TUGAS AKHIR – IF184802

REKOMENDASI ANOTASI OTOMATIS PADA KONTEN PEMBELAJARAN MOOC

PURINA QUROTA AYUNIN
NRP 05111540000008

Dosen Pembimbing 1
Nurul Fajrin Ariyani, S.Kom.,M.Sc.

Dosen Pembimbing 2
Abdul Munif, S.Kom.,M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – IF184802

REKOMENDASI ANOTASI OTOMATIS PADA KONTEN PEMBELAJARAN MOOC

PURINA QUROTA AYUNIN
NRP 05111540000008

Dosen Pembimbing 1
Nurul Fajrin Ariyani, S.Kom.,M.Sc.

Dosen Pembimbing 2
Abdul Munif, S.Kom.,M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – IF184802

AUTOMATIC ANOTATION RECOMMENDATIONS IN MOOC LEARNING CONTENTS

PURINA QUROTA AYUNIN
NRP 05111540000008

Supervisors 1
Nurul Fajrin Ariyani, S.Kom.,M.Sc.

Supervisors 2
Abdul Munif, S.Kom.,M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

REKOMENDASI ANOTASI OTOMATIS PADA KONTEN PEMBELAJARAN MOOC

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Manajemen Informasi
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
PURINA QUROTA AYUNIN
NRP. 05111540000008

Disetujui oleh Pembimbing Tugas Akhir:

1. Nurul Fajrin A., S.Kom., M.Sc. .....
   NIP. 19860722 201504 2 003 (Pembimbing 1)

2. Abdul Munif., S.Kom., M.Sc.Eng. .....
   NIP. 19860823 201504 1 004 (Pembimbing 2)

SURABAYA
JANUARI 2019

[Halaman ini sengaja dikosongkan]

REKOMENDASI ANOTASI OTOMATIS PADA KONTEN PEMBELAJARAN MOOC

Nama Mahasiswa	: Purina Qurota Ayunin
NRP	: 05111540000008
Departemen	: Informatika FTIK-ITS
Dosen Pembimbing 1	: Nurul Fajrin Ariyani, S.Kom.,M.Sc.
Dosen Pembimbing 2	: Abdul Munif, S.Kom.,M.Sc.

ABSTRAK

Saat ini proses belajar mengajar perkuliahan dapat dilakukan hanya dengan mengikuti kelas secara online melalui situs-situs yang menerapkan sistem Massive Open Online Course (MOOC). Namun pada praktiknya, ketika pengajar membuat kursus baru pada situs MOOC, pengajar kurang mendefinisikan secara detail perihal pemberian keterangan kursus. Khususnya pada MOOC yang menggunakan Moodle, pengajar cenderung hanya memberikan materi berupa attachment file atau tugas perminggunya.

Maka pada tugas akhir ini, akan dilakukan pelabelan dengan menganotasikan konten pembelajaran secara otomatis dengan metode pengekstrakan konten pembelajaran yang kemudian diklasifikasikan. Data yang diambil merupakan judul mata kuliah dan deksripsi, capaian pembelajaran, atau pokok bahasan mata kuliah. Kemudian dilakukan proses data mining berupa pengklasifikasian menggunakan metode tanpa Machine Learning dengan menerapkan beberapa rule dan dengan metode Machine Learning dengan Random Forest, Support Vector Machine, dan Naive Bayes.

Dari keempat metode yang diuji, metode dengan hasil terendah didapatkan pada metode klasifikasi tanpa Machine Learning dengan akurasi 71,7%. Sedangkan hasil terbaik diperoleh dari metode menggunakan Machine Learning menggunakan Random Forest Classifier dengan data training yang sudah di-over sampling dengan ADASYN mendapatkan nilai akurasi yaitu

93,3%. Model tersebut juga dikatakan terbaik karena terbukti menghasilkan keluaran label yang sesuai dari data uji baru.
Keywords: MOOC, Naïve Bayes, Random Forest, SVM

AUTOMATIC ANOTATION RECOMMENDATIONS IN MOOC LEARNING CONTENTS

Student Name	: Purina Qurota Ayunin
NRP	: 05111540000008
Departement	: Informatika FTIK-ITS
Supervisor 1	: Nurul Fajrin Ariyani, S.Kom.,M.Sc.
Supervisor 2	: Abdul Munif, S.Kom.,M.Sc.Eng

ABSTRACT

Nowadays, teaching and learning process of lectures can be done only by taking classes online through sites that implement the Massive Open Online Course (MOOC) system. But in practice, when the teacher makes a new course on the MOOC website, the teacher does not define the course in detail in the about the course description. Especially in the MOOC that uses Moodle, teachers tend to only provide material in the form of file attachments or weekly assignments.

In this final project, labeling will be carried out by annotating learning content automatically by extracting the course content which is then classified. The data taken is the subject title and description, learning outcomes, or subject matter. Then the data mining process is carried out in the form of classifying using a method without Machine Learning by applying several rules and using Machine Learning methods with Random Forest, Support Vector Machine, and Naive Bayes.

From the four tested methods, the method with the lowest results was obtained from the classification method without Machine Learning with an accuracy of 71.7%. While the best results are obtained from the method of using Machine Learning, that is using a Random Forest Classifier with training data that has been over-sampled with ADASYN with an accuracy with of 93.3%. The model is also said to be the best because it is proven to produce the appropriate label output from the new test data.

Keywords: MOOC, Naïve Bayes, Random Forest, SVM

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur saya ucapkan kepada Allah Yang Maha Kuasa. Karena atas karunia serta rahmat-Nya saya dapat menyelesaikan tugas akhir yang berjudul:

REKOMENDASI ANOTASI OTOMATIS PADA KONTEN PEMBELAJARAN MOOC

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT Yang Maha Kuasa, karena limpahan rahmat dan karunia-Nya penulis dapat menjalankan perkuliahan di Departemen Informatika Institut Teknologi Sepuluh Nopember dan dapat menyelesaikan Tugas Akhir guna memenuhi syarat kelulusan sebagai Sarjana.
2. Keluarga tercinta, Bapak, Umi, Daus dan Pani yang telah memberikan dukungan moral dan material serta do'a yang tak terhingga untuk penulis. Tak lupa juga kepada anggota keluarga tambahan, yaitu kucing kami Ame, Anis, dan Hiro yang selalu memberikan kebahagiaan kepada penulis.
3. Ibu Nurul Fajrin Ariyani, S.Kom., M.Sc. dan Bapak Abdul Munif, S.Kom., M.Sc. selaku pembimbing I dan II yang telah membimbing, memberi nasihat, serta mengorbankan waktu dan tenaga untuk membimbing saya dalam menyelesaikan Tugas Akhir ini.
4. Dr. Eng. Darlis Herumurti, S.Kom., M.Kom. selaku Ketua Departemen Informatika ITS dan segenap dosen dan karyawan Departemen Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa kuliah di Informatika ITS.
5. Nahda Fauziyah Zahra, Hania Maghfira, dan Anisah Putri Diana sebagai sahabat yang selalu menemani selama hidup sebagai perantau di Surabaya.

6. Rakhma Rufaida Hanum dan Rezky Alamsyah sebagai teman seperjuangan yang saling membantu selama tugas akhir.
7. Muhajir, Fino, Ayas, Shania, Naja, Andre, Yuda, Zevi, Fasma, Pede, Fajar, Rafi, Ariya, Ka Adam, Ka Irfan, Ka Ovan, Ka Hanif, Ka Uli, Ka Ikhsan, dan Ka Fany sebagai keluarga admin Lab Mobile Innovation Studio.
8. Huda, Hania, Hero, dan Eritha sebagai teman seperjuangan Kerja Praktik di PT. PLN (Persero).
9. Teman-teman Informatika ITS angkatan 2015, teman-teman lab Manajemen Informasi, teman-teman club Saman dan Pamor, teman-teman komunitas SSL, teman-teman Humas dan BPH Schematics 2016 dan Schematics 2017, teman-teman Departemen Pengembangan Profesi HMTC Inspirasi dan HMTC Kreasi, serta teman-teman BPU JMMI Integrasi dan BPU JMMI Inspirasi yang tidak bisa disebutkan namanya satu persatu.
10. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis memohon maaf jika terdapat kesalahan maupun kekurangan dalam Tugas Akhir. Oleh karena itu dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan untuk kedepannya. Semoga laporan Tugas Akhir ini dapat berguna bagi pembaca.

Surabaya, Januari 2019

Purina Qurota Ayunin

DAFTAR ISI

[*Halaman ini sengaja dikosongkan*]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1 Latar Belakang

Saat ini proses belajar mengajar perkuliahan tidak lagi harus menghadiri kelas secara langsung. Seseorang dapat mengikuti kelas secara *online* melalui situs-situs yang menerapkan sistem *Massive Open Online Course* (MOOC). MOOC merupakan sebuah sistem kuliah secara terbuka dan berbasis *online* yang sebagian besar tidak memungut biaya. Komisi Eropa mendefinisikan MOOC sebagai "kursus *online* terbuka untuk siapa saja tanpa batasan (gratis dan tanpa batas untuk hadir), biasanya terstruktur sekitar satu set tujuan pembelajaran di bidang studi, yang sering berjalan selama periode waktu tertentu (dengan tanggal awal dan akhir) di atas *platform online* yang memungkinkan kemungkinan interaktif (antara teman sebaya atau antara siswa dan instruktur) yang memfasilitasi penciptaan komunitas belajar. Seperti halnya untuk setiap kursus *online* yang menyediakan beberapa materi pelajaran dan alat penilaian (sendiri) untuk belajar mandiri" [1].

Situs MOOC populer di dunia adalah EdX (dibuat oleh Universitas Harvard dan MIT) dan Coursera. Di Indonesia sendiri situs MOOC yang disediakan antara lain CodeSaya, FOCUS Fisipol UGM, IndonesiaX, MOOCs Universitas Terbuka, Sibejoo, UCEO Universitas Ciputra [2] dan Spada dari Menristek DIKTI. Institut Teknologi Sepuluh Nopember juga memiliki *website* dengan sistem MOOC yang di bangun dengan LMS Moodle yaitu *Share* ITS (bisa diakses di www.share.its.ac.id) dan di jurusan Informatika ITS tersedia *E-Learning* IF ITS (bisa diakses di www.elearning.if.its.ac.id).

Pada umumnya, *Massive Open Online Course* di Indonesia, khusunya untuk *e-learning* di perguruan tinggi dibangun dari *Learning Management System (LMS)* seperti aplikasi Moodle. Moodle adalah aplikasi yang dapat mewadahi media pembelajaran ke dalam web. Dengan menggunakan Moodle, penyedia kursus

dapat membuat materi pembelajaran, kuis, jurnal elektronik dan lain-lain. Untuk *standard* pengembangan dan pendistribusian pembelajaran *online* Moodle mengikuti *standard* SCORM. *Standard* SCORM pada Moodle berguna apabila konten kursus harus di *export* untuk selanjutnya digunakan pada LMS lain. Maka hal ini akan berguna jika suatu kursus dengan materi yang sama ingin digunakan kembali pada semester-semester berikutnya atau jika ada imigrasi konten pembelajaran pada *e-learning*.

Namun pada praktiknya, sebagian besar dosen khususnya di Informatika ITS yang menggunakan *e-learning* Informatika ITS atau *Share* ITS memanfaatkan fitur pembagian materi kuliah perminggu tanpa menjelaskan deskripsi, capaian pembelajaran, dan pokok bahasan materi kuliah secara jelas di halaman kursus tersebut. Hal ini akan menyulitkan mahasiswa dalam memahami gambaran secara umum materi kuliah yang diambil.

Christobal Romero dkk. pada jurnalnya yang berjudul "*Data mining in course management systems: Moodle case study and tutorial*" menyarankan penggunaan Data Mining dalam mengoptimalkan proses kegiatan belajar dengan mengekstrak data kursus kemudian diolah untuk mencapai tujuan tertentu [3].

Maka pada tugas akhir ini, akan dibuat suatu model pengklasifikasian untuk rekomendasi pelabelan konten pembelajaran secara otomatis. Pendekatan yang dilakukan adalah dengan mengekstraksi konten kursus lalu melakukan proses *Natural Language Processing* (NLP) tanpa *Machine Learning* dan menggunakan *Machine Learning* atau *text mining*. Dataset diambil dari beberapa MOOC seperti Spada, *Share* ITS, *E-learning* Informatika ITS, IndonesiaX, KelasKita, *Open Course Ware* Univesitas Indonesia, dan EdX. Data yang diambil mengikuti beberapa data yang ada pada standard metadata SCORM berupa judul mata kuliah sesuai *tag* `<title>`, deskripsi mata kuliah sesuai *tag* `<description>`, capaian pembelajaran sesuai *tag* `<coverage>` dan pokok bahasan sesuai *tag* `<keyword>`. Setiap judul diklasifikasikan sebagai kelas Nama Mata Kuliah dan setiap paragraf yang memiliki kemiripan objek dan konteks bahasan di

klasifikasikan menjadi 3 kelas, yaitu kelas Deskripsi Mata Kuliah yang memiliki konteks penjelasan secara umum materi perkuliahan, Pokok Bahasan Mata Kuliah yang menjelaskan materi-materi perkuliahan, dan Capaian Pembelajaran yang menjelaskan harapan untuk mahasiswa (*student*) setelah mengambil mata kuliah.

Metode NLP tanpa *Machine Learning* yang dilakukan yaitu dengan menerapkan sejumlah *rule* untuk diklasifikasikan menjadi 4 kelas (judul, deskripsi, capaian pembelajaran, dan pokok bahasan). Untuk NLP dengan *Machine Learning*, *rule* tersebut dijadikan fitur. Selain itu juga ditambahkan fitur dengan mengimplementasikan pembobotan *term* dengan metode TF-IDF. Selanjutnya diimplementasikan pada metode *Random Forest*, *Support Vector Machine* (SVM) dan *Naïve Bayes*. Dikarenakan jumlah anggota kelas tidak seimbang, maka pada pembagian *training* dataset diatasi dengan menggunakan metode *over sampling* [4] yaitu dengan metode ADASYN dan SMOTE. Selain itu dilakukan juga pembagian *training* dan *testing* data dengan metode *Cross Validation*.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara menentukan anotasi detail konten kursus pada MOOC sehingga memudahkan pencarian kursus yang sesuai?
2. Bagaimana cara mengekstraksi setiap anotasi pada kursus menjadi sebuah metadata yang sesuai standar SCORM dan Moodle?

1.3 Batasan Masalah

Batasan masalah pada tugas akhir ini antara lain:

1. MOOC yang akan digunakan adalah Spada, Share ITS, E-learning Informatika ITS, IndonesiaX, KelasKita, Open Course Ware Univesitas Indonesia, dan EdX.

2. Kursus yang dipilih merupakan kursus yang memiliki judul dan paragraf deskripsi, pokok bahasan atau capaian pembelajaran.
3. Ruang lingkup untuk data masukan jenis material terbatas pada judul kursus, deskripsi kursus, capaian pembelajaran, dan pokok bahasan.
4. Proses *Natural Language Processing* tanpa *Machine Learning* dilakukan dengan menerapkan *rule* yang penulis buat.
5. Proses *Text Mining* dilakukan menggunakan metode klasifikasi *Random Forest*, *Support Vector Machine* (SVM) dan *Naïve Bayes* dengan bahasa pemrograman Python.
6. Pengklasifikasian dilakukan dengan memberikan satu label per-paragraf atau data keterangan mata kuliah.
7. Konten yang digunakan menggunakan Bahasa Indonesia. Khusus untuk MOOC EdX, dilakukan proses penerjemahan terlebih dahulu dengan pustaka TextBlob pada bahasa pemrograman Python.
8. Tugas akhir tidak membangun metadata untuk profil pengguna (pelajar) pada MOOC.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini yaitu untuk membuat model untuk rekomendasi pengklasifikasian label konten pembelajaran secara otomatis dengan mengekstraksi konten kursus yang tersedia pada situs MOOC. Sehingga mahasiswa atau *user* akan lebih mudah memahami gambaran secara umum mata kuliah yang akan diambil. Maka bagian yang di ekstraksi dari metadata adalah judul dan deskripsi mata kuliah. Kemudian data tersebut di klasifikasi dan dimaknai sebagai deksripsi, capaian pembelajaran atau pokok bahasan.

1.5 Manfaat

Tugas akhir ini diharapkan dapat menghasilkan model yang dapat merekomendasikan pengklasifikasian dalam memberi label pada konten pembelajaran secara otomatis. Penjelasan mata kuliah

yang telah diklasifikasikan berdasarkan deskripsi, pokok bahasan, dan capaian pembelajaran yang bisa membantu mahasiswa dalam memahami kursus yang di *enroll* pada situs *Massive Open Online Course*.

1.6 Metodologi

Langkah-langkah yang harus ditempuh dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Penyusunan Proposal Tugas Akhir

Proposal tugas akhir ini berisi tentang penjelasan mengenai pendahuluan dari tugas akhir yang dibuat. Pendahuluan ini terdiri dari hal-hal yang melatarbelakangi tugas akhir, rumusan masalah yang diangkat, batasan masalah yang ada, tujuan, dan manfaat dari tugas akhir ini. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi dalam pengerjaan tugas akhir ini.

2. Studi Literatur

Pada studi literatur tugas akhir ini dipelajari sejumlah referensi yang relevan terhadap tugas akhir yang akan dikerjakan. Literasi yang didapatkan berasal dari jurnal yang kredibel, dokumentasi website resmi, serta artikel yang dapat dipercaya. Hal-hal yang dipelajari yaitu tentang *MOOC, text processing, klasifikasi Random Forest, klasifikasi SVM, klasifikasi Naïve Bayes, oversampling, ADASYN, SMOTE, dan Cross Validation*.

3. Analisis dan desain metode

Analisis yang dilakukan pada tugas akhir ini dengan memahami dataset yang telah diambil. Lalu dibuat beberapa *rule* untuk metode *Natural Language Processing* tanpa *Machine Learning*. Lalu pada pengklasifikasian dengan *Machine Learning* setiap *rule* tersebut digunakan sebagai fitur. Selain itu juga akan ditambahkan fitur pembobotan *term* dengan metode TF-IDF. Metode pengklasifikasian yang dilakukan adalah klasifikasi *Random Forest*, klasifikasi SVM, dan klasifikasi *Naïve Bayes.* Pada masing-masing metode klasifikasi, pemisahan dataset dilakukan dengan dua cara. Pertama dengan pemisahan dataset berdasar rasio menjadi 33% sebagai *data testing* dan 67% sebagai

data training. Pada pemisahan berdasar rasio, *data training* juga diuji dengan melakukan *sampling* data menggunakan metode *oversampling ADASYN* dan *SMOTE*.

4. Implementasi metode

Penulisan program pada tugas akhir ini menggunakan *tools* berupa bahasa pemrograman Python 3.6.4 dan IDE *Jupyter Notebook* dari Anaconda. Pustaka yang digunakan merupakan pustaka bawaan Python diantaranya NumPy, Panda, serta pustaka lain yang mendukung *text-processing* mulai dari *pre-processing* hingga uji coba dan evaluasi.

Sedangkan untuk perangkat keras dan sistem operasi yang digunakan selama tugas akhir adalah:

- Intel Core i7-7500U @ 2.7GHz (4 CPUs)
- NVIDIA GeForce 930MX 4 GB
- 4 GB of RAM
- Windows 10 Home 64-bit

5. Uji Coba dan Evaluasi

Uji coba pada tugas akhir ini dilakukan untuk mengetahui apakah penggunaan metode ini sudah tepat demi tercapainya tujuan dari tugas akhir ini. Uji coba terlebih dahulu dibuat skenarionya agar pengujiannya tepat sasaran dan sesuai dengan tujuan yang diinginkan.

Evaluasi dalam tugas akhir ini bermanfaat untuk mendapatkan nilai kuantitatif dari skenario uji coba yang dilakukan. Pada tugas akhir ini, metrik yang digunakan adalah *accuracy*, *precision*, *recall*, dan *F-Measure.* Hasil keempat metrik ini diintepretasikan untuk menjawab apakah tujuan dari tugas akhir ini tercapai.

Evaluasi dilakukan dengan berbagai cara dalam membagi data menjadi data *training* dan data *testing*. Pertama dengan pembagian data *training* 67% dan data *testing* 33%, kedua dengan pembagian dengan cara *cross validation* menggunakan *stratified k-fold*, lalu penyeimbangan data *training* menggunakan metode *over sampling* dengan algoritma ADASYN dan SMOTE. Setelah itu dilakukan

prediksi dengan menggunakan model produk dari sistem berdasarkan data test. Lalu, hasil dari kelas-kelas prediksinya dicocokkan dengan kelas/label pada data test. Setelah itu, baru dihitung hasil dari *accuracy, precision, recall,* dan *F-Measure*-nya.

6. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi metode yang telah dibuat. Sistematika penulisan buku tugas akhir ini secara garis besar antara lain:

1. Pendahuluan
   a. Latar Belakang
   b. Rumusan Masalah
   c. Batasan Tugas Akhir
   d. Tujuan
   e. Manfaat
   f. Metodologi
   g. Sistematika Penulisan
2. Tinjauan Pustaka
3. Analisis dan Perancangan Sistem
4. Implementasi
5. Uji Coba dan Evaluasi
6. Kesimpulan dan Saran
7. Daftar Pustaka

1.7 Sistematika Penulisan

Sistematika Penulisan Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini:

BAB I. Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan,

batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

BAB II. Tinjauan Pustaka

Bab ini menjelaskan beberapa teori yang dijadikan penunjang dan berhubungan dengan pokok pembahasan yang mendasari pembuatan tugas akhir.

BAB III. Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan sistem yang akan dibangun. Perancangan sistem meliputi perancangan data dan alur proses dari sistem itu sendiri.

BAB IV. Implementasi

Bab ini berisi implementasi dari perancangan sistem yang telah ditentukan sebelumnya.

BAB V. Uji COba dan Evaluasi

Bab ini membahas pengujian dari metode yang ditawarkan dalam tugas akhir untuk mengetahui kesesuaian metode dengan data yang ada.

BAB VI. Kesimpulan dan Saran

Kesimpulan Bab ini berisi kesimpulan dari hasil pengujian yang telah dilakukan. Bab ini juga membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi data atau daftar istilah yang penting pada tugas akhir ini.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar tugas akhir ini.

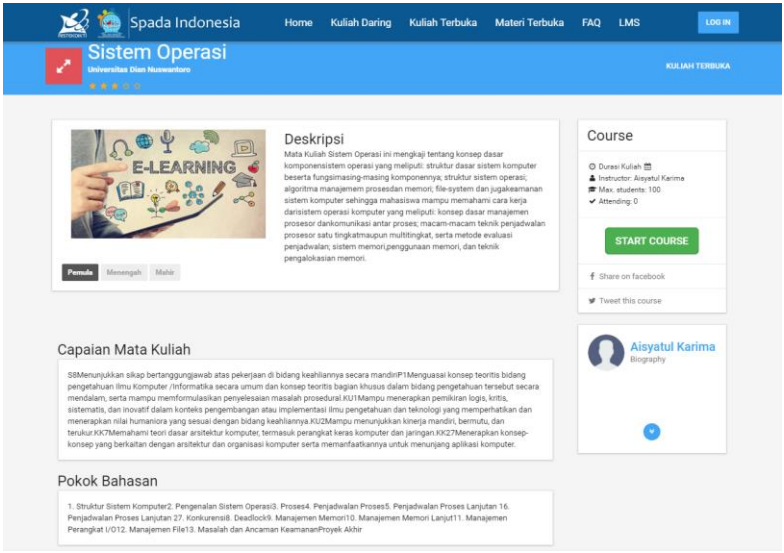
2.1 MOOC

MOOC atau *Massive Open Online Course* merupakan sebuah model baru dalam penyediaan pendidikan tinggi dengan pemanfaatan teknologi internet dan *mobile internet* [5]. Menurut New York Times, tahun 2012 merupakan "*the year of the MOOC*", karena beberapa *platform* menyediakan universitas bergengsi seperti Coursera (Stanford University) dan EdX (Harvard University dan MIT) [1]. Di Indonesia sendiri situs MOOC yang disediakan antara lain CodeSaya, FOCUS Fisipol UGM, IndonesiaX, MOOCs Universitas Terbuka, Sibejoo, UCEO Universitas Ciputra [2] dan Spada dari Menristekdikti. Institut Teknologi Sepuluh Nopember juga memiliki *website* dengan sistem MOOC yang di bangun dengan LMS Moodle yaitu *Share* ITS (bisa diakses di www.share.its.ac.id) dan di jurusan Informatika ITS tersedia *E-Learning* IF ITS (bisa diakses di www.elearning.if.its.ac.id).

Sesuai namanya, MOOC bersifat terbuka dan disediakan bagi siapapun (dalam hal ini mahasiswa) secara bebas dan gratis. Komisi Eropa mendefinisikan MOOC sebagai "kursus *online* terbuka untuk siapa saja tanpa batasan (gratis dan tanpa batas untuk hadir), biasanya terstruktur sekitar satu set tujuan pembelajaran di bidang studi, yang sering berjalan selama periode waktu tertentu (dengan tanggal awal dan akhir) di atas *platform online* yang memungkinkan kemungkinan interaktif (antara teman sebaya atau antara siswa dan instruktur) yang memfasilitasi penciptaan komunitas belajar. Seperti halnya untuk setiap kursus *online*,

MOOC menyediakan beberapa materi pelajaran dan alat penilaian (sendiri) untuk belajar mandiri" [1].

Struktur dasar paling umum pada setiap MOOC yaitu memiliki laman kursus yang berisi judul kursus, keterangan kursus (deskripsi, capaian pembelajaran, dan pokok bahasan), waktu atau durasi pembelajaran, dan nama pengajar.



Gambar 2.1 - Laman Kursus MOOC Spada

Sedangkan pada Moodle sendiri laman kursus terdiri dari beberapa konten pengajaran selama setahun dan beberapa sesi yang digunakan oleh pengajar dalam membagikan materi, seperti *file*, *assignment*, forum, dll. Kursus juga disusun menjadi beberapa kategori. Pada tugas akhir ini data yang diambil hanya penjelasan umum mata kuliah yang terdiri dari judul mata kuliah dan deskripsi mata kuliah, capaian pembelajaran, atau pokok bahasan jika tersedia.

2.2 SCORM

SCORM atau *Sharable Content Object Reference Model* merupakan seperangkat standar teknis untuk produk perangkat lunak yaitu *e-learning*. SCORM memudahkan *programmers* dalam menulis *code* agar aplikasi *e-learning* yang dibuat bisa beradaptasi dengan berbagai *platform* perangkat lunak *e-learning*. Secara khusus, SCORM mengatur bagaimana konten pembelajaran *online* dan *Learning Management Systems* (LMS) berkomunikasi satu sama lain. SCORM tidak berbicara dengan desain instruksional atau masalah pedagogis lainnya karena SCORM murni standar teknis.

Sesuai namanya "*Sharable Content Object*", menunjukan bahwa SCORM mampu membuat berbagai unit materi pelatihan *online* yang bisa dibagikan di seluruh sistem. SCORM mendefinisikan cara membuat "*Sharable Content Object*" atau "SCO" yang dapat digunakan kembali dalam sistem dan konteks yang berbeda. Sedangkan untuk "*Reference Model*" maksudnya adalah SCORM memperhatikan bahwa industri sudah memiliki banyak standar yang mampu memecahkan masalah, maka SCORM hanya mereferensikan standar yang ada ini dan memberi tahu para pengembang bagaimana cara menggunakannya dengan tepat [6].

SCORM menyediakan interaksi dengan *Learning Management System* (LSM) dengan ketentuan dari element-element pada SCORM. Element pada SCORM memberikan informasi tentang bagaimana paket-paket pada SCORM dapat bekerja. LMS dapat menggunakan element-element tersebut untuk mendefinisikan seperangkat aturan yang mengontrol *progress* peserta didik melalui konten-konten yang ditentukan. Setiap element bersifat optional [7]. Struktur metadata LOM SCORM [8] dapat dilihat pada Lampiran 1.a.

2.3 Taksonomi Bloom

Taksonomi berasal dari dua kata dalam bahasa Yunani yaitu *tassein* yang berarti mengklasifikasi dan *nomos* yang berarti aturan. Jadi Taksonomi berarti hierarkhi klasifikasi atas prinsip dasar atau aturan. Istilah ini kemudian digunakan oleh Benjamin Samuel Bloom, seorang psikolog bidang pendidikan yang melakukan penelitian dan pengembangan mengenai kemampuan berpikir dalam proses pembelajaran.

Taksonomi Bloom adalah struktur hierarkhi yang mengidentifikasikan *skills* mulai dari tingkat yang rendah hingga yang tinggi. Tentunya untuk mencapai tujuan yang lebih tinggi, *level* yang rendah harus dipenuhi lebih dulu. Dalam kerangka konsep ini, tujuan pendidikan oleh Bloom dibagi menjadi tiga domain/ranah kemampuan intelektual (*intellectual behaviors*) yaitu kognitif, afektif dan psikomotorik [9]. Dalam setiap taksonomi terkandung kata kerja operasional yang menggambarkan bentuk perilaku yang ingin dicapai melalui suatu pembelajaran. Kata kerja operasional untuk ranah kognitif (pengetahuan) pada Taksonomi Bloom ini kemudian digunakan sebagai salah satu fitur pada tugas akhir seperti yang tertera pada Lampiran 1.b.

2.4 *Natural Language Processing*

Natural Language Processing adalah bagian dari Kecerdasan Buatan yang memfokuskan pada kemampuan komputer dalam memahami dan memproses bahasa manusia, agar komputer lebih dekat dalam pemahaman bahasa setara *level* manusia. Karena komputer tidak memiliki kemampuan intuisi dalam memahami bahasa natural yang manusia lakukan, komputer tidak dapat memahami apa yang sebenarnya sebuah teks maksud. Dengan demikian, kemajuan dalam *Machine Learning* telah memungkinkan komputer dalam melakukan banyak hal yang bermanfaat dalam *natural language*.

Bidang penerapan NLP yang popular diantaranya penjawab pertanyaan (*question answering*), *information extraction*, *sentiment analys*, *machine translation*, *information retrieval*, *summarization*, dan masih banyak lagi. Pada tugas akhir ini, bidang penerapan NLP yang dilakukan adalah *information extraction*. Dimana program mampu mengekstrak data (anotasi metadata) pada MOOC menjadi informasi yang dapat dimanfaatkan.

2.5 *Text Processing*

Text processing atau bisa disebut juga sebagai *Text Mining* adalah sebuah pemrosesan teks untuk menghasilkan informasi atau *insights* pada suata data. Untuk menghasilkan beberapa informasi *Text Mining* menggunakan beberapa metode salah satunya NLP (*Natural Laguage Processing*) [10]. Terdapat beberapa tugas dalam NLP diantaranya:

2.2.1. *Case Folding*

Case folding adalah mengubah semua huruf dalam dokumen menjadi huruf kecil (*lower case*). Hanya huruf ‘a’ sampai dengan ‘z’ yang diterima. Karakter selain huruf dihilangkan dan dianggap delimiter.

2.2.2. *Regular Expression*

Regular Expression atau yang lebih sering disebut *regex* merupakan sebuah teknik yang digunakan untuk mencocokan string teks, seperti karakter tertentu, kata-kata, atau pola karakter. *Regex* memiliki 2 fungsi utama yakni mencari dan mengganti, mencari suatu pola tertentu dalam teks lalu menggantinya menjadi pola yang lain [11].

2.6 *POS Tagging*

Part of Speech (POS) Tagging merupakan suatu proses penetapan salah satu bagian (*part of speech*) pada token atau kata yang diberikan. Bagian dari token antara lain kata benda (*noun*), kata pengganti (*proper noun*), kata kerja (*verbs*), kata sifat

(*adjectives*), kata keterangan (*adverb*), konjungsi (*conjunction*), sub-kategori dan delimeter.

POS Tagging dapat berguna untuk berbagai aplikasi, karena POS Tagging merupakan dasar dan *primary analysis* dalam berbagai *syntactic parsers*. POS *Tagging* juga dapat diaplikasikan untuk ekstraksi informasi, mesin penerjemah, *name entity recognition* (NER), *speech syntesis, lexicographic research*, *term extraction*, dan lain sebagainya [12].

2.7 TF-IDF

Term Frequency – Inverse Document Frequency terbagi menjadi TF dan IDF. TF adalah jumlah/frekuensi kata yang muncul pada dokumen yang bersangkutan. Jika suatu kata pada dokumen tersebut sering muncul, maka bobot yang didapatkan akan lebih besar.

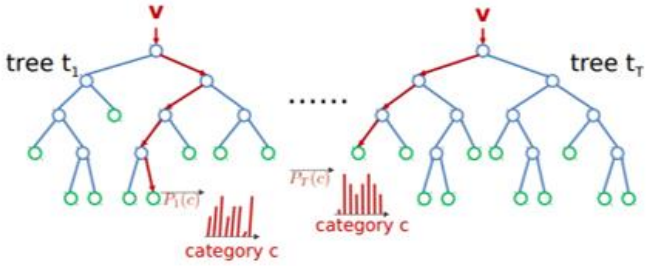
$$TF = \begin{cases} 1 + \log(f_{t,d}), & f_{t,d} > 0 \\ 0, & f_{t,d} = 0 \end{cases} \quad (2.1)$$

IDF adalah distribusi suatu *term* pada semua dokumen yang ada yang menunjukan ketersediaan *term* tersebut pada semua dokumen. Semakin sedikit jumlah *term* pada persebaran dokumen, maka nilai IDF semakin besar. Hal ini menunjukan *term* tersebut merupakan *term* unik dan penting pada dokumen tertentu [13].

$$IDF_j = \log(\frac{D}{df_j}) \quad (2.2)$$

2.8 *Random Forest*

Random forest pertama kali dikenalkan oleh Tin Kam Ho dari Bell Labs pada tahun 1995, merupakan *ensemble learning method* untuk klasifikasi dan regresi. Algoritma untuk menginduksi *Random Forest* pertama kali dibangun oleh Leo Breimen dan Adele Cutler pada tahun 2001 [14]. Algoritma *Random Forest* menggunakan *decision tree classifier* sebagai *base learner*.



Gambar 2.2 - *A forest of Trees*

Formula:

$$P(c|v) = \sum_{t=1}^{T} Pt(c|v) \quad (2.3)$$

Keterangan:

P(c|v) – klasifikasi final test

Pt(c|v) – klasifikasi pada setiap *tree*

T – jumlah tree

2.8.1 Support Vector Machine

Support Vector Machine (SVM) dikembangkan oleh Boser, Guyon, Vapnik, dan pertama kali dipresentasikan pada tahun 1992 di *Annual Workshop on Computational Learning Theory*. Konsep kerja SVM yaitu berusaha menemukan *hypeprlane* terbaik dalam memisahkan dua buah kelas pada *input space* dengan cara mengukur margin *hyperplane* tersebut dan mencari titik maksimalnya. Prinsip dasar SVM adalah *linear classifier* dengan memasukan konsep *kernel trick* pada ruang kerja berdimensi tinggi [15].

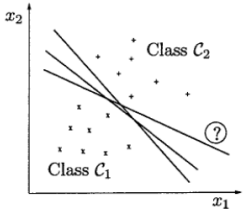
Pada SVM Linear misalkan data *training* direpresentasikan dengan

$$x_1, y_1, \dots, (x_k, y_k), x \in R^n, y \in \{+1, -1\} \quad (2.4)$$

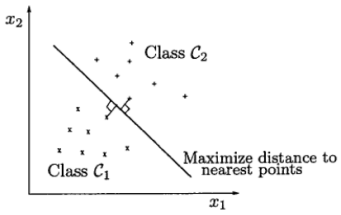
Dapat dipisahkan oleh suatu *hyperlane*

$$x_1, y_1 - b = 0 \quad (2.5)$$

Untuk suatu training set, bisa diperoleh lebih dari satu hyperlane seperti contoh pada Gambar 2.3 dibawah



Gambar 2.3 - Contoh banyaknya alternatif *hyperlane* untuk suatu masalah klasifikasi



Gambar 2.4 - Pendefinisian *hyperlane* yang unik berdasarkan konsep optimal *hyperlane*

Model SVM bertujuan untuk membangun suatu fungsi yang menghasilkan *hyperlane* yang optimal, yaitu *hyperlane* dengan margin terbesar (Gambar 2.4), yang dirumuskan sebagai berikut [16].

$$y_i[(w \cdot x_i) - b] \geq 1. i = 1,2, \dots, l \quad (2.6)$$

2.9 *Naïve Bayes*

Naïve Bayes Classifier merupakan salah satu metode *machine lear*ning yang memanfaatkan perhitungan probabilitas dan statistik yang dikemukakan oleh ilmuan Inggris Thomas Bayes, yaitu memprediksi probabilitas di masa depan berdasarkan pengalaman di masa sebelumnya.

Dasar dari *Naïve Bayes* dipakai dalam pemrograman adalah rumus Bayes:

$$P(A\,|B) = (P(B|A) * P(A)/P(B) \quad (2.7)$$

Peluang kejadian A sebagai B ditentukan dari peluang B saat A, peluang A, dan peluang B. Pada pengaplikasiannya nanti, rumus ini berubah menjadi:

$$P(C_I\,|D) = (P(D|C_I) * P(C_I))/P(D) \quad (2.8)$$

Naïve Bayes Classifier atau bisa disebut sebagai *Multinominal Naïve Bayes* merupakan model penyederhanaan dari metode Bayes yang cocok dalam pengklasifikasian teks atau dokumen. Persamaannya adalah:

$$V_{MAP} = \arg\max P\left(V_j \,|a_1, a_2, \dots\dots, a_n\right) \quad (2.9)$$

Menurut persamaan (2.9), maka persamaan (2.7) dapat ditulis:

$$V_{MAP} = \underset{v_j \in V}{\arg max} \frac{P(a_1,a_2,\dots\dots,a_n \mid v_j)P(v_j)}{P(a_1,a_2,\dots\dots,a_n)} \quad (2.10)$$

$P(a_1, a_2, \dots, a_n)$ konstan, sehingga dapat dihilangkan menjadi

$$V_{MAP} = \underset{v_j \in V}{\arg max}\, P(a_1, a_2, \ldots\ldots, a_n \mid v_j) P(v_j) \qquad (2.11)$$

Karena $P(a_1, a_2, \ldots\ldots, a_n \mid v_j) P(v_j)$ sulit untuk dihitung, maka akan diasumsikan bahwa setiap kata pada dokumen tidak mempunyai keterkaitan.

$$V_{MAP} = \underset{v_j \in V}{\arg max}\, P(v_j) \prod i\, P(a_i | v_j) \qquad (2.12)$$

Keterangan:

$$P(a_i | v_i) = \frac{|\, docs_j \,|}{contoh} \qquad (2.13)$$

$$P(w_k | v_j) = \frac{n_k + 1}{n + |kosakata|} \qquad (2.14)$$

Dimana untuk:

- $P(v_j)$: Probabilitas setiap dokumen terhadap sekumpulan dokumen
- $P(w_k|v_j)$: Probabilitas kemunculan kata w_k pada suatu dokumen dengan kategori class v_j
- | doc | : frekuensi dokumen pada setiap kategori
- | contoh | : jumlah dokumen yang ada
- N_k : frekuensi kata ke-k pada setiap kategori
- Kosakata : jumlah kata pada dokumen test

Pada persamaan (2.14) terdapat suatu penambahan 1 pada pembilang, hal ini dilakukan untuk mengantisipasi jika terdapat suatu kata pada dokumen uji yang tidak ada pada setiap dokumen data *training* [17].

2.10 Dataset Tidak Seimbang

Ketidakseimbangan data telah menjadi masalah dalam *machine learning* karena dapat memengaruhi akurasi dalam klasifikasi. Berbagai metode telah dibuat untuk menangani

ketidakseimbangan dataset agar dapat meningkatkan akurasi dalam prediksi model [18]. Dataset dengan kelas yang tidak seimbang diakibatkan karena ketidakseimbangan rasio antar satu kelas dan lainnya. Contohnya pada dataset tugas akhir ini dimana perbandingan antara kelas tidak seimbang seperti pada Tabel 2.1.

Tabel 2.1 - Dataset tidak seimbang

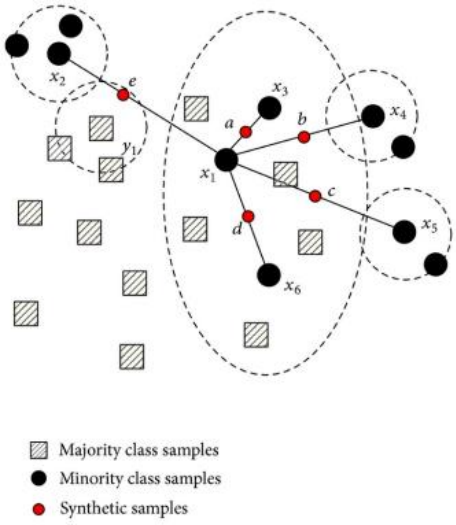
Kelas	Anggota
Nama Mata Kuliah	376
Deskripsi Mata Kuliah	375
Capaian Pembelajaran	365
Pokok Bahasan	20

Pada Tabel 2.1 kelas Pokok Bahasan dengan anggota paling sedikit tidak mencapai 10% dari kelas lain. Ketidakseimbangan kelas bisa berdampak merugikan pada proses *data mining* karena *machine learning* memiliki kesulitan dalam mengklasifikasikan kelas minoritas dengan semestinya [19].

Maka solusi dalam menangani ketidakseimbangan dataset dapat dilakukan dengan dua acara, yaitu metode *under sampling* dan *over sampling*. Metode *under sampling* dilakukan dengan menurunkan *dataset* asli hingga rasio antar kelas paling tidak mencapai 10:1. Beberapa cara dalam melakukan under sampling diantaranya *Condensed Nearest-Neighbor* dan *One-sided Selection*. Namun metode *under sampling* memiliki resiko kehilangan data yang memiliki peran penting dalam klasifikasi di kelas tesebut. Sedangkan metode *over sampling* adalah metode yang menggunakan *synthetic data generation* dalam meningkatkan jumlah *sample* pada dataset. Cara dalam melakukan *over sampling* diantaranya:

2.1.1 SMOTE

Algoritma *Sampling Approach for Support Vector* (SMOTE) melakukan over sampling dengan membuat replikasi dari data minoritas yang dikenal dengan istilah data "sintetis". Metode SMOTE bekerja dengan menemukan *k-nearest neighbors* di kelas minoritas untuk setiap sample pada kelas minoritas. Lalu digambarkan sebuah garis antar *neighbors* dan *generate random point* pada garis tersebut. setelah itu dibuat data sintetis sebanyak prosentase duplikasi yang diinginkan antara data minor dan *knearest neighbors* yang dipilih secara acak [20].



Gambar 2.5 - SMOTE

2.1.2 ADASYN

Ide utama dari ADASYN adalah menggunakan bobot distribusi untuk data pada kelas minoritas berdasarkan pada tingkat kesulitan belajar, sehingga data sintesis dihasilkan dari kelas minoritas yang susah untuk belajar dibandingkan dengan data

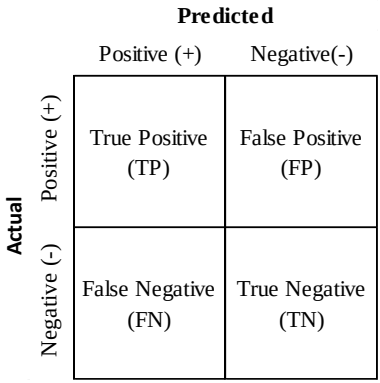
minoritas yang lebih mudah untuk belajar. ADASYN meningkatkan pembelajaran dengan dua cara. Pertama, mengurangi bias yang diakibatkan oleh ketidakseimbangan kelas dan yang kedua secara adaptif menggeser batas keputusan klasifikasi terhadap kesulitan data [21].

2.11 *Cross Validation*

Cross Validation adalah prosedur *resampling* untuk mengevaluasi model di *machine learning* pada sampel data yang terbatas [22]. Dalam *K fold cross validation*, data dibagi menjadi K himpunan bagian. Sehingga setiap kali proses ini, salah satu K himpunan digunakan sebagai *set testing* atau set validasi dan himpunan K lainnya disatukan untuk membentuk satu *set training*. Estimasi kesalahan adalah rata-rata atas semua percobaan K untuk mendapatkan efektivitas total dari model. Ini secara signifikan mengurangi bias karena menggunakan sebagian besar data untuk pemasangan, dan juga secara signifikan mengurangi varians karena sebagian besar data juga digunakan dalam set validasi. Saling menukar posisi pada *training* dan *set testing* juga menambah keefektifan metode ini. Sebagai aturan umum dan bukti empiris, K = 10 umumnya lebih banyak digunakan, tetapi tidak ada yang tetap dan dapat mengambil nilai berapapun [23].

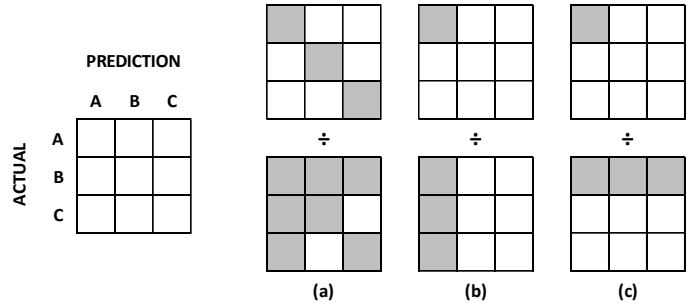
2.12 *Confusion Matrix*

Confusion matrix adalah sebuah metode dalam melakukan perhitungan hasil klasifikasi dalam *data mining* yang dijelaskan pada Gambar 2.6 yang tediri dari: TP, TN, FP, dan FN. Di mana TP adalah *True Prositive* yang artinya adalah jumlah kebenaran antara hasil klasifikasi dengan jumlah seluruh data. TN adalah *True Negative* yang menyatakan jumlah hasil klasifikasi yang diindikasikan benar, tetapi sesungguhnya salah. FP merupakan *False Positive* yakni jumlah dari hasil klasifikasi yang diindikasikan salah, tetapi sesungguhnya benar. Dan FN atau *False Negative* merupakan jumlah dari kesamaan hasil klasifikasi dan yang sesungguhnya adalah salah.



Gambar 2.6 - Confusion Matrix

Penjelasan metrik evaluasi *accuracy*, *precision*, *recall*, dan *F-Measure* yang lebih rinci dipaparkan sebagaimana berikut:



Gambar 2.7 - Ilustrasi *confusion matrix* pada 3 kelas

a. *Accuracy*

Akurasi adalah tingkat kedekatan antara nilai prediksi dengan nilai aktualnya. Akurasi dapat dihitung dengan membagi nilai $TP + TN$ dengan jumlah seluruh data sehingga akurasi dapat dirumuskan dengan

$$accuracy = \frac{TP + \sum TN}{TP + \sum TN + \sum FP + \sum FN}. \qquad (2.15)$$

Dalam tugas akhir ini, kelas yang digunakan akan lebih dari dua. Maka dari itu, pada Gambar 2.7 bagian (a), diilustrasikan bagaimana mendapatkan akurasi pada perspektif kelas A pada *confusion matrix* tiga kelas.

b. Precision

Precision atau presisi merupakan tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. *Precision* dapat dihitung dengan membagi TP dengan jumlah dari TP dan FP sehingga *precision* dapat dirumuskan menjadi

$$precision = \frac{TP}{\sum TP + \sum FP}. \quad (2.16)$$

Pada perspektif kelas A, nilai *precision* dapat diambil dengan cara melihat ilustrasi *confusion matrix* pada Gambar 2.7 bagian (b).

c. Recall

Recall adalah nilai dari ketepatan informasi yang diprediksi relevan pada suatu kelas terhadap data asli pada kelas tersebut. *Recall* dapat dihitung dengan membagi TP dengan jumlah TP dan FN sehingga *recall* dapat dirumuskan dengan

$$recall = \frac{TP}{\sum TP + \sum FN}. \quad (2.17)$$

Pada perspektif kelas A, nilai *recall* dapat diambil dengan melihat ilustrasi *confusion matrix* pada Gambar 2.7 (c).

d. F-Measure

F-Measure memanfaatkan nilai presisi dan *recall.* Memisahkan dokumen-dokumen yang mirip kadang lebih buruk dibandingkan dengan menempatkan pasangan dokumen yang tidak mirip ke dalam satu kelompok. Oleh karena itu, digunakan *F-Measure* dengan nilai FN yang lebih kuat dibandingkan dengan nilai FP. *F-Measure* atau F_1 dapat dihitung dengan menggunakan

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \quad (2.18)$$

Kemudian, hasil dari masing-masing perspektif pada setiap kelas dirata-rata untuk mendapatkan hasil akhir dari *accuracy*, *precision*, *recall*, dan *F-Measure*.

BAB III
Analisis dan Perancangan Sistem

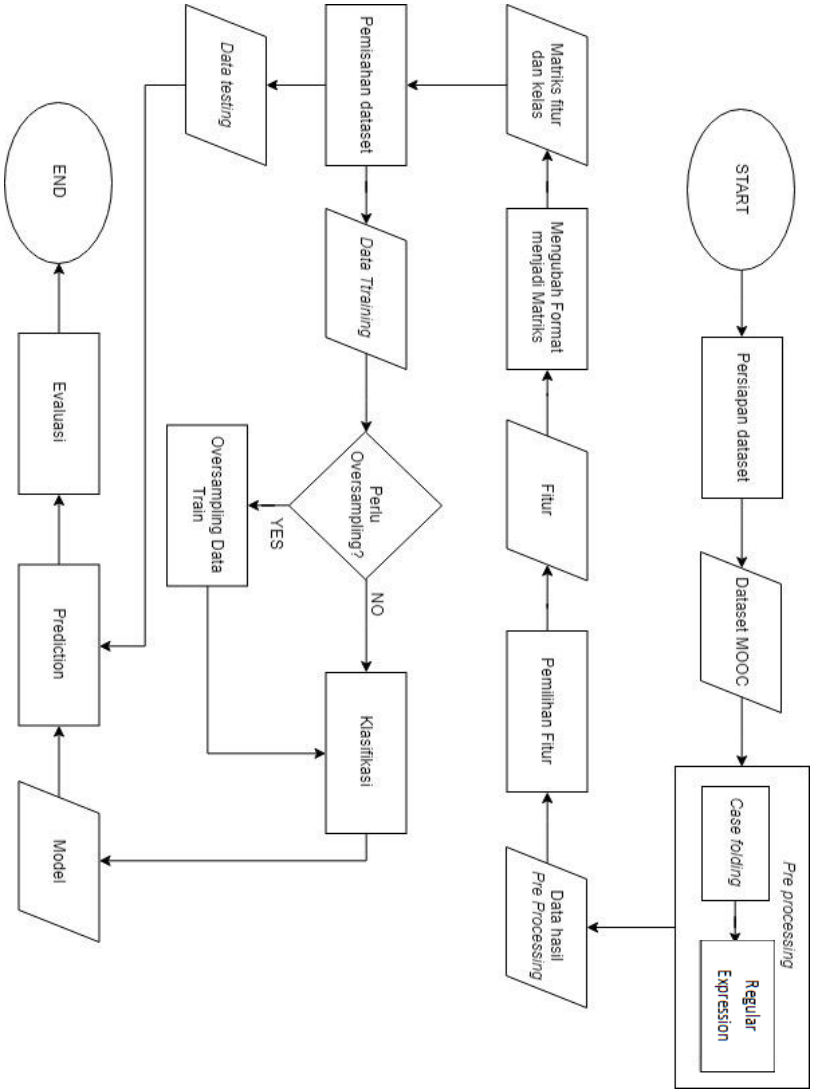
Pada bab ini akan dijabarkan analisis dan perancangan sistem dari tugas akhir yang meliputi tahap-tahap penyelesaian tugas akhir dan algoritma-algoritma yang mendukungnya secara khusus.

3.1. Analisis Metode Secara Umum

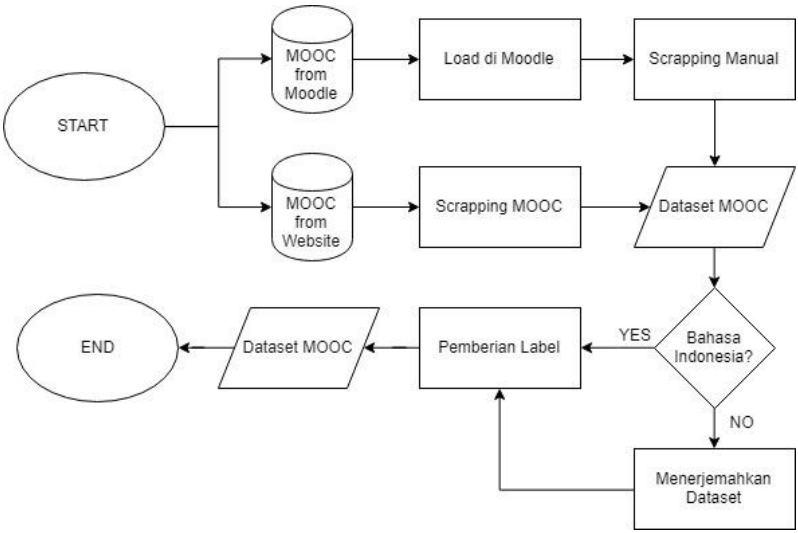
Pada tugas akhir ini akan dibangun suatu model pengklasifikasian yang dapat merekomendasikan pelabelan pada konten pembelajaran secara otomatis pada situs *Massive Open Online Course* yang terdiri dari empat kelas yaitu Nama Mata Kuliah, Deskripsi Mata Kuliah, Capaian Pembelajaran dan Pokok Bahasan. Tahapan proses yang akan dilakukan meliputi perancangan data dan perancangan proses. Adapun gambaran dari seluruh tahapan seperti pada diagram alir Gambar 3.1.

Pertama dilakukan tahap persiapan data seperti pada Gambar 3.2. Didapatkan dua sumber dataset MOOC, ada yang didapatkan dengan cara *scrapping* secara langsung dari website, dan juga dataset yang didapatkan *backup*-nya dari pihak ITS. Dataset MOOC yang berbentuk *backup* dengan format *file* .mbz perlu dilakukan *restore* data terlebih dahulu dan di *load* menggunakan LMS Moodle.

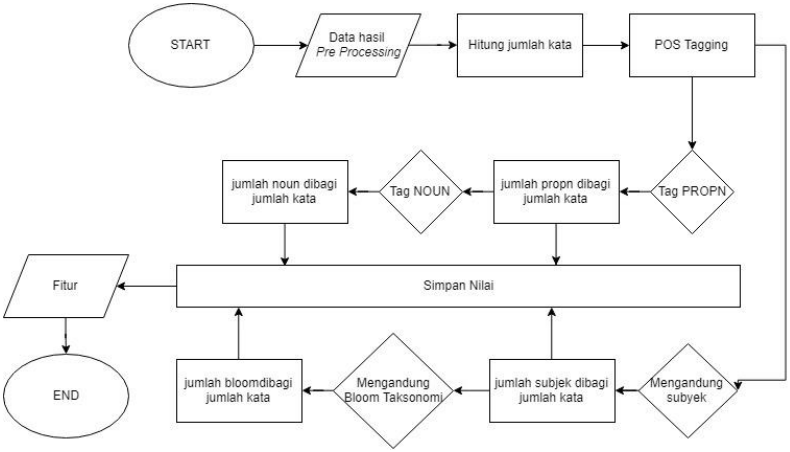
Output tahap persiapan dataset selanjutnya masuk ke tahapan *text mining*. Mulai dari *pre-processing* dengan melakukan *case folding lowercase* dan *regex.* Setelah itu dilakukan pemilihan fitur yang terdiri dari *rule* yang penulis buat dan fitur dari skor TF-IDF. Nantinya akan didapatkan lima fitur dari *rule* yang digambarkan pada Gambar 3.3 dan 150 fitur dari TF-IDF seperti yang dijelaskan secara rinci pada subbab 3.3.2.



Gambar 3.1 - Diagram alir seluruh tahapan



Gambar 3.2 - Diagram alir perancangan data



Gambar 3.3 - Diagram alir pemilihan fitur berdasarkan *rule*

Setelah *rule* telah ditentukan dan pembobotan TF-IDF didapatkan, dilakukan pemisahan dataset menjadi *data training* dan *data testing*. Pemisahan dataset dilakukan dengan cara perbandingan rasio dan dengan cara *cross validation*. Jika persebaran kelas pada *data training* tidak seimbang, dilakukan proses *Over Sampling* yang akan dijelaskan pada subbab 3.3.4. Kemudian dilakukan proses klasifikasi NLP tanpa *Machine Learning* menggunakan *rule* yang penulis tentukan dan NLP dengan *Machine Learning* menggunakan fitur yang didapatkan dengan metode *SVM, Random Forest*, dan *Naïve Bayes*. Model dari NLP tanpa *Machine Learning* serta ketiga metode dengan jenis *data train* yang terdiri dari *data train* hasil pemisahan dengan perbandingan rasio, *data train* hasil *over sampling* dan *data train* hasil *cross validation* disimpan untuk selanjutnya di prediksi dan dievaluasi menggunakan *data testing* serta diuji dengan masukan data baru.

3.2. Perancangan Data

Pada subbab ini akan dijelaskan mengenai perancangan data yang digunakan pada pembuatan tugas akhir. Data yang digunakan merupakan kalimat-kalimat penjelas pada mata kuliah di situs MOOC. Sedangkan untuk situs MOOC yang dipilih serta jumlah data mata kuliah yang diambil adalah: *E-learning* IF ITS tediri dari 0 data mata kuliah, Share ITS terdiri dari dua data mata kuliah, SPADA Daring Menristekdikti terdiri dari 18 mata kuliah, IndonesiaX terdiri dari satu data mata kuliah, KelasKita terdiri dari satu data mata kuliah, *Open Course Ware* Universitas Indonesia terdiri dari 4 data mata kuliah, dan EdX terdiri dari 491 data mata kuliah. Dalam satu data mata kuliah paling tidak terdiri dari nama mata kuliah dan deskripsi/capaian pembelajaran/pokok bahasan mata kuliah sehingga jumlah total data yang diambil sebanyak 1136 data. Data mata kuliah yang diambil dibatasi hanya yang memiliki kategori Ilmu Komputer. Hal tersebut berakibat tidak ada

data dari *E-learning* IF ITS yang dapat diambil karena dataset dari *E-learning* IF ITS yang penulis miliki tidak ada yang menjelaskan mata kuliah sebagai deskripsi kursus yang memenuhi kriteria, atau hanya sekedar judul dan informasi tugas serta materi perminggunya.

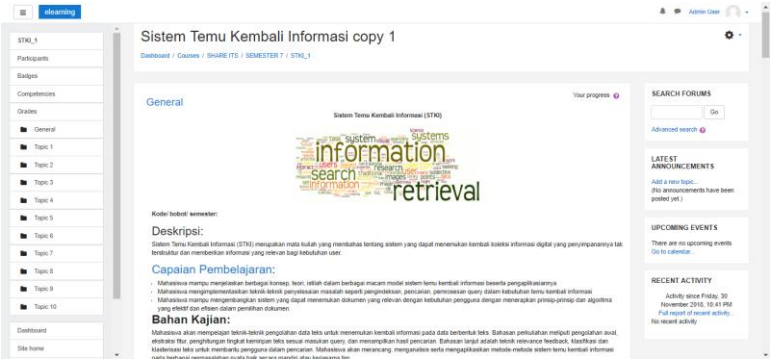
3.2.1 Pengambilan Data

Pengambilan data mata kuliah dilakukan dengan berbagai cara. Untuk data mata kuliah *E-learning IF ITS* didapatkan *backup* data melalui pihak jurusan Informatika ITS dalam format *file* .mbz, untuk data mata kuliah *Share* ITS didapatkan *backup* data melalui pihak Pascasarjana ITS dalam format .mbz, untuk MOOC seperti SPADA Daring Menristekdikti, IndonesiaX, KelasKita, dan *Open Course Ware* Universitas Indonesia diambil dengan proses *scraping* menggunakan *tools* Octoparse, dan untuk MOOC EdX diambil dengan proses *scraping* menggunakan *script* bahasa pemrograman Python dibantu pustaka *BeautifulSoup*.

3.2.2 *Load* Dataset MOOC di Moodle

Setelah data berformat .mbz yakni data mata kuliah dari *E-learning* IF ITS dan *Share* ITS didapatkan, file di *restore* menggunakan LMS Moodle 3.4. Lalu isi kursus atau mata kuliah dapat diambil melalui laman *Site Home* secara manual.

Gambar 3.4 - Laman *Site Home* Moodle



Gambar 3.5 - Laman Kursus di Moodle

3.2.3 Penerjemahan Data

Pada tugas akhir ini, data yang digunakan dibatasi hanya yang berbahasa Indonesia. Maka untuk penyeragaman proses data selanjutnya, diperlukan proses penerjemahan data terlebih dahulu pada dataset yang berasal dari situs EdX. Penerjemahan dataset dari EdX dilakukan dengan *script* bahasa pemrograman python dengan bantuan pustaka *TextBlob*. Dikarenakan pustaka *TextBlob case sensitive*, sebelum diterjemahkan dilakukan preproses *case*

folding lowercase terlebih dahulu dan *regex* untuk menghilangkan karakter lain selain karakter titik (.) dan huruf a sampai z. Hasil penerjemahan kalimat dari Gambar 3.6 menjadi Gambar 3.7.

\"This UX course provides an introduction to the fields of UX research and design. Learners will gain an understanding of what is involved in UX research, including conducting interviews, evaluating systems, and analyzing systems using principles of good design. Learners will also learn about the work involved in UX Design, including the generation of promising design solutions and the creation of prototypes at multiple levels of fidelity. By interleaving successive phases of UX Research and Design, learners will see how to learn from inevitable mistakes and improve towards a product with a great UX.This course is part of the User Experience (UX) Research and Design MicroMasters Program offered by MichiganX.\

Gambar 3.6 - Data EdX Sebelum Diterjemahkan

kursus ux ini memberikan pengantar untuk bidang penelitian dan desain ux. pelajar akan mendapatkan pemahaman tentang apa yang terlibat dalam penelitian ux termasuk melakukan wawancara mengevaluasi sistem dan menganalisis sistem menggunakan prinsip-prinsip desain yang baik. pelajar juga akan belajar tentang pekerjaan yang terlibat dalam desain ux termasuk generasi solusi desain menjanjikan dan penciptaan prototipe di berbagai tingkat kesetiaan. dengan interleaving fase sukses penelitian ux dan desain peserta didik akan melihat bagaimana belajar dari kesalahan yang tak terelakkan dan meningkatkan menuju produk dengan ux besar. Kursus ini adalah bagian dari pengalaman pengguna ux penelitian dan desain program micromasters yang ditawarkan oleh michiganx.

Gambar 3.7 - Data EdX Sesudah Diterjemahkan

3.2.4 Pemberian Label

Setelah dataset dari berbagai MOOC sudah terkumpul dan diterjemahkan, selanjutnya seluruh dataset disatukan menjadi satu *file* berformat .csv. Dataset di-*list* sesuai dari anotasi data pada situs

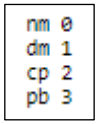
aslinya. Peran dataset tersebut kemudian dijadikan label yang terbagi menjadi 4 bagian, yaitu nama mata kuliah (nm), deskripsi mata kuliah (dm), capaian pembelajaran (cp) dan pokok bahasan (pb).

3.3. Perancangan Proses

Perancangan proses merupakan penjelasan proses apa saja yang dilakukan untuk tugas akhir ini.

3.3.1 *Pre processing*

Tahap *pre-processing* pada tugas akhir ini cukup sedarhana, dataset yang telah diberi label menjadi nm, dm, cp, dan pb diubah menjadi 0, 1, 2, 3 seperti pada Gambar 3.8 agar proses pengkalsifikasian kelas nantinya dikenal dalam bentuk angka. Proses perubahan tipe data tersebut dibantu dengan pustaka *Pandas* pada bahasa pemrograman Python.



Gambar 3.8 - Label Dalam Bentuk Angka

Selanjutnya setiap data dilakukan *case folding lowercase* agar setiap huruf seluruhnya menjadi huruf kecil dan dilakukan *regex* agar semua data hanya terdiri dari huruf a sampai z. Proses text *pre processing* lain seperti mengubah setiap kata menjadi kata dasar (*stemming*) dan membuang kata yang sering muncul (*stopwords*) tidak dilakukan karena tidak diperlukan. Sehingga data yang dihasilkan menjadi seperti pada Gambar 3.9.

0	algorithm and object oriented programming methods
1	mata kuliah algoritma dan metode object orient...
2	cp menggunakan sintaks dan fungsi di dalam ...
3	pengertian algoritma dan pemrograman perbe...
4	basis data
5	mata kuliah ini menjelaskan tentang konsep dan...
6	konsep basis data objek basis data relasiona...
7	data center management menggunakan docker cont...
8	process isolation diperlukan dalam rangka memb...
9	siswa mampu memahami cara kerja docker cont...
10	instalasi docker docker compose pengelo...

Gambar 3.9 - Data Hasil Pre Processing

3.3.2 Pemilihan Fitur

Setelah tahap *pre processing*, tahap selanjutnya adalah proses penentuan fitur berdasarkan *rule* dengan memberikan nilai pada kategori kata tertentu di setiap data dan dengan mencari pembobotan *term*-nya menggunakan TF-IDF..

3.3.2.1 Berdasarkan *Rule*

Gambaran proses pemilihan fitur berdasar *rule* dapat dilihat pada Gambar 3.3. Pertama dengan menghitung jumlah kata pada setiap data. Jumlah kata berpengaruh pada pengklasifikasian karena setiap kelas memiliki rata-rata komposisi kata yang sama dengan jumlah kata yang berbeda. Seperti antara kelas Nama Mata Kuliah dan Pokok Bahasan, kelas tersebut sebagian besar terdiri dari kata benda (*Noun*) namun memilki jumlah kata dengan rata-rata yang berbeda cukup jauh. Tabel 3.1 menunjukan perbandingan nilai maksimum, minimum, dan rata-rata jumlah kata pada setiap kelas.

Tabel 3.1 - Jumlah Kata pada Tiap Kelas

	Maks	Min	Avg
nm	12	1	5.138298
dm	797	1	138.928
cp	191	2	38.24615
pb	99	1	43.5

Jika sudah didapat jumlah katanya, selanjutnya memberikan kelas kata menggunakan metode *POS Tagging*. Setiap kata diberi label kelas kata, dan diambil berapa banyak kata yang berlabel *Noun* atau *Proper Noun*. Tahap ini dilakukan karena diasumsikan data dalam kelas Nama Mata Kuliah sebagian besar tersusun dari kata benda (*Noun*) saja. Begitupun dengan kelas Pokok Bahasan. Dan untuk penandaan *Proper Noun* atau kata pengganti ditunjukan untuk mengidentifikasi kelas Deskripsi Mata Kuliah, karena kelas Deskripsi Mata Kuliah diasumsikan harus mengandung kata pengganti atau *Proper Noun*. Jumlah kata yang berlabel *Noun* dan *Proper Noun* dibagi dengan total jumlah kata dalam satu data. Sehingga dihasilkan angka antara 0 sampai 1 yang menjadi fitur.

Kelas Capaian Pembelajaran diasumsikan dapat diidentifikasi dengan keberadaan kata-kata yang terkandung dalam Taksonomi Bloom seperti pada tabel di Lampiran 1.b. Setiap kata yang terdaftar di Taksonomi Bloom dijumlahkan, lalu hasil penjumlahan dibagi dengan total jumlah kata dalam satu data. Hasilnya berupa angka antara 0 sampai 1 yang digunakan sebagai fitur.

Selain itu juga dibuat daftar yang berperan sebagai subyek kalimat yang tertera dalam Lampiran 1.c. Daftar yang dibuat oleh penulis diasumsikan sebagai penunjuk keterangan kata subyek pada data mata kuliah yang sering disebutkan pada kelas Deskripsi

Mata Kuliah. Setiap kemunculan kata subyek ditandai bernilai 1. Lalu nilai tersebut dijadikan fitur.

Maka didapatkan lima fitur yaitu jumlah kata, kata yang mengandung *POS Tagging Noun*, kata yang mengandung *POS Tagging Proper Noun*, kata yang mengandung subyek, dan kata yang mengandung Taksonomi Bloom.

jk	label	naon	propn	subyek	takso
6	0	0.138889	0.166667	0	0.000000
51	1	0.013456	0.019608	1	0.019608
9	2	0.061728	0.222222	0	0.222222
52	3	0.014793	0.115385	0	0.000000
2	0	0.500000	0.000000	0	0.000000
86	1	0.007436	0.058140	1	0.023256
24	3	0.038194	0.083333	0	0.000000
6	0	0.138889	0.666667	0	0.166667
44	1	0.011880	0.090909	1	0.000000
7	2	0.102041	0.285714	1	0.000000
6	3	0.166667	0.500000	0	0.000000
3	0	0.333333	0.000000	0	0.000000
71	1	0.006546	0.000000	1	0.000000
21	2	0.022676	0.000000	1	0.095238
52	3	0.013314	0.019231	1	0.000000
2	0	0.500000	0.000000	0	0.000000
95	1	0.007645	0.084211	1	0.010526
131	2	0.004487	0.122137	1	0.145038

Gambar 3.10 - Fitur dan Label

3.3.2.2 Menggunakan TF-IDF

Fitur tambahan yang digunakan pada tugas akhir ini adalah dengan mendeteksi *term* yang sering muncul pada satu data lalu dihitung bobot *term*-nya seperti yang dijelaskan pada subbab 2.6. Pertama, pada satu data dihitung jumlah/frekuensi pada setiap *term*-nya. Jika suatu *term* pada data tersebut sering muncul, maka bobot yang didapatkan akan lebih besar. Lalu *term* tersebut dihitung juga bobotnya secara *invers*. Semakin sedikit persebaran *term* tersebut dari seluruh dataset/dokumen, maka bobotnya semakin besar. Hal ini berarti *term* tersebut memiliki keunikan tersendiri pada data-data tertentu.

Pembobotan TF-IDF dibantu dengan pustaka sklearn. Pada parameter fungsi TF-IDFVectorizer ditentukan range n-gram-nya dari 1 sampai 2 kata. Hal ini karena kata pada dataset tugas akhir memiliki kata-kata yang diulang seperti "elemen-elemen", "konsep-konsep", "protokol-protokol", dan lain sebagainya. Selain itu terdapat pula kata-kata yang berpasangan (frase), yang biasanya merupakan nama dari mata kuliah pada kursus seperti "Sistem Operasi", "Sistem Informasi", "Basis Data", "Data Mining", dan lain sebagainya.

Pada pengimplementasian nantinya, *max_feature* yang diambil pada tugas akhir ini sebanyak 150 fitur agar bobot *term* yang merepresentasikan data cukup lengkap.

3.3.3 Pemisahan Dataset

Dataset dipisah menjadi *data testing* dan *data training*. Pemisahan dataset dimaksudkan agar model yang diperoleh memiliki generalisasi yang baik dalam melakukan klasifikasi data.

Dataset dipisah menggunakan metode *stratify* agar semua kelas masuk sebagai *data training* maupun *data testing* dengan perbandingan 67% untuk *data training* dan 33% *data testing*. Proses pemisahan dataset dibantu dengan pustaka *train_test_split* dari *scikit-klearn*.

Selain itu pemisahan dataset juga dicoba dengan metode *Cross Validation*. Dataset di bagi menjadi 10 bagian atau yang disebut dengan *fold*, 9 *fold* sebagai *data training* dan 1 *fold* sebagai *data testing*. Pembagian dataset dengan *Cross Validation* dilakukan dengan menggunakan fungsi *StratifiedKFold()* pada *classifier* dari pustaka *scikit-klearn*.

3.3.4 *Oversampling Data Train*

Hasil pemisahan *data training* dapat dilihat pada Tabel 3.2.

Tabel 3.2 - Persebaran Data Training

Kelas	Anggota
Nama Mata Kuliah	252
Deskripsi Mata Kuliah	251
Capaian Pembelajaran	245
Pokok Bahasan	13

Berdasarkan Tabel 3.2. maka dapat dikatakan bahwa persebaran data tidak seimbang karena kelas dengan anggota paling sedikit tidak mencapai 1/10 jumlah kelas lain. Karena hal itu, seperti yang dijelaskan pada subbab 2.10, penulis mengusulkan metode *Over Sampling* untuk diuji coba dan membandingkan hasil evaluasi. Metode *Over Sampling* yang akan diuji coba adalah SMOTE dan ADASYN.

3.3.5 Klasifikasi

Metode klasifkasi yang digunakan pada tugas akhir ini antara lain *Support Vector Machine (SVM), Random Forest,* dan *Naïve Bayes*. Ketiga metode tersebut selanjutnya akan dibandingkan dan dipilih hasil model pengklasifikasian terbaik untuk menjawab rumusan masalah tugas akhir ini, yaitu menentukan anotasi detail konten kursus pada MOOC dengan menemukan metode klasifikasi mana yang paling baik.

Penentuan SVM sebagai metode klasifikasi karena SVM dinilai memiliki akurasi yang baik dengan bekerja pada sampel data yang relatif kecil, *Random Forest* dinilai sebagai salah satu metode klasifikasi terbaik karena memiliki banyak pohon keputusan sebagai penentu keputusan, dan Naïve Bayes dipilih karena cepat dan mudah diimplementasikan. Selain itu ketiga

metode tersebut juga cocok dalam menyelesaikan permasalahan *multiclass*.

3.3.6 *NLP Tanpa Machine Learning*

Pada tugas akhir ini juga akan dilakukan pengujian dengan metode *Natural Language Processing* tanpa *Machine Learning*, artinya metode pengklasifikasian hanya berdasarkan pada aturan (*rule*) yang penulis tetapkan. Hal ini bertujuan untuk membandingkan ketepatan pada hasil dari masukan data baru terhadap metode pengklasifikasian NLP dengan *Machine Learning* seperti *Random Forest, SVM,* dan *Naïve Bayes*.

Adapun *rule* untuk menentukan klasifikasi pada metode tanpa *Machine Learning* dibuat berdasarkan pengamatan karakter dan struktur penulisan keterangan mata kuliah pada MOOC. Karakter yang selanjutnya dibuat sebagai aturan terdapat pada Tabel 3.3.

Tabel 3.3 - Karakteristik dataset

No	Label	Karakteristik
1	Nama Mata Kuliah	- Mayoritas mengandung kata benda (*Noun*) - Jumlah kata relatif sedikit (maksimal 12 kata)
2	Deskripsi Mata Kuliah	- Jumlah kata relatif banyak (rata-rata 138 kata) - Mengandung kalimat subyek atau kata pengganti (*Proper Noun*)
3	Capaian Pembelajaran	- Mengandung kata yang terdaftar pada Taksonomi Bloom

4	Pokok Bahasan	- Mayoritas mengandung kata benda (*Noun*) - Jumlah kata relatif lebih banyak (rata-rata 43 kata)

Untuk nama mata kuliah diasumsikan hanya mengandung kata benda dan terdiri dari jumlah kata yang relatif sedikit. Deskripsi mata kuliah berisi keterangan mata kuliah yang cukup lengkap hingga memiliki jumlah kata yang relatif lebih banyak dan biasanya mengandung kalimat subjek atau *proper noun*. Capaian pembelajaran merupakan harapan pengajar atau kurikulum kepada mahasiswa atau murid terhadap materi yang diajarkan, maka kalimat pada capaian pembelajaran biasanya mengandung kata-kata dari taksonomi bloom sebagai parameter ketercapaian pembelajaran. Dan terakhir pokok bahasan, biasanya hanya berisi materi-materi yang akan diajarkan dalam perkuliahan. Maka pokok bahasan biasanya hanya mengandung kata kerja dan perbedaannya dengan nama mata kuliah, pokok bahasan biasanya mengandung jumlah kata yang relatif lebih banyak.

3.4. Evaluasi dan Uji Coba

Pada tahap evaluasi hasil prediksi model dibandingkan dengan *groundtruth*. Agar mempermudah, dibuat sebuah *confusion matrix* untuk mendapatkan nilai akurasi, *recall*, *precision*, dan *F-Measure* dari setiap evaluasi model.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dijelaskan implementasi yang dilakukan selama tugas akhir. Bab ini juga akan merinci perangkat serta *tools* yang digunakan pada tugas akhir ini beserta langkah-langkah pengerjaannya.

4.1. Lingkungan Uji Coba

Pada tugas akhir ini, digunakan beberapa perangkat serta *tools* yang membantu dalam pengerjaan tugas akhir. Perangkat keras dan sistem operasi yang digunakan selama tugas akhir adalah:

- Intel Core i7-7500U @ 2.7GHz (4 CPUs)
- NVIDIA GeForce 930MX 4 GB
- 4 GB of RAM
- Windows 10 Home 64-bit

Sedangkan untuk *tools* yang dipakai selama tugas akhir diuraikan pada Tabel 4.1.

Tabel 4.1 - Tools yang digunakan pada tugas akhir

No.	Tools	Deskripsi
1	Python	Bahasa Python digunakan untuk menangani *task Natural Language Processing* (NLP) dan *Machine Learning*.
2	BeautifulSoup	Pustaka untuk mengekstrak data dari HTML dan XML. Memiliki kemampuan untuk menavigasikan bagian yang perlu diekstrak.
3	Octoparse	Aplikasi untuk *scraping* halaman web
3	TextBlob	Pustaka untuk memproses data tekstual. TextBlob pada tugas akhir ini digunakan untuk menerjemahkan teks.
4	Polyglot	Pustaka untuk melakukan *Pos Tagging*

5	re	Pustaka untuk melakukan *regular expression.*
6	imblearn	Pustaka untuk menangani data yang tidak seimbang dengan cara *over sampling.*
7	Scikit-Learn	Pustaka ini digunakan untuk pembagian dataset baik dengan *splitting dataset* dan *cross validation*. Juga digunakan untuk melakukan evaluasi model. Baik *accuracy*, *precision*, *recall*, maupun *F-Measure*.
8	Jupyter Notebook, Sublime, Ms. Excel, dan Ms. Word	Aplikasi untuk pengolahan data dan penulisan program.

4.2. Persiapan Dataset

Pada subab ini akan dijelaskan mengenai persiapan dataset sebelum masuk ke tahap implementasi.

4.2.1. Pengambilan Data

Seperti yang dijelaskan pada subbab 3.2.1, dataset didapatkan dengan berbagai cara. Penjelasan pengambilan data terdapat pada Tabel 4.2.

Untuk dataset mata kuliah E-learning IF ITS didapatkan backup data melalui pihak jurusan Informatika ITS dalam format file .mbz, untuk data mata kuliah Share ITS didapatkan backup data melalui pihak Pascasarjana ITS dalam format .mbz, untuk MOOC seperti SPADA Daring Menristekdikti, IndonesiaX, KelasKita, dan Open Course Ware Universitas Indonesia diambil dengan proses *scraping* menggunakan *tools* Octoparse. Dan untuk dataset mata kuliah dari website EdX diambil dengan proses *scraping* menggunakan script Python dengan pustaka BeautifulSoup.

Tabel 4.2 - Pengambilan data

No	Sumber Dataset	Pengambilan Data
1	E-learning IF ITS dan Share ITS	*Backup* data dari pihak jurusan Informatika ITS dan Pasca sarjana ITS dalam format *file* .mbz
2	SPADA Daring Menristekdikti, IndonesiaX, KelasKita, dan *Open Course Ware* Universitas Indonesia	*Scraping* menggunakan *tools* Octoparse
3	EdX	*Scraping* menggunakan script Python dengan pustaka BeautifulSoup

Pada subbab ini akan dijelaskan proses pengambilan data untuk dataset EdX. Dataset EdX berasal dari website edx.org dan data adalah kursus dengan kategori *Computer Science*. Proses *scraping* menggunakan pustaka BeautifulSoup untuk mem-*parsing tag* pada HTML.

```
1. url = 'https://www.edx.org/course/?subject=Computer%20Science&language=English'
2.
3. browser = webdriver.Chrome('E:/ITS - Teknik Informatika/SEMESTER 7/Tugas-Akhir/Master/App/chromedriver.exe')
4. browser.get(url)
5.
6. SCROLL_PAUSE_TIME = 4.5
7.
8. # Get scroll height
9. last_height = browser.execute_script("return document.body.scrollHeight")
10.
11.while True:
12.    # Scroll down to bottom
```

```
13.     browser.execute_script("window.scrollTo(0, document.
body.scrollHeight);")
14.
15.     # Wait to load page
16.     time.sleep(SCROLL_PAUSE_TIME)
17.
18.     # Calculate new scroll height and compare with last
scroll height
19.     new_height = browser.execute_script("return document
.body.scrollHeight")
20.     if new_height == last_height:
21.         break
22.     last_height = new_height
23.
24. soup = BeautifulSoup(browser.page_source,"html.parser")

25.
26. all_=[]
27. j = soup.find_all('a', class_='course-link')
28. for link in j:
29.     all_.append(link.get('href'))
30. all_
```

Kode Sumber 4.1 - *Parsing* URL halaman kursus

Halaman daftar kursus pada situs EdX memiliki karakter *infinite scroll*, seperti pada Kode Sumber 4.1, maka langkah pertama yang dilakukan adalah dengan mem-*parsing link-link* halaman detail kursus. Pertama dengan menginisiasikan halaman daftar kursus dengan tag *Computer Science* pada variabel url. Lalu proses pengambilan url dilakukan dengan menggunakan *webdriver* Chrome. `SCROLL_PAUSE_TIME` merupakan variabel untuk menentukan waktu tunggu pada sistem ketika *webdriver scrolling* secara otomatis. `last_height` merupakan variabel yang menginisiasikan batas akhir sistem melakukan *scrolling* secara otomatis. Dengan melakukan perulangan, sistem akan *scrolling* secara otomatis ketika halaman telah mencapai `last_height`.

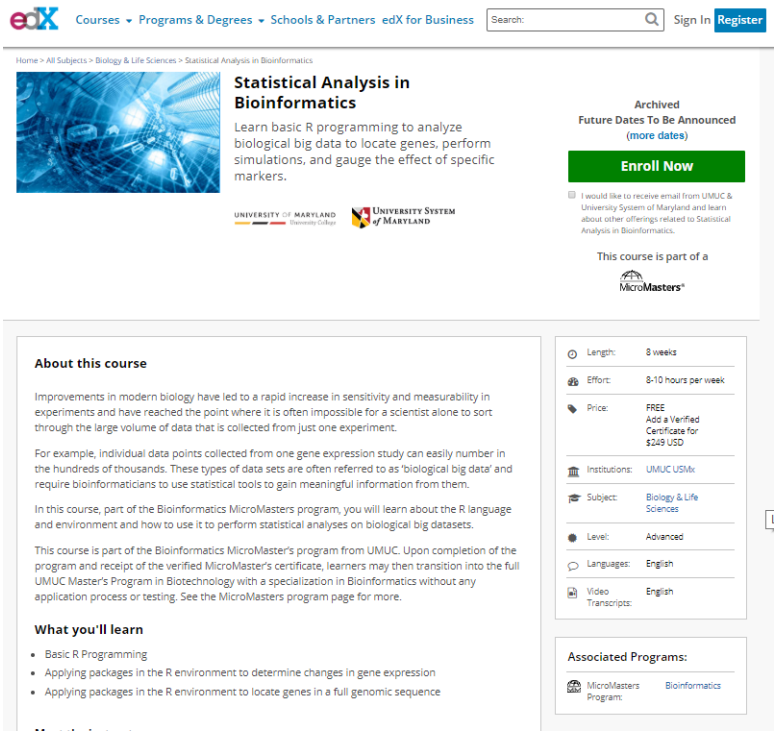
Jika *scrolling* sudah selesai, selanjutnya sistem mem-*parsing link* url halaman kursus dari daftar kursus dengan mencari *tag* <a> dengan class `course-link.` Lalu isi dari atribut `'href'` diambil isinya yang berisi *link-link* untuk alaman detail kursus.

```
1. hdr = {'User-Agent': 'Mozilla/5.0'}
2. req = Request(link,headers=hdr)
3. page = urlopen(req)
4. soup = BeautifulSoup(page)
5. html = str(soup)
6. d_start = html.find('"description":"')
7. d_end = html.find('",', d_start)
8. cp_start = html.find('"educationalOutcome":"')
9. cp_end = html.find('",', cp_start)
10.x={}
11.x['nama_matkul'] = str(soup.find('title'))
12.x['deskripsi_matkul'] = str(html[d_start+14:d_end+2])
13.x['capaian_pembelajaran'] = str(html[cp_start+21:cp_e
   nd+2])
14.matkul.append(x)
```

Kode Sumber 4.2 - *Parsing* detail kursus

Pada Kode Sumber 4.2 dijelaskan program dengan melakukan perulangan terhadap *link* yang sudah didapatkan. Selanjutnya masing-masing link dibuka dan di-*parsing* HTML dengan *tag* `<title>` untuk diekstrak sebagai judul mata kuliah, *class* `“description”` untuk diekstrak sebagai deskripsi mata kuliah dan *class* *“*`educationalOutcome`*”* untuk diekstrak sebagai capaian pembelajaran.

Gambar 4.1 menunjukan tampilan laman kursus pada website EdX. Setelah diekstrak, data disimpan dalam file berformat JSON. Contoh data yang diekstrak dalam file JSON ditunjukan pada Gambar 4.2.



Gambar 4.1 – Laman kursus EdX

{
"nama_matkul": "Statistical Analysis in Bioinformatics",
"deskripsi_matkul": "\"Improvements in modern biology have led to a rapid increase in sensitivity and measurability in experiments and have reached the point where it is often impossible for a scientist alone to sort through the large volume of data that is collected from just one experiment. \\nFor example, individual data points collected from one gene expression study can easily number in the hundreds of thousands. These types of data sets are often referred to as \\u2018biological big data\\u2019 and require bioinformaticians to use statistical tools to gain meaningful information from them.\\nIn this course, part of the Bioinformatics MicroMasters program, you will

learn about the R language and environment and how to use it to perform statistical analyses on biological big datasets.\",",
"capaian_pembelajaran": "\"Basic R Programming\\n\\tApplying packages in the R environment to determine changes in gene expression\\n\\tApplying packages in the R environment to locate genes in a full genomic sequence\\n\",",
"pokok_bahasan": ""
}

Gambar 4.2 - Hasil ekstraksi

4.2.2. Penerjemahan Data

Karena batasan pada tugas akhir ini adalah memproses data dengan menggunakan kursus Bahasa Indonesia, maka perlu dilakukan penerjemahan pada data yang didapatkan dari situs EdX. Penerjemahan dilakukan dengan menggunakan pustaka *TextBlob*. Sebelum diterjemahkan, dilakukan pra proses data terlebih dahulu dengan menghilangkan karakter selain angka, huruf dan karakter titik (.) dengan *regex*, serta menyetarakan semua huruf menjadi *lowercase* dengan *case folding*. Kode program pra proses tertera pada Kode Sumber 4.3.

```
1. cf = []
2. for contenttrans in tqdm(data_mooc):
3.     contentLower= {}
4.     contentClear = {}
5.     contentLower['nama_matkul'] = contenttrans['nama_
   matkul'].lower()
6.     contentLower['deskripsi_matkul'] = contenttrans['
   deskripsi_matkul'].lower()
7.     contentLower['capaian_pembelajaran'] = contenttra
   ns['capaian_pembelajaran'].lower()
8.     contentLower['pokok_bahasan'] = contenttrans['pok
   ok_bahasan'].lower()
9.     contentClear['nama_matkul'] = re.sub(r'[^ a-zA-
   Z0-9.+]', ' ', str(contentLower['nama_matkul']))
10.     contentClear['deskripsi_matkul'] = re.sub(r'[^ a-
   zA-Z0-
   9.+]', ' ', str(contentLower['deskripsi_matkul']))
```

```
11.     contentClear['capaian_pembelajaran'] = re.sub(r'[
    ^ a-zA-Z0-
    9.+]', ' ', str(contentLower['capaian_pembelajaran'])
    )
12.     contentClear['pokok_bahasan'] = re.sub(r'[^ a-zA-
    Z0-9.+]', ' ', str(contentLower['pokok_bahasan']))
13.     cf.append(contentClear)
```

Kode Sumber 4.3 - Pra Proses Penerjemahan

Penerjemahan dilakukan dengan mendeteksi dahulu Bahasa yang tertera pada teks. Variabel `dl_nm` menunjukan Bahasa yang terdeteksi pada teks. Lalu dilakukan penerjemahan dengan fungsi `translate()` dari pustaka TextBlob sebagai fungsi penerjemah dengan parameter `from_lang` yang menunjukan bahasa yang terdeteksi ke parameter `to` yang menunjukan Bahasa yang akan di terjemahkan. Kode program penerjemahan tertera pada Kode Sumber 4.4.

```
1.  matkul = []
2.  for index in range (26, len(cf)):
3.      content=cf[index]
4.      mooc = {}
5.      print(content['nama_matkul'])
6.      dl_nm = TextBlob(content['nama_matkul']).detect_l
    anguage()
7.      mooc['nama_matkul'] = TextBlob(content['nama_matk
    ul']).translate(from_lang = dl_nm, to='id')
8.      mooc['deskripsi_matkul'] = TextBlob(content['desk
    ripsi_matkul']).translate(from_lang =dl_nm, to='id')

9.      mooc['capaian_pembelajaran'] = TextBlob(content['
    capaian_pembelajaran']).translate(from_lang =dl_nm, t
    o='id')
10.     matkul.append(mooc)
11.     print(matkul)
```

Kode Sumber 4.4 - Penerjemahan

4.3. Proses Implementasi

Implementasi proses dilakukan berdasarkan perancangan proses yang dijelaskan pada bab Analisis dan Perancangan Sistem.

4.3.1. *Pre-processing*

Subbab ini akan menjelaskan tahap *pre processing* pada seluruh dataset yang telah didapatkan dan sudah digabung dalam satu *file* berformat csv.

Langkah pertama yang dilakukan adalah mengubah label dataset menjadi angka. Dataset yang telah diberi label menjadi nm, dm, cp, dan pb diubah menjadi 0, 1, 2, 3 seperti yang ditunjukan pada Kode Sumber 4.5.

```
1. label = train_df['type'].unique()
2. label_to_id = {}
3. assign_id = 0
4. for lbl in label:
5.     label_to_id[lbl] = assign_id
6.     assign_id += 1
7.
8. def get_label_id(label):
9.     return label_to_id[label]
10.
11. train_df['label_id'] = train_df['type'].map(get_labe
    l_id)
```

Kode Sumber 4.5 - Mengubah Label Menjadi Angka

Pada Kode Sumber 4.5 ditujukan variabel `label` untuk mengambil data yang unik dari kolom `type` (label pada dataset). Pada setiap data yang unik, dilakukan perulangan untuk diinisiasikan sebagai angka yang ditujukan pada variabel `label_to_id`. Kemudian dibuat fungsi `get_label_id()` dengan parameter `label` untuk menyimpan label yang sudah berbentuk angka ke kolom baru yaitu `label_id`.

Selanjutnya *pre processing* untuk membersihkan kata agar semua kata memiliki format yang seragam dan lebih mudah untuk diproses pada tahap selanjutnya. Fungsi cleanText() di-*parsing* dengan parameter `text` yang berisi kalimat pada dataset. Kemudian dengan menggunakan pustaka re dilakukan *regex* untuk menghilangkan karakter lain selain huruf a sampai z. Setelah itu dilakukan *case folding* dengan mengubah semua teks menjadi huruf kecil (*lowercase*) dengan fugsi `lower()` yang tersedia pada Python. *Pre-processing* untuk membersihkan kata terdapat pada Kode Sumber 4.6.

```
1. def cleanText (text):
2.     textClear = re.sub ('[^ a-zA-
   Z]', ' ',text.lower())
3.     return textClear
```

Kode Sumber 4.6 - Pembersihan kata

4.3.2. Pemilihan Fitur berdasar *Rule*

Selanjutnya dibuat sebuah fungsi yaitu `getPosTag()` dengan parameter `tagText` yang berisi teks yang sudah dibersihkan pada tahap *pre processing* sebelumnya.

Seperti yang tertera pada Kode Sumber 4.7, langkah pertama yang dilakukan pada tahap pemilihan fitur adalah dengan menghitung ada berapa kata yang terdapat dalam satu teks, lalu disimpan pada variabel `jmlKata`.

Selanjutnya menentukan kelas kata pada setiap kata dalam teks. Penetuan kelas kata menggunakan metode *POS Tagging* dengan bantuan pustaka Polyglot. Variabel posTag digunakan untuk menentukan teks apa yang digunakan dan bahasa apa yang akan diproses dalam penentuan kelas kata. Lalu dibuat *array* `listTag` untuk menentukan kelas kata yang akan digunakan dalam program tugas akhir ini, yaitu kata benda (NOUN) dan kata

pengganti (PROPN). Alasan kelas kata yang dipilih adalah *Noun* dan *Proper Noun* terdapat pada subbab 3.3.2.1.

Terdapat variabel `bloom` yang berisi daftar kata pada taksonomi bloom yang disimpan pada file `taksonomi.txt`. Lalu ada juga variabel `subjek` yang berisi daftar kata bertipe "subjek" yang biasa digunakan pada MOOC yang disimpan pada file `subjek.txt`.

```
1. def getPosTag(tagText):
2.     jmlKata = len(tagText.split())
3.     propn = 0
4.     takso = 0
5.     noun = 0
6.     subyek = 0
7.     posTag=Text(tagText, hint_language_code='id')
8.
9.     #menambahkan rule
10.    listTag=['NOUN', 'PROPN']
11.    bloom = open("data/taksonomi.txt").read().splitli
   nes()
12.    subjek = open("data/subjek.txt").read().splitline
   s()
13.
14.    for kata in posTag.pos_tags:
15.        if kata[0] in subjek:
16.            subyek = 1
17.        if kata[0] in bloom:
18.            takso += 1
19.        if kata[1] in listTag:
20.            if kata[1] == 'PROPN':
21.                propn += 1
22.            if kata[1] == 'NOUN':
23.                noun += 1
24.    propn = (propn/jmlKata)
25.    takso = (takso/jmlKata)
26.    noun = (noun/jmlKata)
27.    return propn, takso, noun, subyek, jmlKata
```

Kode Sumber 4.7 - POS Tagging dan Pemilihan Fitur

Langkah selanjutnya adalah pemilihan fitur dengan melakukan perulangan terhadap variabel `posTag` sambil memanggil properti `pos_tags` dari polyglot. Pada setiap kata yang sudah memiliki kelas kata, dilihat apakah kata tersebut juga mengandung kata yang terdapat dalam daftar `subjek`, jika iya simpan nilai 1 pada variabel `subyek` seperti pada Kode Sumber 4.7 baris 15-16. Selanjutnya dilihat apakah kata tersebut juga mengandung kata yang terdapat dalam daftar `bloom`, jika iya lakukan perhitungan secara *increment* dan simpan nilai pada variabel `bloom` seperti pada Kode Sumber 4.7 baris 17-18. Lalu dilakukan juga perhitungan secara *increment* dari kelas kata yang sudah ditandai. Jika mengandung kelas kata PROPN, *increment* variabel propn. Jika mengandung kelas kata NOUN, *increment* variabel naon.

Setelah itu dilakukan perbandingan jumlah kata yang memiliki kelas kata PROPN dan NOUN terhadap keseluruhan jumlah kata serta dilakukan perbandingan kata yang mengandung taksonomi bloom terhadap keseluruhan jumlah kata. Nilai tersebut disimpan masing-masing pada variabel `propn`, `takso`, dan `noun` seperti pada Kode Sumber 4.7 baris 24-26.

Akhirnya didapat lima buah fitur, yaitu jumlah kata pada variabel `jmlKata`, kata yang mengandung bloom taksonomi pada variabel `takso`, perbandingan kata yang mengandung subjek pada variabel `subyek`, perbandingan kata yang memiliki kelas kata PROPN pada variabel `propn`, dan perbandingan kata yang memiliki kelas kata NOUN pada variabel `noun`.

4.3.3. Pemilihan Fitur berdasar TF-IDF

Format data dari langkah sebelumnya direpresentasikan sebagai DataFrame dari pustaka pandas, maka pada Kode Sumber 4.8 diubah ke representasi matriks array pustaka numpy untuk mempermudah proses selanjutnya.

```
1. kalimat = pd.DataFrame(deskr)['kalimat'].as_matrix()
2. label = pd.DataFrame(deskr)['label'].as_matrix()
```

Kode Sumber 4.8 - Mengubah ke bentuk matriks

Transformasikan teks ke fitur dalam bentuk *vector* yang digunakan sebagai *input estimator* menggunakan fungsi `TfidfVectorizer()` dari pustaka *scikit-learn*. Pada variabel `tfidf`, atur parameter yang digunakan pada fungsi `TfidfVectorizer()` dengan mengatur `sublinear_tf` bernilai *True* sehingga rumus IDF yang digunakan memiliki nilai 1 pada numerator dan denumerator untuk menghindari pembangian bernilai 0, `min_df` sebagai threshold untuk mengabaikan kosa kata yang memiliki frekuensi dokumen lebih rendah dari ambang batas, melakukan normalisasi *term vector* dengan mengatur `norm` senilai 12 sesuai standar pustaka *scikit-learn*, memberikan batas *n-gram* dari 1 sampai 2 dengan mengatur `ngram_range` senilai 1 sampai 2, dan terakhir memberi batasan jumlah fitur yang diambil sebanyak 150 fitur dari bobot tertinggi dengan mengatur `max_features` senilai 150. Pengaturan fungsi `TfidfVectorizer()` terdapat pada Kode Sumber 4.9 baris 1.

Variabel `tv` memanggil fungsi `fit_transform()` dengan parameter dari matiks `kalimat.` Maka didapatkan model yang disimpan pada variabel `tv` yang selanjutnya disimpan seperti pada Kode Sumber 4.9 baris 3-4. Selanjutnya kembalikan variabel `tv` menjadi array pada variabel `features` untuk didapatkan nama fiturnya dan selanjutnya digabungkan dengan fitur berdasarkan *rule* yang sebelumnya telah didapatkan.

```
1. tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5,
   norm='l2', encoding='latin-
   1', ngram_range=(1, 2), max_features=150)
2. tv = tfidf.fit_transform(kalimat)
3. with open('data/model/vectorizer.pkl', 'wb') as tvc:
```

```
    pickle.dump(tfidf, tvc)
features = tv.toarray()
labels = label
features.shape
featureName=tfidf.get_feature_names()
```

Kode Sumber 4.9 - Fitur TF-IDF

Langkah selanjutnya adalah menggabungkan fitur TF-IDF dengan fitur berdasarkan rule pada DataFrame `deskr_p`, lalu simpan seluruh fitur pada file berformat csv sebagai backup. Penggabungan fitur TF-IDF terdapat pada Kode Sumber 4.10.

```
for index,ftr in enumerate(featureName):
    deskr_p[ftr]=features[:,index]

deskr_p.to_csv('data/fitur.csv')
```

Kode Sumber 4.10 - Menggabungkan fitur TF-IDF

4.3.4. Pemisahan Dataset

Setelah *pre processing* dan pemilihan fitur, didapatkan variabel `deskr_p` yang disimpan dalam bentuk DataFrame. Selanjutnya dilakukan pemisahan dataset untuk memisahkan data *training* sebagai data latih yang digunakan untuk menghasilkan model dan data *testing* untuk menguji hasil model untuk nantinya dijadikan bahan evaluasi.

```
x=deskr_p.drop(columns=['label', 'kalimat'])
y=deskr_p['label']
```

Kode Sumber 4.11 - Pemisahan Fitur dan Kelas

Kode Sumber 4.11 menjelaskan penentuan `x` sebagai fitur dan `y` sebagai kelas. Fitur yang diambil terdiri dari lima kolom/fitur seperti yang sudah dijelaskan pada tahap pemilihan fitur

berdasarkan *rule* dan 150 fitur dari TF-IDF. Kelas terdiri dari 4 kelas yang sudah diubah bentuknya menjadi angka pada penjelasan tahap *pre processing*.

Pemisahan dataset dilakukan dengan bantuan pustaka sklearn. Pembagian data terbagi menjadi 33% data *testing* dan 67% data *training*. Terdapat parameter yang digunakan yaitu parameter `random_state` yang diinisiasikan 0, agar setiap men-*generate* kembali data baru, nilai data training dan data testing tidak berubah. Lalu digunakan juga parameter *stratify* terhadap nilai y, agar pembagian antar kelas pada data training dan data testing merata. Implementasi pemisahan dataset tertera pada Kode Sumber 4.12.

```
1. X_train, X_test, y_train, y_test = train_test_split(x
   , y, test_size=0.33, random_state=0, stratify=y)
```

Kode Sumber 4.12 - Pemisahan Dataset

4.3.5. *Oversampling Data Train*

Karena persebaran kelas tidak seimbang seperti yang diuraikan pada Tabel 2.1, maka setelah pemisahan dataset dilakukan penyeimbangan *data training* dengan metode *over sampling.*

Untuk bahan evaluasi nantinya, dipilih dua metode over sampling yaitu SMOTE seperti pada Kode Sumber 4.13 dan ADASYN seperti pada Kode Sumber 4.14.

```
1. sm = SMOTE(random_state=0)
2. ftr_smt, cls_smt = sm.fit_sample(X_train, y_train)
```

Kode Sumber 4.13 - Over Sampling pada SMOTE

```
1. ada = ADASYN(sampling_strategy='minority',random_stat
   e=0)
```

```
2. ftr_ada, cls_ada = ada.fit_resample(X_train, y_train
```

Kode Sumber 4.14 - - Over Sampling pada ADASYN

Perbandingan hasil metode over sampling antara SMOTE dan ADASYN pada data *training* terdapat pada Tabel 4.3.

Tabel 4.3 - Hasil *over sampling*

Kelas	0	1	2	3
Sebelum Over Sampling	252	251	245	13
SMOTE	252	252	252	252
ADASYN	252	252	251	245

4.3.6. Klasifikasi *Random Forest*

Implementasi pengklasifikasian dengan *Random Forest Classifier* dibantu dengan pustaka sklearn. Parameter yang diatur pada metode ini antara lain dengan menentukan `n_estimators` sebagai jumlah pohon sebanyak 100. Semakin bertambah jumlah pohon, menghasilkan performa yang lebih baik dan membuat prediksi lebih stabil. Meskipun semakin memperlambat proses komputasi. Selain itu parameter `random_state` diatur bernilai 0. Pengaturan parameter pada fungsi `RandomForestClassifier` ini digunakan untuk semua percobaan pengklasifikasian *Random Forest* dan dapat dilihat pada Kode Sumber 4.15.

```
1. from sklearn.ensemble import RandomForestClassifier
2. RandomForest = RandomForestClassifier(n_estimators=10
   0, random_state=0)
```

Kode Sumber 4.15 - Pengaturan Parameter *RandomForestClassifier*

4.3.6.1. Klasifikasi dengan Pembagian Dataset Berdasar Rasio

Implementasi *Random Forest* dengan pemisahan dataset berdasar rasio sehingga didapatkan *data training* sebanyak 67% dimasukan dalam parameter *method* `fit` untuk dilatih seperti pada baris ke-1 di Kode Sumber 4.16. Selanjutnya dilakukan prediksi dengan *data testing* seperti pada baris ke-4, hingga didapatkan akurasi, presisi, *recall* dan *f score*-nya. Langkah selanjutnya adalah penyimpanan model seperti pada baris 8-9.

```
1. RF = RandomForest.fit(X_train, y_train)
2.
3. y_RF = RF.predict(X_test)
4. print('Akurasi', accuracy_score(y_test, y_RF))
5. target_names = ['class 0', 'class 1', 'class 2', 'cla
   ss 3']
6. print(classification_report(y_test, y_RF, target_name
   s=target_names))
7.
8. with open('data/model/rf_model.pkl', 'wb') as rft:
9.     pickle.dump(RF, rft)
```

Kode Sumber 4.16 - Random Forest Dengan Pembagian Dataset 33:67

4.3.6.2. Klasifikasi dengan Cross Validation

Implementasi *Random Forest* dengan pembagian dataset secara cross validation dengan fungsi `StratifiedKFold` bernilai `n_splits` = 10. Sehingga didapatkan *data testing* sebanyak 1/10 bagian dan *data training* sebanyak 9/10 bagian. Selanjutnya dengan perulangan senilai jumlah k, variabel `x` dan `y` di split sebanyak k bagian. Lalu dilakukan pemisahan antara *data training* dan *data testing* seperti pada baris 5-6 Kode Sumber 4.17. *Data training* kemudian dilatih pada *method* `fit` seperti pada baris ke-8 di Kode Sumber 4.17. Selanjutnya dilakukan prediksi dengan *data testing* seperti pada baris ke-9, hingga didapatkan akurasi, presisi, *recall* dan *f score*-nya. Langkah selanjutnya adalah

penyimpanan model dengan nilai akurasi terbaik seperti pada baris 16-19.

```
1. skf = StratifiedKFold(n_splits=10)
2. currentMax = 0
3.
4. for train_index, test_index in skf.split(x, y):
5.     X_train_cv, X_test_cv = x[train_index], x[test_in
   dex]
6.     y_train_cv, y_test_cv = y[train_index], y[test_in
   dex]
7.
8.     RFc = RandomForest.fit(X_train_cv, y_train_cv)
9.     y_RF_cv = RFc.predict(X_test_cv)
10.        akurasi = accuracy_score(y_test_cv, y_RF_cv)

11.
12.        print ('Akurasi', akurasi)
13.        target_names = ['class 0', 'class 1', 'class
   2', 'class 3']
14.        print(classification_report(y_test_cv, y_RF_c
   v, target_names=target_names))
15.
16.        if(currentMax  < akurasi):
17.            currentMax = akurasi
18.            with open('data/model/rf_cv_model.pkl', '
   wb') as rfc:
19.                pickle.dump(RFc, rfc)
```

Kode Sumber 4.17 - Random Forest Dengan Pembagian Dataset Cross Validation

4.3.6.3. Klasifikasi dengan Over Sampling SMOTE

Implementasi *Random Forest* dengan *data training* yang sudah di-*sampling* dengan metode *over sampling* SMOTE dimasukan menjadi parameter *method* `fit` untuk dilatih seperti pada baris ke-1 di Kode Sumber 4.18. Selanjutnya dilakukan prediksi dengan *data testing* seperti pada baris ke-3, hingga

didapatkan akurasi, presisi, *recall* dan *f score*-nya. Langkah selanjutnya adalah penyimpanan model seperti pada baris 8-9.

```
1. RF_smt = RandomForest.fit(ftr_smt, cls_smt)
2.
3. y_RF_smt = RF_smt.predict(X_test)
4. print('Akurasi', accuracy_score(y_test, y_RF_smt))
5. target_names = ['class 0', 'class 1', 'class 2', 'cla
   ss 3']
6. print(classification_report(y_test, y_RF_smt, target_
   names=target_names))
7.
8. with open('data/model/rf_smt_model.pkl', 'wb') as rfs
   :
9.     pickle.dump(RF_smt, rfs)
```

Kode Sumber 4.18 - Random Forest Dengan Over Sampling SMOTE

4.3.6.4. Klasifikasi dengan Over Sampling ADASYN

Implementasi *Random Forest* dengan *data training* yang sudah di-*sampling* dengan metode *over sampling* ADASYN dimasukan menjadi parameter *method* `fit` untuk dilatih seperti pada baris ke-1 di Kode Sumber 4.19. Selanjutnya dilakukan prediksi dengan *data testing* seperti pada baris ke-3, hingga didapatkan akurasi, presisi, *recall* dan *f score*-nya. Langkah selanjutnya adalah penyimpanan model seperti pada baris 8-9.

```
1. RF_ada = RandomForest.fit(ftr_ada, cls_ada)
2.
3. y_RF_ada = RF_ada.predict(X_test)
4. print(accuracy_score(y_test, y_RF_ada))
5. target_names = ['class 0', 'class 1', 'class 2', 'cla
   ss 3']
6. print(classification_report(y_test, y_RF_ada, target_
   names=target_names))
7.
8. with open('data/model/rf_ada_model.pkl', 'wb') as rfa
   :
```

```
9.      pickle.dump(RF_ada, rfa)
```

Kode Sumber 4.19 - Random Forest Dengan Over Sampling ADASYN

4.3.7. Klasifikasi *Support Vector Machine*

Implementasi pengklasifikasian dengan *Support Vector Machine* dibantu dengan pustaka sklearn. Parameter yang diatur pada metode ini antara lain dengan menentukan `kernel` sebagai linear dan gamma di-set *auto*. Pengaturan parameter pada fungsi `SVC` ini digunakan untuk semua percobaan pengklasifikasian SVM dan dapat dilihat pada Kode Sumber 4.20.

```
1. from sklearn import svm
2.
3. svm = svm.SVC(kernel='linear', gamma='auto')
4. svm
```

Kode Sumber 4.20 - Pengaturan Parameter SVM

4.3.7.1. Klasifikasi dengan Pembagian Dataset Berdasar Rasio

Implementasi *Support Vetor Machine* dengan pemisahan dataset berdasar rasio sehingga didapatkan *data training* sebanyak 67% dimasukan dalam parameter *method* `fit` untuk dilatih seperti pada baris ke-1 di Kode Sumber 4.21. Selanjutnya dilakukan prediksi dengan *data testing* seperti pada baris ke-3, hingga didapatkan akurasi, presisi, *recall* dan *f score*-nya. Langkah selanjutnya adalah penyimpanan model seperti pada baris 8-9.

```
1. svm_linear = svm.fit(X_train, y_train)
2.
3. y_SVM_lin = svm_linear.predict(X_test)
4. print('Akurasi', accuracy_score(y_test, y_SVM_lin))
5. target_names = ['class 0', 'class 1', 'class 2', 'cla
   ss 3']
6. print(classification_report(y_test, y_SVM_lin, target
   _names=target_names))
7.
```

```
8. with open('data/model/svm_linear_model.pkl', 'wb') as
    sl:
9.     pickle.dump(svm_linear, sl)
```

Kode Sumber 4.21 - SVM Dengan Pembagian Dataset 33:67

4.3.7.2. Klasifikasi dengan Cross Validation

Implementasi *Support Vector Machine* dengan pembagian dataset secara cross validation dengan fungsi `StratifiedKFold` bernilai `n_splits` = 10. Sehingga didapatkan *data testing* sebanyak 1/10 bagian dan *data training* sebanyak 9/10 bagian. Selanjutnya dengan perulangan senilai jumlah k, variabel `x` dan `y` di split sebanyak k bagian. Lalu dilakukan pemisahan antara *data training* dan *data testing* seperti pada baris 5-6 Kode Sumber 4.22. *Data training* kemudian dilatih pada *method* `fit` seperti pada baris ke-8 di Kode Sumber 4.22. Selanjutnya dilakukan prediksi dengan *data testing* seperti pada baris ke-9, hingga didapatkan akurasi, presisi, *recall* dan *f score*-nya. Langkah selanjutnya adalah penyimpanan model dengan nilai akurasi terbaik seperti pada baris 15-18.

```
1. skf = StratifiedKFold(n_splits=10)
2. currentMax=0
3.
4. for train_index, test_index in skf.split(x, y):
5.     X_train_cv, X_test_cv = x[train_index], x[test_in
   dex]
6.     y_train_cv, y_test_cv = y[train_index], y[test_in
   dex]
7.
8.     svm_cv = svm.fit(X_train_cv, y_train_cv)
9.     y_SVM_cv = svm_cv.predict(X_test_cv)
10.        akurasi = accuracy_score(y_test_cv, y_SVM_cv)

11.        print('Akurasi', akurasi)
12.        target_names = ['class 0', 'class 1', 'class
   2', 'class 3']
```

```
13.         print(classification_report(y_test_cv, y_SVM_
    cv, target_names=target_names))
14.
15.         if(currentMax  < akurasi):
16.             currentMax = akurasi
17.             with open('data/model/svm_cv_model.pkl',
    'wb') as svmc:
18.                 pickle.dump(svm_linear, svmc)
```

Kode Sumber 4.22 - SVM Dengan Pembagian Dataset Cross Validation

4.3.7.3. Klasifikasi dengan Over Sampling SMOTE

Implementasi *Support Vector Machine* dengan *data training* yang sudah di-*sampling* dengan metode *over sampling* SMOTE dimasukan menjadi parameter *method* `fit` untuk dilatih seperti pada baris ke-1 di Kode Sumber 4.23. Selanjutnya dilakukan prediksi dengan *data testing* seperti pada baris ke-3, hingga didapatkan akurasi, presisi, *recall* dan *f score*-nya. Langkah selanjutnya adalah penyimpanan model seperti pada baris 8-9.

```
1. svm_smt = svm.fit(ftr_smt, cls_smt)
2.
3. y_SVM_smt = svm_smt.predict(X_test)
4. print('Akurasi', accuracy_score(y_test, y_SVM_smt))
5. target_names = ['class 0', 'class 1', 'class 2', 'cla
   ss 3']
6. print(classification_report(y_test, y_SVM_smt, target
   _names=target_names))
7.
8. with open('data/model/svm_smt_model.pkl', 'wb') as ss
   :
9.     pickle.dump(svm_smt, ss)
```

Kode Sumber 4.23 - SVM Dengan Over Sampling SMOTE

4.3.7.4. Klasifikasi dengan Over Sampling ADASYN

Implementasi *Support Vector Machine* dengan *data training* yang sudah di-*sampling* dengan metode *over sampling*

ADASYN dimasukan menjadi parameter *method* `fit` untuk dilatih seperti pada baris ke-1 di Kode Sumber 4.24. Selanjutnya dilakukan prediksi dengan *data testing* seperti pada baris ke-3, hingga didapatkan akurasi, presisi, *recall* dan *f score*-nya. Langkah selanjutnya adalah penyimpanan model seperti pada baris 8-9.

```
1. svm_ada = svm.fit(ftr_ada, cls_ada)
2.
3. y_SVM_ada = svm_ada.predict(X_test)
4. print('Akurasi', accuracy_score(y_test, y_SVM_ada))
5. target_names = ['class 0', 'class 1', 'class 2',
   'class 3']
6. print(classification_report(y_test, y_SVM_ada,
   target_names=target_names))
7.
8. with open('data/model/svm_ada_model.pkl', 'wb') as
   sal:
9.     pickle.dump(svm_ada, sal)
```

Kode Sumber 4.24 - SVM Dengan Over Sampling ADASYN

4.3.8. Klasifikasi *Naïve Bayes*

Implementasi pengklasifikasian dengan *Naïve Bayes* dibantu dengan pustaka sklearn. Parameter yang diatur pada metode ini antara lain dengan menentukan `alpha` senilai 1 sesuai *default* pada sklearn. Pengaturan parameter pada fungsi `MultinomialNB` ini digunakan untuk semua percobaan pengklasifikasian Naïve Bayes dan dapat dilihat pada Kode Sumber 4.25.

```
1. from sklearn.naive_bayes import MultinomialNB
2.
3. nb = MultinomialNB(alpha=1)
4. nb
```

Kode Sumber 4.25 - Pengaturan Parameter Naive Bayes

4.3.8.1. Klasifikasi dengan Pembagian Dataset Berdasar Rasio

Implementasi *Naïve Bayes* dengan pemisahan dataset berdasar rasio sehingga didapatkan *data training* sebanyak 67% dimasukan dalam parameter *method* `fit` untuk dilatih seperti pada baris ke-1 di Kode Sumber 4.26. Selanjutnya dilakukan prediksi dengan *data testing* seperti pada baris ke-3, hingga didapatkan akurasi, presisi, *recall* dan *f score*-nya. Langkah selanjutnya adalah penyimpanan model seperti pada baris 8-9.

```
1. mnb = nb.fit(X_train, y_train)
2.
3. y_mnb = mnb.predict(X_test)
4. print('Akurasi', accuracy_score(y_test, y_mnb))
5. target_names = ['class 0', 'class 1', 'class 2', 'cla
   ss 3']
6. print(classification_report(y_test, y_mnb, target_nam
   es=target_names))
7.
8. with open('data/model/nb_model.pkl', 'wb') as nbm:
9.     pickle.dump(mnb, nbm)
```

Kode Sumber 4.26 - Naive Bayes Dengan Pembagian Dataset 33:67

4.3.8.2. Klasifikasi dengan Cross Validation

Implementasi *Naïve Bayes* dengan pembagian dataset secara cross validation dengan fungsi `StratifiedKFold` bernilai `n_splits` = 10. Sehingga didapatkan *data testing* sebanyak 1/10 bagian dan *data training* sebanyak 9/10 bagian. Selanjutnya dengan perulangan senilai jumlah k, variabel `x` dan `y` di split sebanyak k bagian. Lalu dilakukan pemisahan antara *data training* dan *data testing* seperti pada baris 5-6 Kode Sumber 4.27. *Data training* kemudian dilatih pada *method* `fit` seperti pada baris ke-8 di Kode Sumber 4.27. Selanjutnya dilakukan prediksi dengan *data testing* seperti pada baris ke-9, hingga didapatkan akurasi, presisi, *recall* dan *f score*-nya. Langkah selanjutnya adalah

penyimpanan model dengan nilai akurasi terbaik seperti pada baris 16-19.

```
1. skf = StratifiedKFold(n_splits=5)
2. currentMax=0
3. 
4. for train_index, test_index in skf.split(x, y):
5.     X_train_cv, X_test_cv = x[train_index], x[test_in
   dex]
6.     y_train_cv, y_test_cv = y[train_index], y[test_in
   dex]
7. 
8.     nb_cv = nb.fit(X_train_cv, y_train_cv)
9.     y_nb_cv = mnb.predict(X_test_cv)
10.         akurasi = accuracy_score(y_test_cv, y_nb_cv)

11. 
12.         print('Akurasi', akurasi)
13.         target_names = ['class 0', 'class 1', 'class
   2', 'class 3']
14.         print(classification_report(y_test_cv, y_nb_c
   v, target_names=target_names))
15. 
16.         if(currentMax  < akurasi):
17.             currentMax = akurasi
18.             with open('data/model/nb_cv_model.pkl', '
   wb') as nbc:
19.                 pickle.dump(mnb, nbc)
```

Kode Sumber 4.27 - Naive Bayes Dengan Pembagian Dataset Cross Validation

4.3.8.3. Klasifikasi dengan Over Sampling SMOTE

Implementasi *Naïve Bayes* dengan *data training* yang sudah di-*sampling* dengan metode *over sampling* SMOTE dimasukan menjadi parameter *method* `fit` untuk dilatih seperti pada baris ke-1 di Kode Sumber 4.28. Selanjutnya dilakukan prediksi dengan *data testing* seperti pada baris ke-3, hingga didapatkan akurasi, presisi, *recall* dan *f score*-nya. Langkah selanjutnya adalah penyimpanan model seperti pada baris 8-9.

```
1. mnb_smt = nb.fit(ftr_smt, cls_smt)
2.
3. y_mnb_smt = mnb_smt.predict(X_test)
4. print(accuracy_score(y_test, y_mnb_smt))
5. target_names = ['class 0', 'class 1', 'class 2', 'cla
   ss 3']
6. print(classification_report(y_test, y_mnb_smt, target
   _names=target_names))
7.
8. with open('data/model/nb_smt_model.pkl', 'wb') as nbs
   :
9.     pickle.dump(mnb_smt, nbs)
```

Kode Sumber 4.28 - Naive Bayes Dengan Over Sampling SMOTE

4.3.8.4. Klasifikasi dengan Over Sampling ADASYN

Implementasi *Naïve Bayes* dengan *data training* yang sudah di-*sampling* dengan metode *over sampling* ADASYN dimasukan menjadi parameter *method* `fit` untuk dilatih seperti pada baris ke-1 di Kode Sumber 4.29. Selanjutnya dilakukan prediksi dengan *data testing* seperti pada baris ke-3, hingga didapatkan akurasi, presisi, *recall* dan *f score*-nya. Langkah selanjutnya adalah penyimpanan model seperti pada baris 8-9.

```
1. mnb_ada = nb.fit(ftr_ada, cls_ada)
2.
3. y_mnb_ada = mnb_ada.predict(X_test)
4. print(accuracy_score(y_test, y_mnb_ada))
5. target_names = ['class 0', 'class 1', 'class 2', 'cla
   ss 3']
6. print(classification_report(y_test, y_mnb_ada, target
   _names=target_names))
7.
8. with open('data/model/nb_ada_model.pkl', 'wb') as nba
   :
9.     pickle.dump(mnb_ada, nba)
```

Kode Sumber 4.29 - Naive Bayes Dengan Over Sampling ADASYN

4.3.9. Klasifikasi Tanpa Machine Learning

Implementasi pengklasifikasian tanpa *Machine Learning* dibuat penulis berdasarkan pengamatan struktur dan karakter penulisan keterangan mata kuliah pada MOOC. Maka dibuat *rule* untuk metode tanpa *machine learning* seperti pada Kode Sumber 4.30.

Pertama dibuat variabel `countAcc` untuk menghitung nilai yang diterima sebagai nilai akurasi jika syarat dari *rule* tersebut terpenuhi pada suatu data.

Pada baris 8-10 menunjukan *rule* untuk Nama Mata Kuliah, maka jika memenuhi syarat Nama Mata Kuliah nilai `countAcc` di-*increment*.

Pada baris 11-13 menunjukan *rule* untuk Deskripsi Mata Kuliah, maka jika memenuhi syarat Deskripsi Mata Kuliah nilai `countAcc` di-*increment.*

Pada baris 14-16 menunjukan *rule* untuk Capaian Pembelajaran, maka jika memenuhi syarat Capaian Pembelajaran nilai `countAcc` di-*increment.*

Pada baris 17-19 menunjukan *rule* untuk Pokok Bahasan, maka jika memenuhi syarat Pokok Bahasan nilai `countAcc` di-*incremen.*

Lalu pada baris 23-25 menunjukan perhitungan akurasi dengan menjumlahkan seluruh `countAcc` yang memenuhi syarat dibagi dengan seluruh data.

```
1. from sklearn.metrics import confusion_matrix
2. countAcc = 0
3. pred = []
4. deskr_ptr=deskr_p.transpose()
5.
6. for index, data in enumerate(deskr_ptr):
```

```
7.      kata = len(deskr_ptr[index]['kalimat'].split())
8.      if (kata<=12) and (deskr_ptr[index]["naon"] >= 0.
   5) and (deskr_ptr[index]["subyek"] == 0) and (deskr_p
   tr[index]["label"]==0):
9.          countAcc += 1
10.         pred.append(0)
11.     elif (deskr_ptr[index]["subyek"] == 1) and (deskr
   _ptr[index]["propn"] == 1) and(deskr_ptr[index]["labe
   l"]==1):
12.         countAcc += 1
13.         pred.append(1)
14.     elif (deskr_ptr[index]["subyek"] == 0) and (deskr
   _ptr[index]["takso"] == 1) and (deskr_ptr[index]["lab
   el"]==2):
15.         countAcc += 1
16.         pred.append(2)
17.       elif (kata>12) and (deskr_ptr[index]["naon"] >=
    0.5) and (deskr_ptr[index]["subyek"] == 0) and (desk
   r_ptr[index]["label"]==3):
18.             countAcc += 1
19.             pred.append(3)
20.       else:
21.             pred.append(4)
22.
23.     summ = np.sum(confusion_matrix(deskr_p['label'],
   pred))
24.     a = (countAcc / summ) * 100
25.     print ('akurasi: ', a)
```

Kode Sumber 4.30 - Klasifikasi tanpa Machine Learning

BAB V
UJI COBA DAN EVALUASI

Pada bab ini, akan dijabarkan hasil uji coba beserta evaluasi dari sistem yang telah dibuat. Pengujian dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

5.1. Lingkungan Pengujian

Lingkungan pengujian pada pengerjaan tugas akhir ini dilakukan pada perangkat keras dan sistem operasi sebagai beriktu:

- Intel Core i7-7500U @ 2.7GHz (4 CPUs)
- NVIDIA GeForce 930MX 4 GB
- 8 GB of RAM
- Windows 10 Home 64-bit

5.2. Data Uji Coba

Data yang digunakan pada tugas akhir ini adalah judul dan keterangan mata kuliah pada situs MOOC. Dataset yang diambil berjumlah 1136 data. Pada pengujian tugas akhir ini, pembagian data dibagi menjadi beberapa metode. Pertama dengan pemisahan *data training* 67% dan *data testing* 33% sehingga didapatkan 761 d*ata training* dan 375 *data testing*. Kedua dengan pemisahan *data testing* dan *data training* dengan *cross validation* dengan pembagian 10 *fold* sehingga didapatkan 908 *data training* dan 227 *data testin*g pada masing-masing fold.

Selanjutnya untuk prediksi klasifikasi, *data training* hasil pemisahan dataset yang dipisah berdasar rasio diuji dengan metode penyeimbangan data yaitu metode *over sampling* menggunakan algortitma SMOTE dan ADASYN. Pembagian dataset pada masing-masing kelas dijabarkan pada Tabel 5.1.

Tabel 5.1 - Dataset

Kelas	Rasio				Cross Validation	
	Data Testing	Data Training			Data Testing	Data Training
		Tanpa Oversampling	SMOTE	ADASYN		
0	124	252	252	252	75	300
1	120	251	252	251	75	300
2	124	245	252	247	73	292
3	7	13	252	245	4	16

Selanjutnya terdapat contoh data masukan baru untuk dijadikan data uji coba. Lalu dilihat hasil keluaran kelas yang dihasilkan dari masing-masing model klasifikasi yang sudah dilatih. Contoh data baru yang akan diuji terdapat pada Tabel 5.2.

Tabel 5.2 - Data uji coba

Data Uji	Data	Label Seharusnya
Data Uji 1	Data Mining	Judul Mata Kuliah
Data Uji 2	Mata kuliah ini tentang pengantar data mining yang merupakan suatu metode yang sangat menarik perhatian industri informasi saat ini adalah karena tersedianya data dalam jumlah yang besar dan semakin besarnya kebutuhan untuk mengubah data tersebut menjadi informasi dan pengetahuan yang berguna.Data mining adalah metode untuk mengekstraksi dan menemukan pengetahuan dari data yang berukuran/berjumlah besar,	Deskripsi Mata Kuliah

	pengetahuan inilah yang sangat berguna untuk pengambilan kebijakan di industri tersebut.Beberapa metode yang disajikan disini adalah, association rules, clustering, klasifikasi untuk menemukan pengetahuan dari data.	
Data Uji 3	Menguasai minimal dua perangkat lunakstatistika, termasuk perangkat lunak yang berbasis open source. Mampu melakukan perancangan percobaan,pengumpulan dan pembangkitan data (dalam bentuk survei, percobaan, atausimulasi), pengorganisasian data, analisis data menggunakan teknik-teknikstatistika, dan penarikan kesimpulan secara sahih dengan memanfaatkan minimalsatu perangkat lunak statistika. Mampu mendokumentasikan, menyimpan, mengamankan danmenemukan kembali data untuk menjamin keshahihan dan mencegah plagiasi	Capaian Pembelajaran
Data Uji 4	Materi kuliah ini adalah: Pengantar Data Besar, Statistics Descriptive, Association Rules, Clustering, Classification.	Pokok Bahasan

5.3. Skenario Uji Coba

Subbab ini akan menjelaskan skenario uji yang telah dilakukan. Terdapat 3 skenario untuk dilakukan uji coba dengan metode klasifikasi dengan melakukan perbandingan terhadap

persebaran *data training* yang telah dijabarkan pada Table 5.1 dan terdapat satu skenario untuk menguji dengan metode tanpa *machine learning*. Masing-masing model selanjutnya diuji dengan data baru untuk melihat kelas yang dihasilkan. Berikut merupakan penjelasan setiap skenario pengujian:

1. Skenario Pengujian 1: Dalam skenario ini akan dilakukan pengujian berupa nilai akurasi, presisi, *recall*, dan *f-measure* dengan menggunakan metode klasifikasi *Random Forest* menggunakan prediksi pada *data training* tanpa *over sampling*, *data training* dengan *over sampling* SMOTE, *data training* dengan *over sampling* ADASYN dan *data training* dengan persebaran menggunan *cross validation*. Sedangkan skor presisi, *recall*, dan *f-measure* yang ditampilkan pada laporan skenario yaitu presisi, *recall*, dan *f-measure* dengan *average weighted*.
2. Skenario Pengujian 2: Dalam skenario ini akan dilakukan pengujian berupa nilai akurasi, presisi, *recall*, dan *f-measure* dengan menggunakan metode klasifikasi *Support Vector Machine* menggunakan prediksi pada *data training* tanpa *over sampling*, *data training* dengan *over sampling* SMOTE, *data training* dengan *over sampling* ADASYN dan *data training* dengan persebaran menggunan *cross validation*. Sedangkan skor presisi, *recall*, dan *f-measure* yang ditampilkan pada laporan skenario yaitu presisi, *recall*, dan *f-measure* dengan *average weighted*.
3. Skenario Pengujian 3: Dalam skenario ini akan dilakukan pengujian berupa nilai akurasi, presisi, *recall*, dan *f-measure* dengan menggunakan metode klasifikasi *Naïve Bayes* menggunakan prediksi pada *data training* tanpa *over sampling*, *data training* dengan *over sampling* SMOTE, *data training* dengan *over sampling* ADASYN

dan *data training* dengan persebaran menggunan *cross validation*. Sedangkan skor presisi, *recall*, dan *f-measure* yang ditampilkan pada laporan skenario yaitu presisi, *recall*, dan *f-measure* dengan *average weighted*.

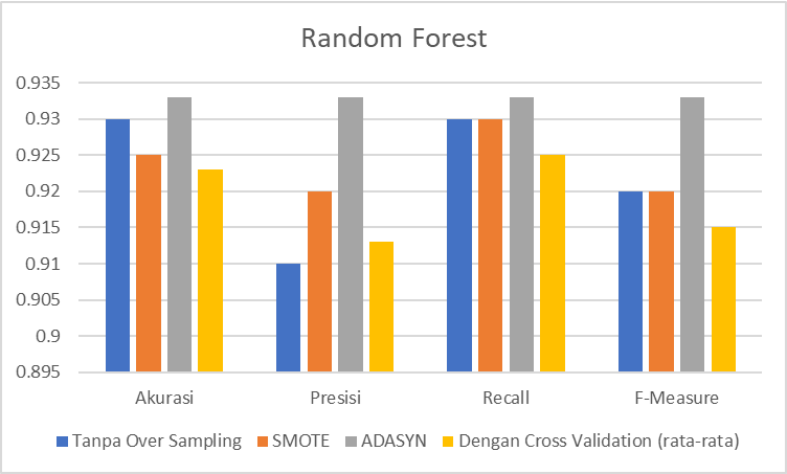
4. Skenario Pengujian 4: Dalam skenarion akan dilakukan pengujian berupa nilai akurasi dari pencocokan antara rule yang telah penulis buat dengan masukan data.

5.3.1. Skenario Pengujian 1

Pada skenario ini dilakukan uji klasifikasi menggunakan metode *Random Forest*. Percobaan pengujian dilakukan dengan memprediksi model menggunakan *data training* tanpa *over sampling*, *data training* dengan *over sampling* SMOTE, *data training* dengan *over sampling* ADASYN dan *data training* dengan persebaran menggunakan *cross validation*. Hasil pengujian terdapat pada Tabel 5.3.

Tabel 5.3 – Skenario Pengujian 1

Data Training	**Random Forest**			
	Akurasi	**Presisi**	**Recall**	**F-Measure**
Tanpa Over Sampling	0.930	0.91	0.930	0.920
SMOTE	0.925	0.92	0.93	0.92
ADASYN	0.933	0.933	0.933	0.933
Dengan Cross Validation (rata-rata)	0.923	0.913	0.925	0.915



Gambar 5.1 - Grafik evaluasi Skenario 1

Berdasarkan hasil pengujian, model dengan akurasi terbaik dengan skor 0.933 adalah klasifikasi *Random Forest* dengan persebaran data yang telah di-*over sampling* menggunakan metode ADASYN. Sedangkan model dengan akurasi terendah dengan skor 0.923 yaitu klasifikasi *Random Forest* dengan persebaran data menggunakan metode *Cross Validation*, nilai standar deviasi akurasi dari *cross validation* adalah 0.06245.

Model hasil *training* selanjutnya diuji dengan data baru, dan dapat dilihat keluaran kelas yang dihasilkan. Hasil pengujian dengan data baru terdapat pada Tabel 5.4.

Tabel 5.4 - Keluaran hasil model Skenario 1

Data Training	Random Forest			
	Data Uji 1	**Data Uji 2**	**Data Uji 3**	**Data Uji 4**
Tanpa Over Sampling	Nama Mata Kuliah	Deskripsi Mata Kuliah	Capaian Pembelajaran	Deskripsi Mata Kuliah
SMOTE	Nama Mata Kuliah	Deskripsi Mata Kuliah	Capaian Pembelajaran	Pokok Bahasan
ADASYN	Nama Mata Kuliah	Deskripsi Mata Kuliah	Capaian Pembelajaran	Pokok Bahasan
Dengan Cross Validation (rata-rata)	Nama Mata Kuliah	Deskripsi Mata Kuliah	Capaian Pembelajaran	Capaian Pembelajaran

Dari hasil pengujian metode *Random Forest* pada persebaran *data trainin*g yang di-*over sampling* menggunakan metode SMOTE dan ADASYN, dihasilkan keluaran label yang sesuai terhadap keempat data uji baru. Sedangkan persebaran *data training* yang tidak di-*over sampling* menghasilkan keluaran label pada Data Uji 4 yang keliru, dimana keluaran label seharusnya Pokok Bahasan menjadi Deskripsi Mata Kuliah. Begitu juga pada metode yang menggunakan persebaran *data training* dengan metode *cross validation*, keluaran label pada Data Uji 4 menghasilkan label yang keliru, dimana keluaran label seharusnya Pokok Bahasan menjadi Capaian Pembelajaran.

5.3.2. Skenario Pengujian 2

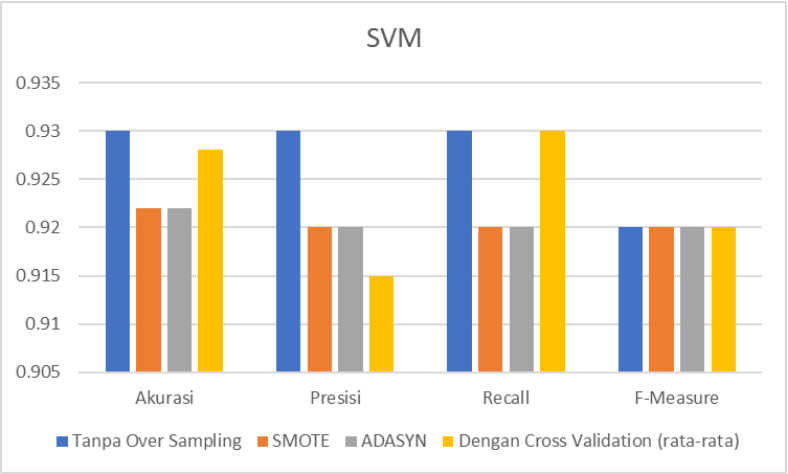
Pada skenario ini dilakukan uji klasifikasi menggunakan metode *Support Vector Machine*. Percobaan pengujian dilakukan

dengan memprediksi model menggunakan *data training* tanpa *over sampling*, *data training* dengan *over sampling* SMOTE, *data training* dengan *over sampling* ADASYN dan *data training* dengan persebaran menggunan *cross validation*. Hasil pengujian terdapat pada Tabel 5.5.

Tabel 5.5 – Skenario Pengujian 2

Data Training	**SVM**			
	Akurasi	**Presisi**	**Recall**	**F-Measure**
Tanpa Over Sampling	0.93	0.93	0.93	0.92
SMOTE	0.922	0.92	0.92	0.92
ADASYN	0.922	0.92	0.92	0.92
Dengan Cross Validation (rata-rata)	0.928083	0.915	0.93	0.92

Berdasarkan hasil pengujian, model dengan akurasi terbaik dengan skor 0.93 adalah klasifikasi *SVM* dengan persebaran data berdasar rasio. Sedangkan model dengan akurasi terendah dengan skor 0.922 yaitu klasifikasi *SVM* dengan persebaran data yang sudah di-*over sampling* dengan metode SMOTE dan ADASYN.



Gambar 5.2 - Grafik evaluasi Skenario 2

Model hasil training selanjutnya diuji dengan data baru, dan dapat dilihat keluaran kelas yang dihasilkan. Hasil pengujian dengan data baru terdapat pada Tabel 5.6.

Dari hasil pengujian data baru menggunakan model yang dilatih dari keempat jenis *data training* dengan metode klasifikasi SVM, dihasilkan keluaran label yang sesuai terhadap Data Uji 1, 2, dan 3. Sedangkan keluaran label pada Data Uji 4 mengalami kekeliruan, dimana label yang dihasilkan seharusnya Pokok Bahasan.

Tabel 5.6 - Keluaran hasil model Skenario 2

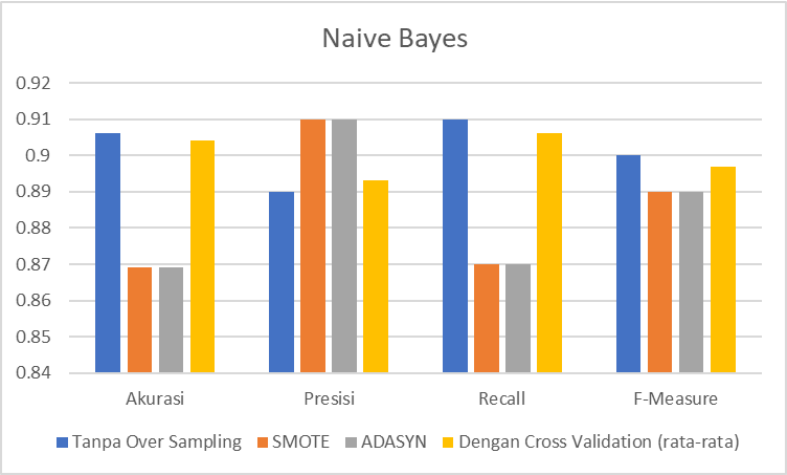
Data Training	SVM			
	Data Uji 1	Data Uji 2	Data Uji 3	Data Uji 4
Tanpa Over Sampling	Nama Mata Kuliah	Deskripsi Mata Kuliah	Capaian Pembelajaran	Nama Mata Kuliah
SMOTE	Nama Mata Kuliah	Deskripsi Mata Kuliah	Capaian Pembelajaran	Nama Mata Kuliah
ADASYN	Nama Mata Kuliah	Deskripsi Mata Kuliah	Capaian Pembelajaran	Nama Mata Kuliah
Dengan Cross Validation (rata-rata)	Nama Mata Kuliah	Deskripsi Mata Kuliah	Capaian Pembelajaran	Capaian Pembelajaran

5.3.3. Skenario Pengujian 3

Pada skenario ini dilakukan uji klasifikasi menggunakan metode *Naïve Bayes*. Percobaan pengujian dilakukan dengan memprediksi model menggunakan *data training* tanpa *over sampling*, *data training* dengan *over sampling* SMOTE, *data training* dengan *over sampling* ADASYN dan *data training* dengan persebaran menggunan *cross validation*. Hasil pengujian terdapat pada Tabel 5.7.

Tabel 5.7 - Skenario Pengujian 3

Data Training	Naïve Bayes			
	Akurasi	Presisi	Recall	F-Measure
Tanpa Over Sampling	0.906	0.89	0.91	0.90
SMOTE	0.869	0.91	0.87	0.89
ADASYN	0.869	0.91	0.87	0.89
Dengan Cross Validation (rata-rata)	0.904	0.893	0.906	0.897



Gambar 5.3 - Grafik Evaluasi Skenario 3

Berdasarkan hasil pengujian, model dengan akurasi terbaik dengan skor 0.906 adalah klasifikasi *Naïve Bayes* dengan persebaran data berdasar rasio. Sedangkan model dengan akurasi terendah dengan skor 0.869 yaitu klasifikasi *Naïve Bayes* dengan

persebaran data yang sudah di-*over sampling* dengan metode SMOTE dan ADASYN.

Model hasil training selanjutnya diuji dengan data baru, dan dapat dilihat keluaran kelas yang dihasilkan. Hasil pengujian dengan data baru terdapat pada Tabel 5.8.

Tabel 5.8 - Keluaran hasil model Skenario 3

Data Training	**Naïve Bayes**			
	Data Uji 1	**Data Uji 2**	**Data Uji 3**	**Data Uji 4**
Tanpa Over Sampling	Nama Mata Kuliah	Deskripsi Mata Kuliah	Capaian Pembelajaran	Nama Mata Kuliah
SMOTE	Nama Mata Kuliah	Deskripsi Mata Kuliah	Capaian Pembelajaran	Nama Mata Kuliah
ADASYN	Nama Mata Kuliah	Deskripsi Mata Kuliah	Capaian Pembelajaran	Nama Mata Kuliah
Dengan Cross Validation (rata-rata)	Nama Mata Kuliah	Deskripsi Mata Kuliah	Capaian Pembelajaran	Nama Mata Kuliah

Dari hasil pengujian data baru menggunakan model yang dilatih dari keempat jenis *data training* dengan metode klasifikasi *Naïve Bayes*, dihasilkan keluaran label yang sesuai terhadap Data Uji 1, 2, dan 3. Sedangkan keluaran label pada Data Uji 4 mengalami kekeliruan, dimana label yang dihasilkan seharusnya Pokok Bahasan.

5.3.4. Skenario Pengujian 4

Pada skenario ini dilakukan perhitungan akurasi menggunakan *rule* yang penulis buat. Hasil akurasi yang didapatkan adalah 71,74%. Selanjutnya diuji dengan data uji baru. Hasil pengujian dengan data uji dapat dilihat pada Tabel 5.9.

Dari hasil pengujian data uji, dihasilkan keluaran label yang sesuai terhadap Data Uji 1, 2, dan 3. Sedangkan keluaran label pada Data Uji 4 mengalami kekeliruan, dimana label yang dihasilkan seharusnya Pokok Bahasan.

Tabel 5.9 - Keluaran hasil model Skenario 4

Data Training	**Tanpa Machine Learning**			
	Data Uji 1	**Data Uji 2**	**Data Uji 3**	**Data Uji 4**
Hasil	Nama Mata Kuliah	Deskripsi Mata Kuliah	Capaian Pembelajaran	Deskripsi Mata Kuliah

5.4. Evaluasi

Pengujian skenario 1 diperoleh hasil terbaik menggunakan klasifikasi *Random Forest* dengan evaluasi akurasi, presisi, *recall* dan *f-measure* pada *data training* yang telah diseimbangkan menggunakan metode *over sampling* ADASYN. Skor akurasi yang diperoleh adalah 93,3%. Pengujian menggunakan data baru dari model yang dilatih menggunakan *data training* dari metode *over sampling* ADASYN juga menghasilkan label yang sesuai dari keempat data uji.

Pengujian skenario 2 diperoleh hasil terbaik menggunakan klasifikasi *Support Vector Machine* dengan evaluasi akurasi, presisi, *recall* dan *f-measure* pada *data training* yang dibagi dengan rasio 67% tanpa metode *over sampling.* . Skor akurasi yang

diperoleh adalah 93%. Pengujian menggunakan data baru dari model yang dilatih menggunakan *data training* yang dibagi dengan rasio 67% tanpa metode *over sampling* juga menghasilkan label yang sesuai pada Data Uji 1, Data Uji 2, dan Data Uji 3. Data Uji 4 menghasilkan label yang keliru, hal ini dikarenakan data latih untuk label Pokok Bahasan memiliki representasi yang lebih sedikit dan jumlah data kelas tidak seimbang.

Pengujian skenario 3 menggunakan klasifikasi *Naïve Bayes* diperoleh hasil terbaik untuk akurasi, presisi, *recall* dan *f-measure* pada *data training* yang dibagi dengan rasio 67% tanpa metode *over sampling* dengan nilai akurasi 90,6%. Pengujian menggunakan data baru dari model yang dilatih menggunakan *data training* yang dibagi dengan rasio 67% tanpa metode *over sampling* juga menghasilkan label yang sesuai pada Data Uji 1, Data Uji 2, dan Data Uji 3. Data Uji 4 menghasilkan label yang keliru, hal ini dikarenakan data latih untuk label Pokok Bahasan memiliki representasi yang lebih sedikit dan jumlah data kelas tidak seimbang.

Pengujian skenario 4 menggunakan klasifikasi tanpa *machine learning* menghasilkan akurasi 71,74%. Akurasi yang didapatkan merupakan skor dari *rule* yang telah ditentukan oleh penulis. Pengujian menggunakan data baru dari *rule* yang telah dibuat menghasilkan label yang sesuai pada Data Uji 1, Data Uji 2, dan Data Uji 3. Data Uji 4 menghasilkan label yang keliru, hal ini dikarenakan *rule* yang dibaca oleh sistem merupakan program dengan menggunakan *condition statement* (*if* dan *elif* pada *python*). Jadi ketika mendapati syarat dari label Deskripsi Mata Kuliah telah terpenuhi untuk Data Uji 4, maka label Deskripsi Mata Kuliah dianggap sebagai keluaran yang benar oleh sistem.

Dari keempat skenario, dapat disimpulkan model dengan hasil terbaik adalah model yang menggunakan metode klasifikasi

Random Forest. Selain menghasilkan nilai akurasi terbaik, keluaran label pada pengujian data baru menghasilkan label yang sesuai pada seluruh data uji. Hal ini dikarenakan pengaturan parameter pada model dinilai sudah tepat dan pemilihan metode *over sampling* dinilai baik untuk persebaran data yang tidak seimbang.

[Halaman ini sengaja diksongkan]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperoleh selama pengerjaan tugas akhir dan saran mengenai pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

6.1 Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Cara menentukan anotasi detail konten kursus pada MOOC tugas akhir ini dilakukan dengan melakukan pengklasifikasian tanpa *Machine Learning* dan dengan *Machine Learning* yaitu *text processing* mulai dari *pre processing* hingga pengklasifikasian kelas. Dari ketiga skenario dengan *Machine Learning*, didapatkan model dengan akurasi terbaik yaitu dengan metode klasifikasi *Random Forest*. Selain mendapatkan akurasi terbaik, yaitu 93,3%, pengujian dengan data baru terbukti menghasilkan *output* label yang sesuai. *Random Forest* mendapatkan hasil terbaik karena model yang dibangun dinilai tepat dengan menentukan jumlah pohon keputusan sebanyak 100. Sedangkan pengklasifikasian dengan hasil terendah didapatkan dari metode tanpa *Machine Learning* dengan nilai akurasi 71,7%. Hal ini dikarenakan *rule* pada pengklasifikasian tanpa metode *Machine Learning* yang dibaca oleh sistem merupakan program dengan menggunakan *condition statement* (*if* dan *elif* pada *python*). Jadi jika mendapati syarat dari Label ID awal telah terpenuhi, *condition statement* selanjutnya tidak dilihat lagi yang bisa mengakibatkan kekeliruan dalam keluaran hasilnya.
2. Cara mengekstraksi setiap anotasi pada kursus menjadi sebuah metadata yang sesuai standar SCORM dan Moodle

diambil mengikuti beberapa data yang ada pada standard metadata SCORM berupa judul mata kuliah sesuai *tag* `<title>`, deskripsi mata kuliah sesuai *tag* `<description>`, capaian pembelajaran sesuai *tag* `<coverage>` dan pokok bahasan sesuai *tag* `<keyword>.` Pada saat proses pengambilan konten kursus dari website MOOC EdX. langkah yang dilakukan adalah proses *scraping* dengan mem-*parsing* HTML web EdX. Pada metadata HTML, *Tag* `<title>` diekstrak sebagai judul mata kuliah, *class* `“description”` diekstrak sebagai deskripsi mata kuliah dan *class* *“*`educationalOutcome`*”* diekstrak sebagai capaian pembelajaran. Sehingga dihasilkan metadata berformat JSON dengan data yang dihasilkan yaitu Judul Mata Kuliah, Deskripsi Mata Kuliah, dan Capaian Pembelajaran.

6.2 Saran

Berikut ini adalah saran untuk pengembangan penelitian selanjutnya. Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi, dan pengujian yang telah dilakukan.

1. Penggunaan metode *over sampling* ketika *data training* yang didapatkan tidak seimbang. Karena terbukti, dari keempat jenis persebaran *data training*, pada *confusion matrix* model yang dilatih menggunakan *data training* yang di-*over sampling* didapatkan *True Positive* untuk kelas minoritas (kelas Pokok Bahasan). Keterangan *confusion matrix* dapat dilihat pada Lampiran 2.
2. Model terbaik yang didapatkan dapat diimplementasikan menjadi plugin atau API pada berbagai MOOC. Agar ketika pengajar menambahkan keterangan, sistem secara otomatis akan memberi label pada keterangan menjadi anotasi yang sesuai dan sudah terlatih.

3. Pemilihan fitur lebih divariasikan, seperti melengkapi *rule* dan pembobotan kata. Pembobotan kata dapat dicoba menggunakan metode lain seperti GloVe atau Word2Vec.
4. Memperluas batasan kategori mata kuliah, tidak hanya yang memiliki topik tentang Ilmu Komputer.
5. Mengekstaksi MOOC lain sebagai dataset untuk memperkaya karakteristik MOOC yang digunakan pada model.
6. Memperluas batasan konten pada MOOC, bukan hanya membuat model untuk mengekstraksi konten kursus saja, tetapi juga konten forum kelas, pengajar, peserta didik, dan lain sebagainya.
7. Membuat model yang mampu mengklasifikasikan satu paragraf atau data yang terdiri dari beberapa label.

[Halaman ini sengaja dikosongkan]

Daftar Pustaka

[1] T. R. Liyanagunawardena, "Massive Open Online Courses," *Humanities,* pp. 35-41, 2015.

[2] Universitas Negeri Jakarta, "KampusUNJ," Universitas Negeri Jakarta, 5 November 2016. [Online]. Available: https://kampusunj.com/situs-kuliah-online/. [Accessed 16 May 2018].

[3] C. Romero, V. Sebastian and E. Garcia, "Data mining in course management systmes: Moodle case study and tutorial," pp. 368-384, 2007.

[4] H. He, Y. Bai, E. A. Garcia and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, Hong Kong, China, 2008.

[5] A. Adams, S. A. Williams and T. R. Liyanagunawardena, "MOOCs: a Systematic Study of the Published Literature 2008-2012," *International Review of Research in Open and Distance Learning · January 2013*, 2013.

[6] Rustici Software, "SCORM Explained," Rustici Software, [Online]. Available: https://scorm.com/scorm-explained/ . [Accessed 18 May 2018].

[7] RELOAD, "Reusable e-Learning Object Authoring & Delivery," RELOAD, [Online]. Available: http://tecfaetu.unige.ch/staf/staf-

k/bozelle/staf18/help/topic5/subtopic5.html. [Accessed 27 May 2018].

[8] SCORM, "SCORM 2004 Metadata Structure," 2004. [Online]. Available: https://scorm.com/scorm-explained/technical-scorm/content-packaging/metadata-structure/. [Accessed 17 January 2019].

[9] W. Madya and R. Utari, "Taksonomi Bloom, Apa dan Bagaimana Menggunakannya?," 2011.

[10] R. Z and B. O, "Analysis of methods and means of text mining," *ECONTECHMOD. AN INTERNATIONAL QUARTERLY JOURNAL*, vol. 6, pp. 73-78, 2017.

[11] S. Salomon and S. Hansum, Spam Filter Situs Jejaring Sosial Mahasiswa Menggunakan Regular Expression, Tangerang: Universitas Multimedia Nusantara, 2017.

[12] A. A. Kardan and M. B. Imani, "Improving Persian POS Tagging Using the Maximum Entropy Model".

[13] W. Informatikalogi, "https://informatikalogi.com/term-weighting-tf-idf/," 11 07 2017. [Online]. [Accessed November 2018].

[14] B. V. Thampi, C. Lukashin and T. Wong, "Novel application of Random Forest method in CERES scene type classification," *Science System Applications Inc., Hampton, VA*, 2003.

[15] A. S. Nugroho, B. A. Witarto and D. Handoko, "IlmuKomputer.com," [Online]. Available:

http://asnugroho.net/papers/ikcsvm.pdf. [Accessed 29 November 2018].

[16] E. Suwandi, Prediksi Indeks Harga Saham Gabungan (Ihsg) Dengan Pendekatan Least Squares Support Vector Machines, Bina Nusantara, 2014.

[17] S. L. B. Ginti and R. P. Trinanda, "Teknik Data Mining Menggunakan Metode Bayes Classifier Untuk Optimalisasi Pencarian Pada Aplikasi Perpustakaan (Studi Kasus : Perpustakaan Universitas Pasundan – Bandung)," *Jurnal Teknologi dan Informasi UNIKOM,* vol. Vol 1 No 6, 2014.

[18] R. Malhotra and S. Kamal, "Tool to Handle Imbalancing Problem in Software Defect Prediction Using Oversampling Methods," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Udupi, 2017.

[19] Y. E. Kurniawati, A. E. Permanasari and S. Fauziyati, "Adaptive Synthetic–Nominal (ADASYN–N) and Adaptive Synthetic–KNN (ADASYN KNN) for Multiclass Imbalance Learning on Laboratory Test Data," in *4th International Conference on Science and Technology (ICST), Yogyakarta, Indonesia*, Yogyakarta, 2018.

[20] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal Of Artificial Intelligence Research, Volume 16, pages 321-357, 2002*, 2001.

[21] H. He, Y. Bai, E. A. Garcia and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," *2008 IEEE International Joint Conference on Neural Networks*

(IEEE World Congress on Computational Intelligence), 2008.

[22] J. Brownlee, "A Gentle Introduction to k-fold Cross-Validation," Machine Learning Mastery, [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/. [Accessed 29 November 2018].

[23] P. Gupta, "Cross-Validation in Machine Learning," Medium - Towards Data Sscience, [Online]. Available: https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f. [Accessed 30 November 2018].

LAMPIRAN

1. Data Yang Digunakan

a. Struktur Metadata LOM SCORM

Berikut ini merupakan struktur metadata SCORM dengan *tag root* LOM, dan *sub-root* yang ditampilkan merupakan *tag* `<general>` saja.

Tag	Keterangan
<general>	Element metadata secara umum
<identifier>	Element untuk nilai yang mengidentifikasi objek pembelajaran
<catalog>	Skema katalog (atau ruang lingkup) di mana objek pembelajaran sedang diidentifikasi (URI, ISBN, ISSN, dll).
<entry>	Pengidentifikasi objek pembelajaran yang unik dalam katalog.
<title>	Judul kursus
<language>	Bahasa yang digunakan objek pembelajaran ini untuk berkomunikasi dengan pelajar.
<description>	Deskripsi tekstual dari objek pembelajaran.

<keyword>	Satu atau lebih kata kunci yang menggambarkan objek pembelajaran. Setiap entri kata kunci dapat berisi lebih dari satu kata jika entri tersebut mewakili frasa.
<coverage>	Menjelaskan ruang lingkup (waktu, budaya, geografi, wilayah, dll) pelatihan dalam objek pembelajaran.
<structure>	Menjelaskan struktur organisasi konten. (atomik, koleksi, jaringan, hierarkis, linier)
<aggregationLevel>	Granularitas fungsional objek pembelajaran (1 = tingkat granularitas terkecil [aset mentah], 4 = tingkat granularitas tertinggi [kursus lengkap])

b. Taksonomi Bloom

Pengetahuan	Pemahaman	Aplikasi
mengutip	memperkirakan	memerlukan
menyebutkan	menjelaskan	menyesuaikan
menggambar	mencirikan	mengalokasikan
membilang	merinci	mengurutkan
mengidentifikasi	mengasosiasikan	menerapkan
mendatfar	membandingkan	menugaskan
menunjukkan	menghitung	memperoleh
memberi label	mengkontraskan	mencegah
memberi indeks	mengubah	mencanangkan
memasangkan	mempertahankan	mengkalkulasi
menamai	menguraikan	menangkap
menandai	menjalin	memodifikasi
membaca	membedakan	mengklasifikasikan
menyadari	mendiskusikan	melengkapi
menghafal	mencontohkan	membangun
meniru	menerangkan	membiasakan
mencatat	mengemukakkan	mendemonstrasikan
mengulang	mempolakan	menurunkan
mereproduksi	memperluas	menentukan
meninjau	Menyimpulkan	menemukan
memilih	Meramalkan	menggambarkan
menyatakan	Merangkum	menemukan kembali
mempelajari	Menjabarkan	menggunakan
mentabulasi		melatih
menelusuri		menggali
menulis		membuka
		mengemukakan

Pengetahuan	Pemahaman	Aplikasi
		membuat faktor
		membuat gambar
		membuat grafik
		mengadaptasi
		menyelidiki
		memanipulasi
		mempercantik
		mengoperasikan
		mempersoalkan

Analisis	Evaluasi	Kreasi
menganalisis	mempertimbangkan	mengabstraksi
mengaudit	menilai	menganimasi
memeriksa	menyimpulkan	mengatur
membuat blueprint	mengarahkan	mengumpulkan
membuat garis besar	mengkritik	mendanai
memecahkan	menimbang	mengkategorikan
mengkarakteristikkan	memutuskan	mengkode
membuat dasar pengelompokkan	memisahkan	mengkombinasikan
merasionalkan	memprediksi	menyusun
menegaskan	memperjelas	mengarang
membuat dasar pengkontras	merangking	menanggulangi
mengkorelasikan	menafsirkan	menghubungkan
mendeteksi	memberi pertimbangan	menciptakan
mendiagnosis	membenarkan	mengkreasikan
mendiagramkan	mengukur	mengkoreksi
mendiversifikasi	memproyeksi	memotret
menyeleksi	memerinci	merancang

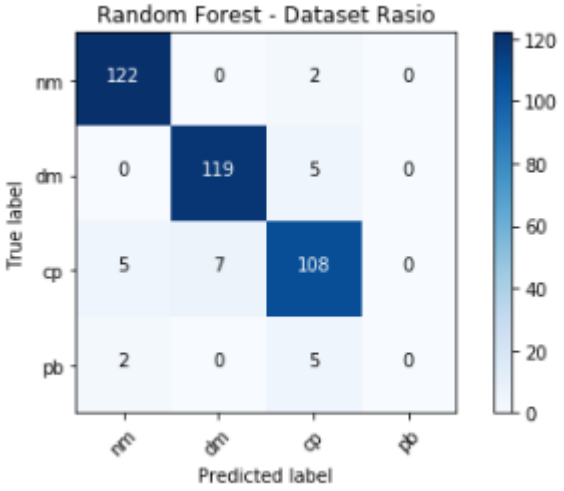
Analisis	Evaluasi	Kreasi
memerinci ke bagian-bagian	menggradasi	mengembangkan
menominasikan	merentangkan	merencanakan
mendokumentasikan	merekomendasikan	mendikte
menjamin	melepaskan	meningkatkan
menguji	merangkum	memfasilitasi
mencerahkan	mendukung	membentuk
menjelajah	mengetes	merumuskan
membagankan	memvalidasi	menggeneralisasikan
membuat kelompok	membuktikan kembali	menumbuhkan
mengilustrasikan		menangani
menginterupsi		mengirim
menelaah		memperbaiki
menata		menggabungkan
mengelola		memadukan
memaksimalkan		membatasi
meminimalkan		mengajar
mengoptimalkan		membuat model
memerintahkan		mengimprovisasi
menggarisbesarkan		membuat jaringan
memberi tanda		mengorganisasikan
memberi kode		mensketsa
memprioritaskan		mereparasi
mengedit		

c. Subjek

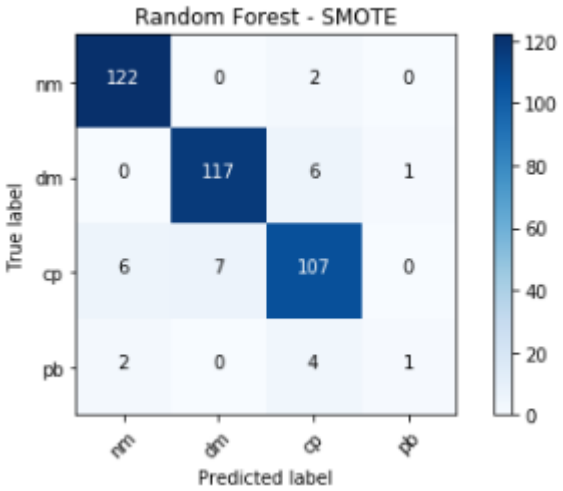
Subyek
Mahasiswa
Siswa
Pelajar
Peserta didik
Anda
Kami
Kita
Kursus
Kuliah
perkuliahan

2. Confusion Matrix

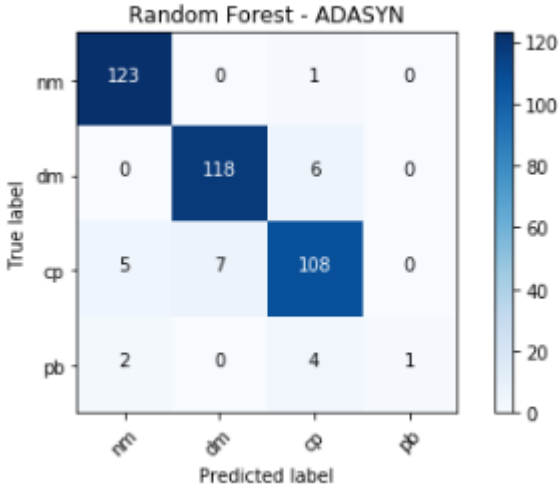
a. Random Forest



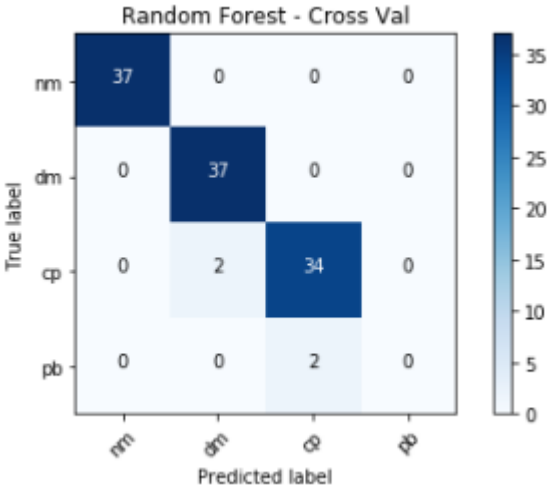
b. Random Forest – SMOTE



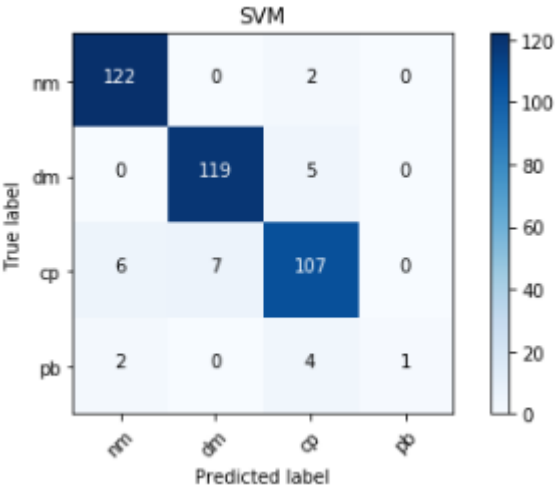
c. Random Forest – ADASYN



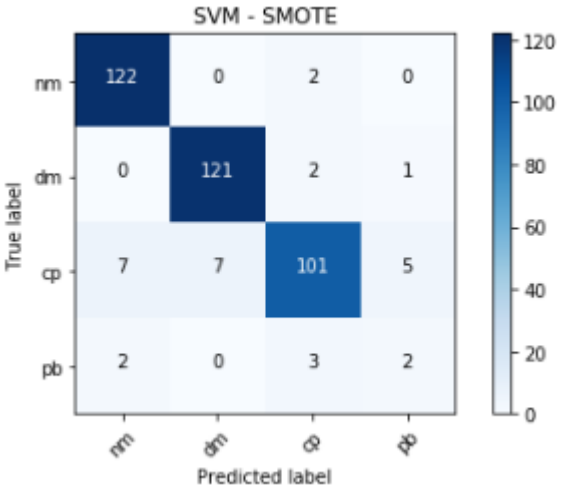
d. Random Forest – Cross Validation



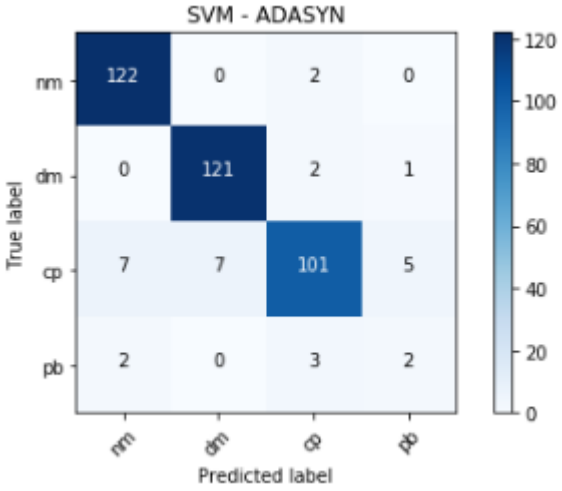
e. SVM



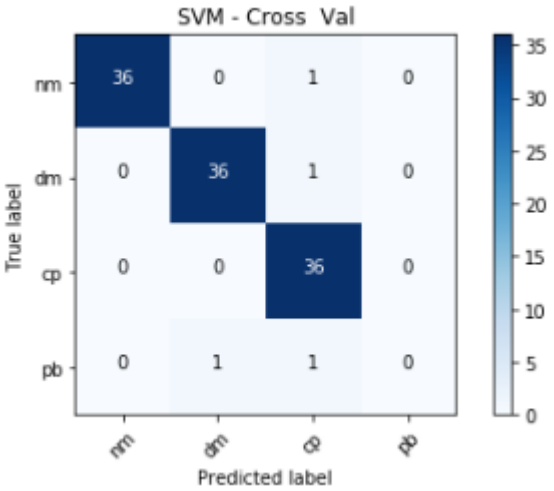
f. SVM – SMOTE



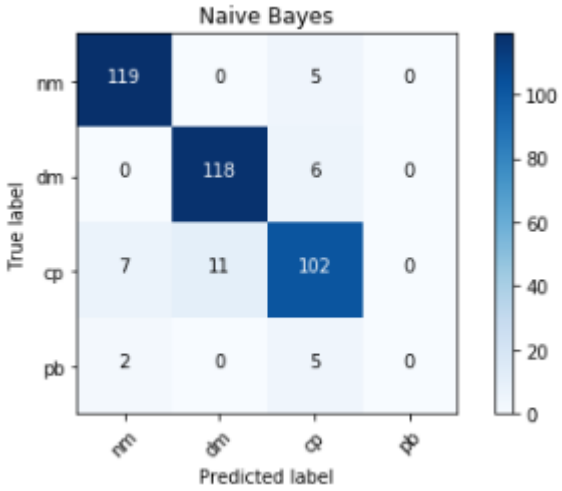
g. SVM – ADASYN



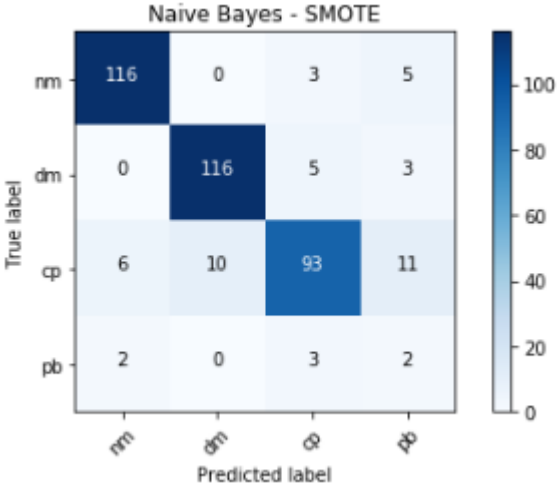
h. SVM – Cross Validation



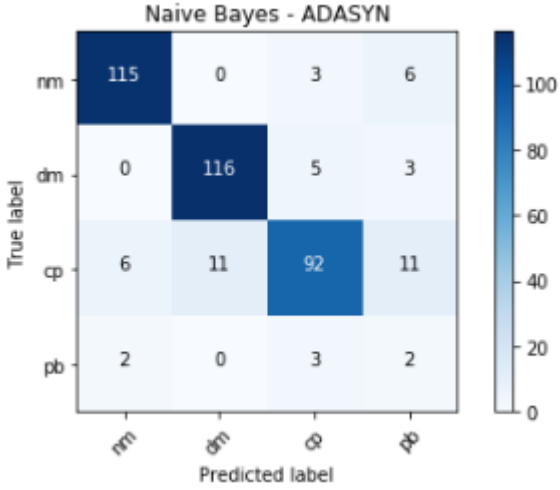
i. Naïve Bayes



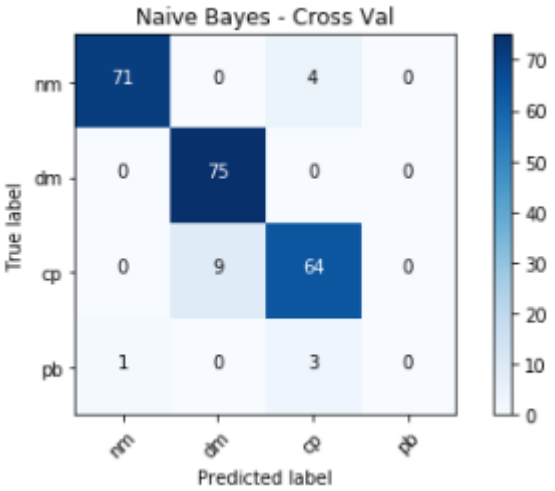
j. Naïve Bayes – SMOTE



k. Naïve Bayes – ADASYN



l. Naïve Bayes – Cross Validation



BIODATA PENULIS

Penulis, bernama lengkap Purina Qurota Ayunin, lahir di Bogor, 12 Oktober 1996. Penulis merupakan mahasiswa Teknik Informatika ITS 2015. Penulis aktif dalam berbagai keorganisasian mahasiswa, diantaranya menjadi staff Departemen Pengembangan Profesi Himpunan Mahasiswa Teknik ComputerInformatika dan staff Badan Pelayanan Umat Jamaah Masjid Manarul Ilmi. Selain itu juga Mahasiswa aktif dalam berbagai kepanitiaan acara HMTC maupun JMMI, big event jurusan Informatika yaitu Schematics, big event ITS yaitu Gerigi, dan lain sebagainya. Penulis juga tercatat sebagai Administrator MIS. Dalam menyelesaikan pendidikan sarjana, penulis mengambil bidang minat Manajemen Informasi (MI).