.8	58.2	14	2.8	81.2	15	1.8	58.2	44	6.4	197.6	280	1	50	60	425	1.8	0.64	0.36	0.5	1.5
Undying	0	22	2.4	79.6	10	0.8	29.2	27	2.8	94.2	59	6	203	310	4.43	57	65	150	1.7	0.3	0.3	0.6	1.5
Alchemist	0	25	2.1	75.4	16	1.2	44.8	25	1.8	68.2	66	5.1	188.4	295	2.29	49	58	150	1.7	0.35	0.65	0.6	1.5
Phantom Lancer	1	21	2	69	29	2.8	96.2	21	2	69	71	6.8	234.2	285	4.14	51	73	150	1.7	0.5	0.5	0.6	3
CrystalMaiden	2	16	2	64	16	1.6	54.4	16	2.9	85.6	48	6.5	204	275	1.29	35	41	600	1.7	0.55	0	0.5	1.5
Invoker	2	16	2.2	68.8	14	1.9	59.6	16	4	112	46	8.1	240.4	280	1	35	41	600	1.7	0.4	0.7	0.5	1.5

Tabel 5.34: Dataset karakter pemain untuk *training*.

name	type	minHP	maxHp	hpGrowth	minMP	maxMp	mpGrowth	minStr	maxStr	strGrowth	minMag	maxMag	magGrowth	minEnd	maxEnd	endGrowth	minSpd	maxSpd	spdGrowth	minLuck	maxLuck	luckGrowth
Knight 1	0	224	620	4	100	298	2	1	88	0.87	1	32	0.31	1	81	0.8	1	43	0.42	1	56	0.55
Priest 1	1	223	520	3	154	550	4	1	60	0.59	1	68	0.67	1	57	0.56	1	58	0.57	1	57	0.56
Assassin 1	2	173	470	3	127	424	3	1	49	0.48	1	55	0.54	1	48	0.47	1	72	0.71	1	76	0.75
Magician 1	3	203	500	3	134	530	4	1	56	0.55	1	72	0.71	1	53	0.52	1	62	0.61	1	57	0.56
Knight 2	0	283	580	3	130	427	3	1	75	0.74	1	45	0.44	1	85	0.84	1	45	0.44	1	58	0.57
Priest 2	1	193	490	3	160	556	4	1	50	0.49	1	70	0.69	1	60	0.59	1	62	0.61	1	60	0.59

Tabel 5.35: Dataset karakter pemain untuk *testing*.

name	type	minHP	maxHp	hpGrowth	minMP	maxMp	mpGrowth	minStr	maxStr	strGrowth	minMag	maxMag	magGrowth	minEnd	maxEnd	endGrowth	minSpd	maxSpd	spdGrowth	minLuck	maxLuck	luckGrowth
Assassin 2	2	177	573	4	121	418	3	1	75	0.74	1	47	0.46	1	63	0.62	1	64	0.63	1	67	0.66
Magician 2	3	220	517	3	150	546	4	1	41	0.4	1	74	0.73	1	55	0.54	1	53	0.52	1	52	0.51

Halaman ini sengaja dikosongkan

Daftar Pustaka

Bangay, S. and Makin, O. (2014). Generating an attribute space for analyzing balance in single unit RTS game combat. IEEE Conference on Computatonal Intelligence and Games, CIG, pages 1–8. (Dikutip pada halaman 3).

Barton, M. and Stacks, S. (2019). Dungeons and Desktops: The History of Computer Role-Playing Games 2e. CRC Press. (Dikutip pada halaman 33).

Brathwaite, B. and Schreiber, I. (2009). Challenges for game designers. Course Technology. (Dikutip pada halaman 9, 11, 14).

Buduma, N. and Locascio, N. (2017). Fundamentals of deep learning: Designing next-generation machine intelligence algorithms. "O'Reilly Media, Inc.". (Dikutip pada halaman 41, 43).

Camelo, J. H., Rabêlo, R. L., and Passos, E. (2014). Automatic mapping between gameplay and aesthetics rpg character attributes through fuzzy system. In 2014 Brazilian Symposium on Computer Games and Digital Entertainment, pages 220–229. (Dikutip pada halaman 2, 3).

Christyowidiasmoro, Putra, R. C. A., and Susiki, S. M. (2016). Measuring level of difficulty in game using challenging rate (CR) on 2D Real time Strategy Line Defense game. Proceedings - 2015 International Electronics Symposium: Emerging Technology in Electronic and Information, IES 2015. (Dikutip pada halaman 3).

Cover, T. M. and Hart, P. E. (1967). Nearest Neighbor Pattern Classification. IEEE Transactions on Information Theory. (Dikutip pada halaman 18).

Fandom (2020). Table of Hero Attribute. `https://dota2.gamepedia.com/Table_of_hero_attributes` [Terakhir diakses pada tanggal 12 Juni 2020]. (Dikutip pada halaman 94, 95).

Friedman, H. J. (1997). Data mining and statistics: What's the connection? Department of Statistics and Standford Linear Accelerator Center, Standford University. (Dikutip pada halaman 16).

Heinsoo, R., Collins, A., and Wyatt, J. (2008). Dungeons & Dragons player's handbook: Arcane, Divine, and Martial Heroes. Wizards of the Coast. (Dikutip pada halaman 2, 50).

Koster, R. (2013). Theory of fun for game design. "O'Reilly Media, Inc.". (Dikutip pada halaman 11).

Koza, J. R., Bennett, F. H., Andre, D., and Keane, M. A. (1996). Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming. Springer Netherlands. (Dikutip pada halaman 16).

Marín-Lora, C., Chover, M., Sotoca, J. M., and García, L. A. (2020). A game engine to make games as multi-agent systems. Advances in Engineering Software, 140(April 2019):102732. (Dikutip pada halaman 3).

Moore, M. (2016). Basics of game design. CRC Press. (Dikutip pada halaman 38, 39, 40).

Nick, Y. (1999-2009). "The Daedalus Project" the psychology of MMORPGs. `http://www.nickyee.com/daedalus` [Terakhir diakses pada tanggal 12 Juni 2020]. (Dikutip pada halaman 40).

Panumate, C., Xiong, S., and Iida, H. (2015). An approach to quantifying Pokemon's entertainment impact with focus on battle. Proceedings - 3rd International Conference on Applied Computing and Information Technology and 2nd International Conference on Computational Science and Intelligence, ACIT-CSI 2015, pages 60–66. (Dikutip pada halaman 32, 36).

Schuller, D. (2017). How to Make an RPG. `http://howtomakeanrpg.com/a/how-to-make-an-rpg-stats.html` [Terakhir diakses pada tanggal 9 Agustus 2020]. (Dikutip pada halaman 2, 50).

Stenström, C. D. (2012). Gameplay design for role-playing battle systems. Master's thesis, Department of Computer Science and Engineering, Chalmers University of Technology, SE - 412 96, Ǵoteborg, Sweden. (Dikutip pada halaman 1, 33).

Wenz, K. (2013). Theorycrafting: Knowledge production and surveillance. Information, Communication & Society, 16(2):178–193. (Dikutip pada halaman 2).

Wu, Z. H., Lai, K., Lin, L. A., Huang, M. H., and Tai, W. K. (2018). Procedurally Generating Game Level with Specified Difficulty. 2018 IEEE Games, Entertainment, Media Conference, GEM 2018, pages 71–78. (Dikutip pada halaman 3).

BIOGRAFI PENULIS

Nur Rohman Widiyanto, lahir pada 26 Maret 1991 di Lamongan, Jawa Timur. Penulis menempuh pendidikan Sarjana di Jurusan Teknik Elektro, FTI, ITS angkatan 2009, kemudian mengambil bidang studi Teknik Komputer dan Telematika. Saat di kuliah penulis aktif menjadi asisten laboratorium Telematika (B201) dan pernah menjabat sebagai koordiator asisten Lab B201 periode 2011/2012. Selama masa kuliah penulis aktif dalam mengikuti ajang perlombaan seperti PKM (Program Kreatifitas Mahasiswa), aktif dalam *development group networking* dan juga sebagai *administrator* jaringan Lab B201. Kemudian penulis menyelesaikan penelitian atau Tugas Akhir dengan judul "Pengukuran Performansi Mesin Virtual pada Komputasi Awan". Di sisilain penulis sangat tertarik dengan segala hal yang berhubungan dengan komputer, dan berencana mendalami cabang ilmu komputer lain selain jaringan komputer. Pada tahun 2017 penulis resmi melanjutkan studi Master di bidang keahlian Jaringan Cerdas Multimedia, Jurusan Teknik Elektro, FTEIC, ITS. Sebelum resmi melanjutkan studi, penulis pernah bekerja sebagai *Lead Network Engineer* di PT. Niltava Teknologi Indonesia. Selama menempuh studi Master penulis memfokuskan diri dalam penelitian tentang kecerdasan buatan, maka dibuatlah penelitian dengan judul "Perhitungan Kenaikan Atribut *Gameplay* untuk Pemain dan *Non-Player Character* pada Permainan *Role-Playing Game* berbasis K-NN dan Naive Bayes". Penulis dapat dihubungi melalui email: *rohwid@gmail.com* dan ponsel: *+6285731925725*.

Halaman ini sengaja dikosongkan