TUGAS AKHIR – IF184802

ANALISIS KINERJA METODE OVERSAMPLING DAN ENSEMBLE LEARNING PADA PERMASALAHAN KLASIFIKASI DATASET TIDAK SEIMBANG

YULIA NIZA

NRP 05111840000053

Dosen Pembimbing I

Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

NIP 197512202001122002

Dosen Pembimbing II

Dini Adni Navastara, S.Kom., M.Sc.

NIP 198510172015042001

Program Studi S-1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2022

TUGAS AKHIR – IF184802

ANALISIS KINERJA METODE OVERSAMPLING DAN ENSEMBLE LEARNING PADA PERMASALAHAN KLASIFIKASI DATASET TIDAK SEIMBANG

YULIA NIZA

NRP 05111840000053

Dosen Pembimbing I

Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

NIP 197512202001122002

Dosen Pembimbing II

Dini Adni Navastara, S.Kom., M.Sc.

NIP 198510172015042001

Program Studi S-1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2022

FINAL PROJECT – IF184802

PERFORMANCE ANALYSIS OF OVERSAMPLING AND ENSEMBLE LEARNING METHODS ON UNBALANCED DATASET CLASSIFICATION PROBLEMS

YULIA NIZA

NRP 05111840000053

Advisor I

Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

NIP 197512202001122002

Advisor II

Dini Adni Navastara, S.Kom., M.Sc.

NIP 198510172015042001

Study Program Bachelor

Department of Informatics Engineering

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya

2022

LEMBAR PENGESAHAN

ANALISIS KINERJA METODE OVERSAMPLING DAN ENSEMBLE LEARNING PADA PERMASALAHAN KLASIFIKASI DATASET TIDAK SEIMBANG

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Komputer pada
Program Studi S-1
Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh : **YULIA NIZA**
NRP. 05111840000053

Disetujui oleh Tim Penguji Tugas Akhir :

1. Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. Pembimbing I
2. Dini Adni Navastara, S.Kom., M.Sc. Pembimbing II
3. Dr. Eng. Nanik Suciati, S.Kom, M.Kom. Penguji
4. Abdul Munif, S.Kom., M.Sc. Penguji

SURABAYA
Juli, 2022

APPROVAL SHEET

PERFORMANCE ANALYSIS OF OVERSAMPLING AND ENSEMBLE LEARNING METHODS ON UNBALANCED DATASET CLASSIFICATION PROBLEMS

FINAL PROJECT

Submitted to fulfill one of the requirements
for obtaining a degree Computer Science at
Undergraduate Study Program of Bachelor of Informatics
Department of Informatics Engineering
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember

By : **YULIA NIZA**
NRP. 05111840000053

Approved by Final Project Examiner Team:

1. Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. — Advisor I
2. Dini Adni Navastara, S.Kom., M.Sc. — Advisor II
3. Dr. Eng. Nanik Suciati, S.Kom, M.Kom. — Examiner I
4. Abdul Munif, S.Kom., M.Sc. — Examiner

SURABAYA
July, 2022

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP	:	Yulia Niza/ 05111840000053
Program studi	:	S-1 Teknik Informatika
Dosen Pembimbing 1 / NIP	:	Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. / 197512202001122002
Dosen Pembimbing 2/ NIP	:	Dini Adni Navastara, S.Kom., M.Sc. / 198510172015042001

Dengan ini menyatakan bahwa Tugas Akhir dengan judul "Analisis Kinerja Metode *Oversampling* dan *Ensemble Learning* pada Permasalahan Klasifikasi Dataset Tidak Seimbang" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.
Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 26 Juli 2022

Mengetahui Dosen Pembimbing I	Dosen Pembimbing II	Mahasiswa
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.	Dini Adni Navastara, S.Kom., M.Sc.	Yulia Niza
NIP. 197512202001122002	NIP. 198510172015042001	NRP. 05111840000053

STATEMENT OF ORIGINALITY

The undersigned below:

Name of student / NRP	:	Yulia Niza/ 05111840000053
Department	:	S-1 Informatics Engineering
Advisor 1 / NIP	:	Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. / 197512202001122002
Advisor 2/ NIP	:	Dini Adni Navastara, S.Kom., M.Sc. / 198510172015042001

Hereby declare that Final Project with the title of "Performance Analysis of Oversampling And Ensemble Learning Methods on Unbalanced Dataset Classification Problems" is the result of my own work, is original, and is written by following the rules of scientific writing.
If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Surabaya, July 26, 2022

Acknowledge Advisor I	Advisor II	Student
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.	Dini Adni Navastara, S.Kom., M.Sc.	Yulia Niza
NIP. 197512202001122002	NIP. 198510172015042001	NRP. 05111840000053

ABSTRAK

ANALISIS KINERJA METODE OVERSAMPLING DAN ENSEMBLE LEARNING PADA PERMASALAHAN KLASIFIKASI DATASET TIDAK SEIMBANG

Nama Mahasiswa / NRP	**:**	**Yulia Niza / 05111840000053**
Departemen	**:**	**Teknik Informatika FTEIC - ITS**
Dosen Pembimbing I	**:**	**Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.**
Dosen Pembimbing II	**:**	**Dini Adni Navastara, S.Kom., M.Sc.**

Abstrak

Data tidak seimbang (*imbalanced data*) merupakan keadaan apabila terdapat perbedaan jumlah persebaran data yang cukup tinggi antara kelas mayoritas dan kelas minoritas. Permasalahan utama pada kasus data tidak seimbang adalah ketika kelas minoritas yang memiliki informasi lebih penting untuk diprediksi, akan tetapi cenderung disalah klasifikasikan sebagai kelas mayoritas. Penanganan data tidak seimbang sangat penting dilakukan, karena banyaknya kasus klasifikasi pada dunia nyata yang sangat krusial apabila terjadi kesalahan klasifikasi. Salah satu cara untuk menangani permasalahan data tidak seimbang, dilakukan menggunakan metode *oversampling*. Selain itu, pemilihan penggunaan metode klasifikasi juga dapat berpengaruh terhadap performa klasifikasi yang dihasilkan. Untuk itu, digunakan beberapa metode klasifikasi dari *single learning* dan *ensemble learning* untuk mengetahui kinerja dari metode yang dihasilkan.

Pada penelitian ini, dilakukan analisis kinerja metode *oversampling* dan *ensemble learning* berdasarkan evaluasi model yang dihasilkan. Hasil penelitian menunjukkan bahwa metode *oversampling* dapat meningkatkan hasil klasifikasi kelas data minoritas, serta untuk metode *ensemble learning* menghasilkan performa yang cukup baik pada beberapa dataset yang diujikan. Berdasarkan hasil evaluasi menggunakan *confusion matrix*, pada dataset *binary* yang dilakukan pengukuran performa menggunakan *recall*, menghasilkan performa tertinggi masing-masing dataset sebesar 0.6883 untuk dataset *Haberman's Survival* dengan metode klasifikasi KNN dan *oversampling* Borderline-SMOTE, untuk dataset COVID-19 dihasilkan *recall* tertinggi sebesar 0.8391 menggunakan metode klasifikasi KNN dan *oversampling* SMOTE, lalu untuk dataset *Credit Card Fraud* menghasilkan nilai *recall* tertinggi sebesar 0.9476 dengan metode klasifikasi XGBoost dan *oversampling* ROS. Sedangkan untuk dataset *multiclass* pada *Car Evaluation*, dilakukan pengukuran performa menggunakan *F1-Score* yang menghasilkan nilai tertinggi sebesar 0.9989 menggunakan metode XGBoost dan *Borderline*-SMOTE. Hasil dari performa setiap metode *oversampling* dan *ensemble learning* bergantung dengan karakteristik dari masing-masing dataset.

Kata kunci: ***Data Tidak Seimbang, Ensemble Learning, Klasifikasi, Oversampling.***

ABSTRACT

PERFORMANCE ANALYSIS OF OVERSAMPLING AND ENSEMBLE LEARNING METHODS ON UNBALANCED DATASET CLASSIFICATION PROBLEMS

Student Name / NRP	**:**	**Yulia Niza / 05111840000053**
Department	**:**	**Informatics Engineering - ITS**
Advisor	**:**	**Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.**
Second Advisor	**:**	**Dini Adni Navastara, S.Kom., M.Sc.**

Abstract

Unbalanced data is a condition if there is a difference in the amount of data distribution that is quite high between the majority class and the minority class. The main problem in the case of unbalanced data is when the minority class which has more important information to predict, but tends to be misclassified as the majority class. Handling unbalanced data is very important, because there are many classification cases in the real world which are very crucial in the event of a classification error. One way to deal with the problem of unbalanced data is using the oversampling method. In addition, the selection of the use of classification methods can also affect the performance of the resulting classification. For this reason, several classification methods from single learning and ensemble learning are used to determine the performance of the resulting methods.

In this study, an analysis of the performance of the oversampling and ensemble learning methods was carried out based on the evaluation of the resulting model. The results show that the oversampling method can improve the classification results of minority data classes, and the ensemble learning method produces a fairly good performance on some of the tested datasets. Based on the results of the evaluation using a confusion matrix, the binary dataset performed performance measurements using recall, resulting in the highest performance of each dataset of 0.6883 for the Haberman's Survival dataset with the KNN classification method and Borderline-SMOTE oversampling, for the COVID-19 dataset the highest recall was 0.8391. using the KNN classification method and SMOTE oversampling, then for the Credit Card Fraud dataset the highest recall value is 0.9476 with the XGBoost classification method and ROS oversampling. As for the multi-class dataset on Car Evaluation, performance measurements were carried out using the F1-Score which resulted in the highest score of 0.9989 using the XGBoost and Borderline-SMOTE methods. The results of the performance of each oversampling and ensemble learning method depend on the characteristics of each dataset.

Keywords: ***Classification, Ensemble Learning, Oversampling, Unbalanced Data***

KATA PENGANTAR

Bismillahirrahmanirrahim

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul

"ANALISIS KINERJA METODE OVERSAMPLING DAN ENSEMBLE LEARNING PADA PERMASALAHAN KLASIFIKASI DATASET TIDAK SEIMBANG"

Harapan dari penulis, semoga apa yang tertulis di dalam buku tugas akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini dan ke depannya, serta dapat memberikan kontribusi yang nyata.

Dalam pelaksanaan dan pembuatan tugas akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT. Dan Nabi Muhammad SAW. yang telah membimbing penulis selama hidup.
2. Keluarga penulis (Ayah, Ibu, Kakak dan keluarga penulis yang lain) yang selalu memberikan dukungan baik berupa doa, moral, dan material yang tak terhingga kepada penulis, sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Ibu Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. dan Ibu Dini Adni Navastara, S.Kom., M.Sc. selaku pembimbing I dan II yang telah membimbing, memberikan motivasi, dan masukan dalam menyelesaikan Tugas Akhir ini.
4. Seluruh mahasiswa Teknik Informatika ITS angkatan 2018 yang telah menemani penulis dan berjuang bersama dalam menyelesaikan Tugas Akhir ini.
5. Kakak-kakak tingkat Teknik Informatika ITS yang bersedia membagikan ilmu dan pengalaman seputar Tugas Akhir.
6. Semua teman dekat penulis yang memberikan semangat dan menghibur penulis selama pengerjaan Tugas Akhir.
7. Serta semua pihak yang telah membantu penulis dalam menyelesaikan Tugas Akhir ini yang tidak dapat disebutkan penulis satu persatu.

Penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini. Namun, penulis memohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya. Selain itu, penulis berharap laporan Tugas Akhir ini dapat berguna bagi pembaca secara umum. Tetaplah berjuang karena akan ada selalu jalan untuk orang yang tidak pernah menyerah. Semoga kita semua selalu diberi kebahagiaan lahir dan batin dan kesuksesan dunia akhirat. Aamiin.

Surabaya, 15 Juli 2022

Yulia Niza

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Data tidak seimbang (*imbalanced data*) merupakan keadaan apabila terdapat perbedaan jumlah persebaran data yang cukup tinggi antara kelas mayoritas dan kelas minoritas. Kelas mayoritas merupakan kelas yang terdiri dari sebagian besar dari dataset, sedangkan kelas minoritas merupakan kelas yang terdiri dari sebagian kecil dari dataset. Permasalahan utama pada kasus data tidak seimbang adalah ketika kelas minoritas yang memiliki informasi lebih penting untuk diprediksi, akan tetapi cenderung disalah klasifikasikan sebagai kelas mayoritas. Hal tersebut akan berdampak pada performa klasifikasi secara keseluruhan, di mana model menjadi lebih mudah memprediksi data uji sebagai kelas mayoritas dan membuat pengaruh kelas minoritas menjadi sangat kecil, sehingga menyebabkan model yang dihasilkan lebih condong ke kelas mayoritas.

Penanganan data tidak seimbang sangat penting dilakukan, karena banyaknya kasus klasifikasi pada dunia nyata yang sangat krusial apabila terjadi kesalahan klasifikasi, misalnya diagnosis medis, deteksi penipuan kartu kredit, deteksi kerusakan produk, dan sebagainya. Untuk meminimalisir kesalahan hasil prediksi tersebut, maka diperlukan adanya penyeimbangan data antar kelas pada dataset dengan cara *resampling*. Terdapat dua cara untuk melakukan *resampling*, yakni *undersampling* dan *oversampling*. *Undersampling* adalah cara untuk mengurangi sampel data pada kelas mayoritas menjadi sama dengan kelas minoritas, sedangkan *oversampling* adalah cara untuk meratakan distribusi data dengan menduplikasi data kelas minoritas menjadi sejumlah kelas mayoritas. Di sini akan difokuskan *resampling* dengan menggunakan metode *oversampling*, karena setiap data memiliki informasi yang penting dalam pembuatan model *machine learning* yang mana apabila dilakukan *undersampling* akan mengurangi atau menghilangkan informasi yang terdapat pada dataset.

Belakangan ini, terdapat beberapa penelitian mengenai dataset tidak seimbang. Diantaranya, pada studi kasus *credit card fraud detection* dengan membandingkan dengan beberapa metode *machine learning*, yakni *naive bayes*, *k-Nearest Neighbor*, dan *Logistic Regression Classifiers* (Awoyemi, John O; Adetunmbi, Adebayo O; Oluwadare, 2017). Penelitian mengenai cara meningkatkan klasifikasi dataset tidak seimbang menggunakan *oversampling* dan *Gradient Boosting* dengan salah satu datasetnya yakni *haberman's survival* (Cahyana, Nurheri and Khomsah, Siti and Aribowo, 2019). Penelitian lainnya menggunakan dataset multi-kelas tidak seimbang mengenai pemilihan fitur penting untuk klasifikasi data berdasarkan metode *machine learning* dengan salah satu studi kasusnya adalah *Car Evaluation* Dataset (Chen et al., 2020). Serta penelitian mengenai metode yang digunakan untuk meningkatkan akurasi klasifikasi multi-kelas pada data yang tidak seimbang (Sevastianov & Shchetinin, 2020).

Terdapat beberapa penelitian yang sudah pernah dilakukan untuk mengatasi dataset tidak seimbang, salah satunya menggunakan metode *oversampling*. Diantaranya penelitian yang membandingkan performa dari beberapa metode *oversampling*, antara lain *Random Oversampling*(ROS), ADASYN, SMOTE, dan *Borderline-SMOTE* yang dikombinasikan dengan metode *machine learning* yakni *Random Forest, Logistic Regression, dan k-Nearest Neighbor* (KNN) dengan studi kasus dataset *European cardholder transaction* (Wibowo & Fatichah, 2021a). Penelitian lainnya mengenai penggunaan metode *oversampling* berbasis pemangkasan dengan *smotehed bootsrap resampling* untuk penanganan kelas data tidak seimbang pada dataset medis COVID-19 (Wibowo & Fatichah, 2021b). Penelitian lainnya mengenai metode yang efisien untuk menentukan ukuran sampel dalam *oversampling* berdasarkan kompleksitas klasifikasi untuk data yang tidak seimbang (Lee & Kim, 2021).

Berdasarkan penelitian yang pernah dilakukan sebelumnya, metode klasifikasi yang digunakan sebagian besar merupakan *single learning* dan hanya sedikit yang menggunakan *ensemble learning*. Sedangkan perbandingan dan optimalisasi model menggunakan metode *ensemble learning* dapat meningkatkan performa dari model klasifikasi lemah (Farooq et al., 2021)

Pada tugas akhir ini, akan dilakukan analisis perbandingan kinerja dari metode *oversampling*, yakni *Synthetic Minoritasity Over-sampling Technique* (SMOTE), *Borderline*-SMOTE, *Adaptive Synthetic* (ADASYN), dan *Random Over Sampling* (ROS). Dengan menggunakan beberapa dataset, diantaranya *binary-class* yakni *Heberman's Survival, Credit Card Fraud Detection.* COVID-19 dan *multi-class* yakni *Car Evaluation Dataset*. Data hasil *oversampling* kemudian dilakukan pembuatan model menggunakan beberapa metode klasifikasi, diantaranya metode *ensemble learning*, yakni *Random Forest*, Bagging, Adaboost, XGBoost, dan metode *single learning*, yakni *k-Nearest Neighbor* (KNN), *Support Vector Machine* (SVM), dan *Naive Bayes*. Setelah pembuatan model, dilakukan perbandingan dari hasil evaluasi dari setiap model untuk mengetahui performa metode *oversampling* dan metode klasifikasi yang paling optimal dari setiap studi kasus yang diujikan.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut :

1. Bagaimana cara melakukan *preprocessing* pada dataset tidak seimbang?
2. Bagaimana cara menerapkan metode *oversampling* untuk mengatasi data tidak seimbang?
3. Bagaimana cara menerapkan metode *ensemble learning* dalam menyelesaikan permasalahan klasifikasi kelas data tidak seimbang?
4. Bagaimana evaluasi kinerja metode *oversampling* dan *ensemble learning* untuk mengatasi data tidak seimbang?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, di antaranya sebagai berikut :

1. Bahasa pemrograman yang digunakan dalam sistem ini adalah *Python*.
2. Metode *resampling* data yang digunakan terbatas menggunakan metode *oversampling.*
3. Dataset yang digunakan terdiri dari *Heberman's Survival*, *Credit Card Fraud Detection*, COVID-19 dan *Car Evaluation* yang diperoleh dari *Kaggle* dan *UCI Machine Learning Repository*.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah sebagai berikut :

1. Menerapkan metode *oversampling* dan *ensemble learning* untuk melakukan klasifikasi dataset tidak seimbang.
2. Melakukan analisis hasil kinerja metode *oversampling* dan *ensemble learning* pada permasalahan dataset tidak seimbang.

1.5 Manfaat

Manfaat dari Tugas Akhir ini antara lain:

1. Hasil dari analisis dapat dijadikan sebagai bahan referensi dalam mengatasi permasalahan data tidak seimbang pada kasus dunia nyata lainnya.
2. Hasil dari analisis dapat dijadikan sebagai pertimbangan untuk menentukan metode *oversampling* paling sesuai berdasarkan dengan karakteristik dataset tidak seimbang yang digunakan.

BAB 2 TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar atau penjelasan dari metode yang digunakan dalam Tugas Akhir. Bab ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan riset yang berkaitan.

2.1 Hasil Penelitian Terdahulu

Hasil dari penelitian terdahulu yang digunakan antara lain :

1. Penelitian dengan judul *An in-depth performance analysis of the oversampling techniques for high-class imbalanced dataset* yang dilakukan oleh Wibowo dan Chastine Fathichah pada tahun 2021 dengan dataset *European cardholder transaction* dengan metode *oversampling Random Oversampling*(ROS), ADASYN, SMOTE, dan Borderline-SMOTE serta metode klasifikasi *Random Forest, Logistic Regression,* dan *k-Nearest Neighbor* (KNN). Pada penelitian ini terbatas menggunakan satu dataset saja (Wibowo & Fatichah, 2021a).
2. Penelitian dengan judul *Pruning-based oversampling technique with smoothed bootstrap resampling for imbalanced clinical dataset of COVID-19* yang dilakukan oleh Wibowo dan Chastine Fathichah pada tahun 2021 dengan dataset COVID-19 *by Hospital Israelita Albert Einstein* dengan metode *oversampling* antara lain *Random Oversampling*(ROS), ADASYN, SMOTE, *Borderline*-SMOTE, dan TRIM-SBR serta metode klasifikasi *Random Forest, Logistic Regression, Support Vector Machine.* Penelitian ini terbatas menggunakan sebuah metode *ensemble learning* sebuah studi kasus dataset saja (Wibowo & Fatichah, 2021b).
3. Penelitian dengan judul *Selecting critical features for data classifcation based on machine learning methods* yang dilakukan oleh Rung-Ching Chen, Christine Dewi1, Su-Wen Huang, dan Rezzy Eko Caraka pada tahun 2020 menggunakan beberapa dataset tidak seimbang, diantaranya *Bank Marketing, Car Evaluation Database, Human Activity Recognition Using Smartphones* (Chen et al., 2020).
4. Penelitian dengan judul *Improving Imbalanced Dataset Classification Using Oversampling and Gradient Boosting* yang dilakukan oleh Cahyana, Nurheri and Khomsah, Siti, Aribowo, Agus Sasmito pada tahun 2019 menggunakan metode *oversampling* ADASYN, SMOTE, dan *Borderline*-SMOTE serta *Gradient Boosting* yang digunakan sebagai metode klasifikasi. Serta menggunakan tujuh dataset, diantaranya *Mammography, Liver Disorders, Diabetes (Pima Indian), Indian Liver, Habberman, and Immunotherapy* (Cahyana, Nurheri and Khomsah, Siti and Aribowo, 2019).
5. Penelitian berjudul *Credit card fraud detection using machine learning techniques: A comparative analysis* yang dilakukan oleh Awoyemi, John O, Adetunmbi, Adebayo O, Oluwadare S pada tahun 2017 dengan menggunakan teknik *hybrid oversampling* dan *undersampling* pada data tidak seimbang tersebut. Kinerja model dievaluasi menggunakan akurasi, sensitivitas, spesifisitas, presisi, *Matthews correlation coefficient and balanced classification rate*. Hasil penelitian menunjukkan akurasi optimal untuk *naïve bayes, k-nearest neighbor* dan *logistic regression* masing-masing adalah 97,92%, 97,69% dan 54,86%. Hasil perbandingan menunjukkan bahwa *k-nearest neighbor* berkinerja lebih baik daripada *naïve bayes* dan *logistic regression* (Awoyemi, John O; Adetunmbi, Adebayo O; Oluwadare, 2017).
6. Penelitian berjudul *An efficient method to determine sample size in oversampling based on classification complexity for imbalanced data* oleh Dohyun Lee dan Kyoungok Kim tahun 2021 membahas mengenai cara untuk meningkatkan efektivitas dari penggunaan *oversampling* dengan menambahkan data sintesis ke lokasi yang sesuai. Efektivitas ukuran

sampel yang diusulkan dalam *oversampling* dievaluasi menggunakan beberapa algoritma *boosting* dengan metode *oversampling* yang berbeda untuk 16 set data yang tidak seimbang. Hasilnya menunjukkan bahwa ukuran sampel yang diusulkan mencapai kinerja klasifikasi yang lebih baik daripada ukuran sampel untuk mencapai keseimbangan kelas. (Lee & Kim, 2021).

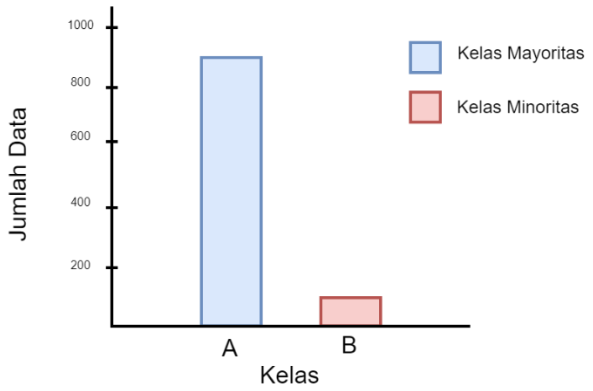
7. Penelitian berjudul *On methods for improving the accuracy of multi-class classification on imbalanced data* yang dilakukan oleh Sevastiaonov dan Shchetinin pada tahun 2020 mengenai metode untuk meningkatkan akurasi pada klasifikasi multi-*class* pada data tidak seimbang, di mana digunakan kombinasi algoritma klasifikasi dan metode seleksi fitur RFE, *Random Forest* dan *Boruta* dengan metode *oversampling* SMOTE dan ADASYN dihasilkan bahwa penggunaan metode *oversampling* dan pemilihan fitur paling informatif dapat meningkatkan hasil akurasi, serta metode paling efektif diperoleh dengan menggunakan *Random Forest* dan *oversampling* ADASYN (Sevastianov & Shchetinin, 2020).
8. Penelitian berjudul *Handling imbalanced data using ensemble learning in software defect prediction* yang dilakukan oleh Malhotra R, Jain J pada tahun 2020 mengenai cara mengatasi data tidak seimbang menggunakan metode *ensamble learning.* Pada penelitian ini dihasilkan bahwa penggunaan teknik *resampling* sebelum mengklasifikasi menggunakan metode *ensemble* telah meningkatkan prediksi model secara signifikan dibandingkan dengan model boosting klasik. RUSBoost adalah pemenang tak terbantahkan di antara semua diikuti oleh MSMOTEBoost dan SMOTEBoost (Malhotra & Jain, 2020).
9. Penelitian berjudul *Predictive modeling for sustainable high-performance concrete from industrial wastes: A comparison and optimization of models using ensemble learners* pada tahun 2021 yang menghasilkan bahwa respon model individu ditingkatkan dengan menggunakan algoritma bagging dan *boosting*. Secara keseluruhan, *random forest* dan *decision tree* dengan bagging memberikan kinerja model yang kuat dengan R2 0.92 dengan *error* paling kecil. Secara rata-rata, model *ensemble* dalam *machine learning* dapat meningkatkan performa model (Farooq et al., 2021).
10. Penelitian lainnya yang pernah dilakukan yakni *A Comparative Review of SMOTE and ADASYN in Imbalanced Data Classification* yang dilakukan oleh Brandt J, Lanzén E pada tahun 2020. Penelitian ini membandingkan metode *oversampling* SMOTE dan ADASYN yang dilakukan pada tiga set data yang tidak seimbang menggunakan tiga model klasifikasi dan metrik evaluasi yang berbeda, sambil memvariasikan cara data diproses sebelumnya. Hasilnya menunjukkan bahwa SMOTE dan ADASYN meningkatkan kinerja pengklasifikasi dalam banyak kasus. Juga ditemukan bahwa SVM dalam hubungannya dengan SMOTE berkinerja lebih baik daripada dengan ADASYN karena tingkat ketidakseimbangan kelas meningkat. Selanjutnya, baik SMOTE dan ADASYN meningkatkan kinerja relatif dari *Random Forest* seiring dengan meningkatnya tingkat ketidakseimbangan kelas. Namun, tidak ada metode pra-pemrosesan yang secara konsisten mengungguli yang lain dalam kontribusinya terhadap kinerja yang lebih baik karena tingkat ketidakseimbangan kelas bervariasi (Brandt & Lanzén, 2020).

2.2 Dasar Teori

2.2.1 Data Tidak Seimbang

Data tidak seimbang *(imbalanced data)* merupakan suatu keadaan di mana distribusi kelas data tidak sama, di mana jumlah kelas data (*instance*) yang satu lebih sedikit atau lebih banyak dibanding dengan jumlah kelas data lainnya. Kelompok kelas data yang lebih sedikit dikenal sebagai kelas minoritas (*minority*), sedangkan kelompok kelas data yang lebih banyak

disebut dengan kelas mayoritas (*majority*). Ilustrasi dari dataset tidak seimbang dapat dilihat pada Gambar 2.1.



Gambar 2.1 Ilustrasi Data Tidak Seimbang

Klasifikasi pada data dengan kelas tidak seimbang merupakan masalah utama pada *machine learning* dan *data mining*, misalnya pada masalah medis, masalah klasifikasi teks, dan sosial media. Jika bekerja pada data tidak seimbang, hampir semua algoritma klasifikasi akan menghasilkan akurasi yang jauh lebih tinggi untuk kelas mayoritas daripada kelas minoritas. Perbedaan ini merupakan suatu indikator performa klasifikasi yang buruk. Pada beberapa kasus, kelas minoritas justru lebih penting untuk diidentifikasi daripada kelas mayoritas (Siringoringo, 2018).

Beberapa metode untuk mengatasi ketidakseimbangan kelas dapat dibagi menjadi tiga kategori. Pertama, dengan teknik tingkat data yang berusaha menyeimbangkan distribusi data dengan metode *oversampling* dan *undersampling*. Kedua, pendekatan tingkat algoritma yaitu dengan mengembangkan algoritma baru atau memodifikasi metode yang ada untuk memperhitungkan arti dari kelas minor. Ketiga, yaitu dengan mengombinasikan pendekatan algoritma dan pendekatan level data (Syukron & Subekti, 2018).

2.2.2 *Scaling* Dataset

Banyak algoritma *machine learning* berkinerja lebih baik atau lebih cepat ketika fitur berada pada skala yang relatif sama dan/atau mendekati terdistribusi normal. Tujuan penerapan *scaling* data adalah untuk memastikan data berada pada skala yang hampir sama sehingga setiap fitur sama pentingnya dan membuatnya lebih mudah untuk diproses oleh sebagian besar algoritma *machine learning*. Terdapat beberapa cara untuk melakukan *scaling* dataset, dua diantaranya yang digunakan pada penelitian ini adalah *Standard Scaler* dan *MinMaxScaler*.

1. *Standard Scaler*

Standard Scaler menstandardisasi fitur dengan mengurangi setiap data dengan *mean* dan kemudian menskalakan ke varians unit. Varians unit berarti membagi semua nilai dengan standar deviasi. Standardisasi mengikuti kaidah dari distribusi *Gaussian* (atau distribusi Normal). Formula dari *Standard Scaler* dapat dilihat pada persamaan 2.1.

$$Z = \frac{x_i - \mu}{\sigma} \tag{2.1}$$

Hasil dari *Standard Scaler* dilambangkan dengan *Z*, dengan *x* merupakan nilai setiap data, μ adalah *mean* (rata-rata) dan σ adalah standar deviasi dari *mean*. *Standard Scaler* menghasilkan distribusi dengan standar deviasi sama dengan 1, sehingga *Standard Scaler*

membuat rata-rata distribusi 0 dan sekitar 68% dari nilai akan berada di antara -1 dan 1 (Ferreira et al., 2019).

2. *MinMax Scaler*

Normalisasi baik digunakan ketika diketahui distribusi data tidak mengikuti kaidah distribusi *Gaussian*. Ini dapat berguna dalam algoritma yang tidak mengasumsikan distribusi data apa pun, seperti *K-Nearest Neighbors* dan *Neural Networks* (Ferreira et al., 2019). Formula *MinMax* Scaler dapat dilihat pada persamaan 2.2.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.2}$$

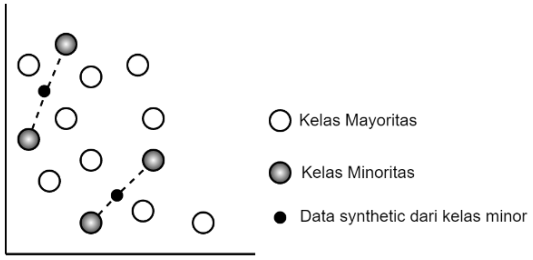
Dengan x_{norm} adalah hasil dari *MinMax Scaler x* adalah data, *x min* adalah data terendah, dan x_{max} adalah data tertinggi. Rentang *default* untuk fitur yang dikembalikan oleh *MinMaxScaler* adalah 0 hingga 1. Serta, *MinMaxScaler* tidak mengurangi pentingnya *outlier*.

2.2.3 Metode *Oversampling* pada Klasfikasi Dataset Tidak Seimbang

Sampling merupakan bagian dari ilmu statistik yang memfokuskan penelitian terhadap pemilihan data yang dihasilkan dari satu kumpulan populasi data. Metode sampling atau yang lebih dikenal dengan *resample* adalah metode umum yang digunakan dalam menyelesaikan permasalahan data tidak seimbang. Dengan adanya penerapan sampling pada data yang tidak seimbang, tingkat ketidakseimbangan semakin kecil dan klasifikasi dapat dilakukan dengan tepat. Untuk metode *oversampling* dilakukan dengan menyeimbangkan jumlah distribusi data dengan meningkatkan jumlah data kelas minor. Metode yang digunakan dalam penelitian ini adalah *Synthetic Minoritasity Oversampling Technique* (SMOTE), *Borderline*-SMOTE, *Adaptive Synthetic* (ADASYN), dan *Random Over Sampling* (ROS).

1. *Synthetic Minority Oversampling Technique* (SMOTE)

Synthetic Minority Oversampling Technique (SMOTE) diperkenalkan oleh Nithes V. Chawla sebagai salah satu teknik untuk menangani data yang tidak seimbang (Muqiit WS & Nooraeni, 2020). SMOTE merupakan teknik yang digunakan untuk menyeimbangkan jumlah distribusi data pada sampel kelas data minoritas dengan cara menyeleksi data sampel tersebut hingga jumlah sampel data menjadi seimbang dengan jumlah sampel data kelas mayoritas (Kasanah et al., 2019). Metode SMOTE bekerja dengan cara mengelompokkan data terdekat yang dipilih berdasarkan dari jarak *euclidean* antara kedua data. SMOTE bekerja dengan memanfaatkan algoritma *k-Neighbor* untuk membuat data sintetis. SMOTE pertama dimulai dengan memilih data acak dari kelas minoritas, kemudian ke tetangga terdekat dari data tersebut ditetapkan. Data sintetik kemudian akan dibuat di antara data acak dan *k-nearest* yang dipilih secara acak. Ilustrasi mengenai algoritma SMOTE dapat dilihat pada Gambar 2.2.

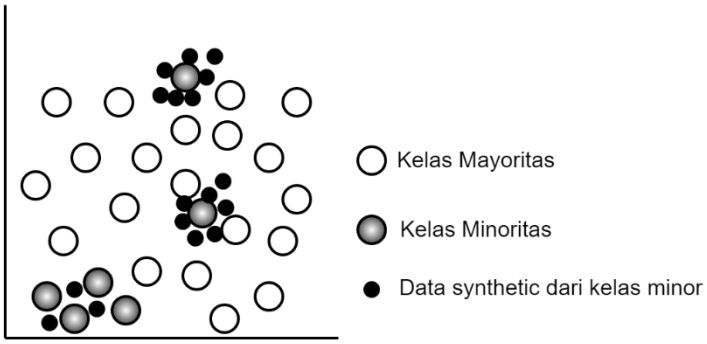


(Sastrawan, 2009)

Gambar 2.2 Ilustrasi SMOTE

2. *Adaptive Synthetic* (ADASYN)

ADASYN menghitung distribusi densitas di kelas minoritas sebelum menduplikasi data berdasarkan kriteria yang diperoleh saat menghitung densitas kelas (Brandt & Lanzén, 2020). Pendekatan ini membantu untuk fokus pada titik tengah area kelas minoritas tergantung pada dataset. Pembangkitan data sintetis akan berbanding terbalik dengan kepadatan kelas minoritas. Artinya, lebih banyak data sintetis dibuat di wilayah ruang fitur yang kepadatan minoritasnya rendah, dan lebih sedikit atau tidak ada pada ruang yang kepadatannya tinggi. Ilustrasi mengenai algoritma ADASYN dapat dilihat pada Gambar 2.3.

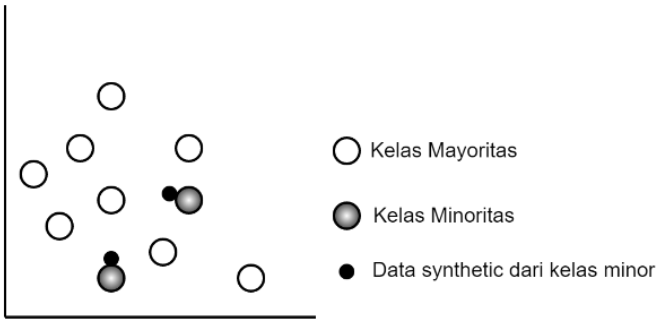


(AlShourbaji et al., 2021)

Gambar 2.3 Ilustrasi ADASYN

3. *Random Over Sampling* (ROS)

Random Oversampling (ROS) bekerja dengan memilih data kelas minor untuk diduplikasi. Hasil dari *Random Oversampling* tidak selalu meningkatkan prediksi kelas minor. Apabila data kelas minor diduplikasi dalam jumlah yang besar, maka akan sulit untuk mengidentifikasi data yang memiliki kemiripan karakteristik namun berada di kelas yang berbeda. ROS melakukan duplikasi dengan memilih satu set acak kelas minoritas untuk replikasi data acak (Han et al., 2019). Karena proses pengambilan sampel dilakukan secara acak, maka *Random Oversampling* memiliki kelemahan yaitu membutuhkan waktu pelatihan yang lama, dan kemungkinan terjadinya *overfitting*. Ilustrasi mengenai algoritma ROS dapat dilihat pada Gambar 2.4.



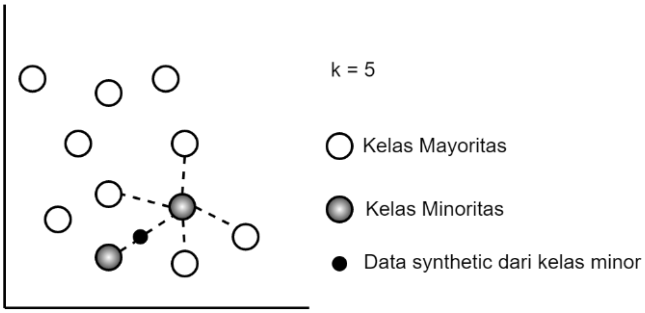
(Sastrawan, 2009)

Gambar 2.4 Ilustrasi ROS

4. *Borderline*-SMOTE

Borderline-SMOTE merupakan metode yang menerapkan metode SMOTE pada data *borderline*, karena dianggap sering kesalahan klasifikasi. *Synthetic* data baru akan dibangkitkan

sepanjang garis antara kelas minor dan tetangga terdekat yang dipilih sebelumnya (AlShourbaji et al., 2021). Langkah yang dilakukan pada *borderline*-SMOTE adalah dengan menentukan *k-nearest neighbor* untuk tiap data kelas minor lalu memeriksa apakah data kelas minor masuk himpunan *danger* atau tidak. Himpunan *danger* adalah himpunan yang berisi data kelas minor dengan mayoritas *nearest neighbor*-nya adalah data kelas mayor. Untuk tiap data di himpunan *danger*, dilakukan proses SMOTE. Ilustrasi mengenai algoritma *Borderline*-SMOTE dapat dilihat pada Gambar 2.5.



(AlShourbaji et al., 2021)

Gambar 2.5 Ilustrasi Borderline-SMOTE

2.2.4 Metode *Single Learning* pada Klasifikasi Dataset Tidak Seimbang

Pada metode *single learning*, digunakan sebuah klasifier untuk membuat keputusan tanpa dipengaruhi oleh model lainnya. Beberapa metode klasifikasi *single learning* yang digunakan pada penelitian ini yang dijelaskan pada subbab berikut ini.

1. *K-Nearest Neighboirs* (KNN)

K-Nearest Neighbors (KNN) merupakan algoritma pembelajaran yang bersifat non-parametrik dan *lazy*. Non-parametrik berarti tidak ada asumsi untuk distribusi data yang mendasarinya. Dengan kata lain, struktur model ditentukan dari dataset. Hal ini sangat membantu dalam praktik, di mana sebagian besar dataset di dunia nyata tidak mengikuti asumsi teoritis matematika. Sedangkan *lazy* berarti bahwa ia tidak menggunakan titik data pelatihan untuk melakukan *generalization*. Ini berarti fase pelatihannya berlangsung cukup cepat, namun kurangnya *generalization*, berarti KNN menyimpan semua data pelatihan, sehingga semua data pelatihan diperlukan selama fase pengujian. Dalam algoritma KNN, untuk setiap titik data uji akan dilihat titik data pelatihan terdekat sejumlah *k* dengan diukur berdasarkan salah satu fungsi jarak pada (2.3), atau (2.4), atau (2.5) dengan *k* merupakan jumlah dimensi, x_i, y_i merupakan titik koordinat data dan *p* adalah l_p – *norm*. Kemudian mengambil kelas yang paling sering muncul (*majority vote*), lalu menetapkan kelas tersebut ke data uji (Gandhi, 2018).

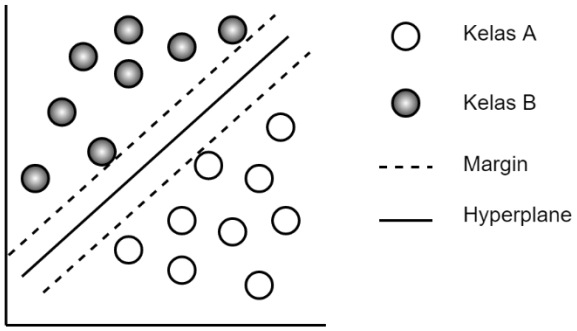
$$Jarak\ Euclidean = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \tag{2.3}$$

$$Jarak\ Manhattan = \sum_{i=1}^{k}|x_i - y_i| \tag{2.4}$$

$$Jarak\ Minkowski = \sum_{i=1}^{k}(|x_i - y_i|^p)^{\frac{1}{p}} \tag{2.5}$$

2. *Support Vector Machine* (SVM)

Support Vector Machine (SVM) merupakan analisis klasifikasi yang menggunakan bidang pemisah *(hyperplane)* dalam melakukan klasifikasi. Pada ruang berdimensi *d*, bidang pemisah akan berdimensi *d−1*. SVM dapat menghasilkan berbagai kemungkinan bidang pemisah. Bidang pemisah terbaik yaitu yang memiliki margin paling besar *(maximal margin hyperplane)*. Margin adalah jarak terdekat amatan data latih dengan bidang pemisah. Amatan yang memiliki jarak terdekat dengan bidang pemisah disebut *support* (Pratama et al., 2018). SVM berusaha menemukan *hyperplane* yang terbaik pada *input space*. Prinsip dasar SVM adalah *linear classifier*, dan selanjutnya dikembangkan agar dapat bekerja pada problem *non-linear* dengan memasukkan konsep *kernel trick* pada ruang kerja berdimensi tinggi. Ilustrasi dari algoritma SVM dapat dilihat pada Gambar 2.6.



Gambar 2.6 Ilustrasi *Support Vector Machine* (SVM)

3. Naïve Bayes

Naïve Bayes merupakan sebuah pengklasifikasi probabilitas sederhana yang mengaplikasikan *Teorema Bayes* dengan asumsi tidak ada ketergantungan *(independent)* yang tinggi. Salah satu keuntungan algoritme *Naive Bayes* ialah dalam menentukan estimasi parameter, yang diperlukan dalam proses pengklasifikasian hanya jumlah data pelatihan yang kecil. Karena diasumsikan sebagai *variable independent*, maka hanya varians dari suatu variabel dalam sebuah kelas yang dibutuhkan untuk menentukan klasifikasi, bukan keseluruhan dari matriks kovarians (Yang, 2018). Persamaan umum dari *Naive Bayes* dapat dilihat pada persamaan 2.6.

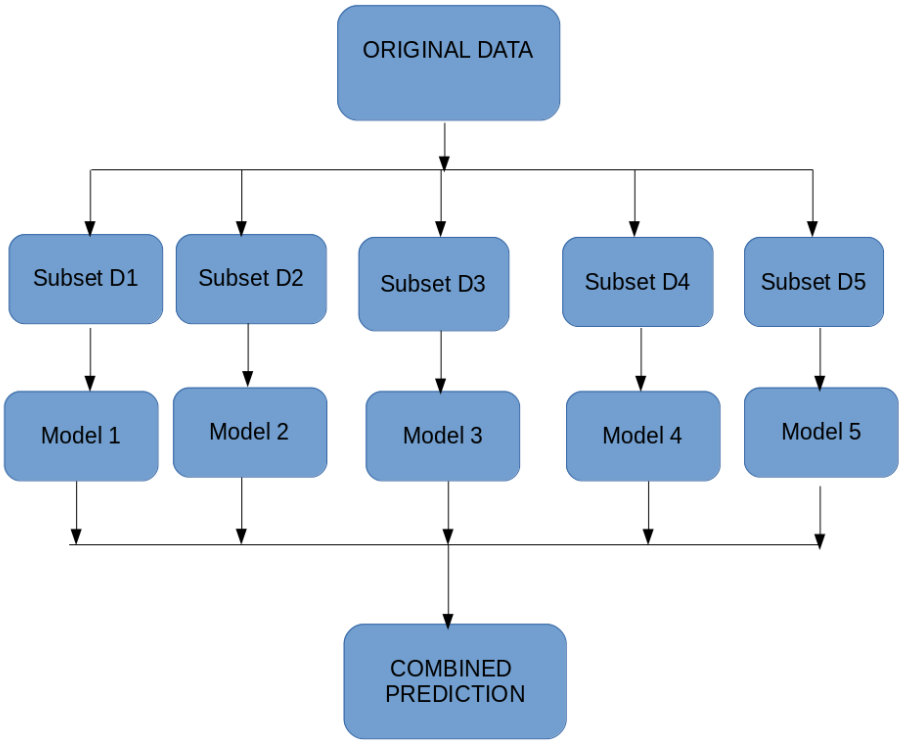
$$P(C|F1 \dots Fn) = \frac{P(C)P(F1 \dots Fn|C)}{P(F1 \dots Fn)} \quad (2.6)$$

Di mana variabel *C* merepresentasikan kelas, sementara variabel $F_1 \dots Fn$ merepresentasikan berbagai karakteristik petunjuk yang dibutuhkan untuk melakukan klasifikasi. Maka persamaan tersebut menjelaskan bahwa peluang masuknya sampel dengan karakteristik tertentu dalam kelas *C* adalah peluang munculnya kelas *C* sebelum masuknya sampel tersebut, dikali dengan peluang kemunculan berbagai karakteristik sampel pada kelas *C* dibagi dengan peluang kemunculan karakteristik-karakteristik sampel secara global *(evidence)*.

2.2.5 Metode *Ensemble Learning* pada Klasifikasi Dataset Tidak Seimbang

Metode *Ensemble Learning* pada *machine learning* yakni menggabungkan keputusan dari beberapa model untuk meningkatkan kinerja model secara keseluruhan. Terdapat beberapa metode *ensemble* yang digunakan pada penelitian ini.

1. Bagging



Gambar 2.7 Ilustrasi Bagging

Bagging adalah singkatan dari *Bootstrap Aggregating*, pertama kali diusulkan oleh Breiman pada tahun 1996. *Bootstrapping* adalah teknik pengambilan sampel di mana membuat subset pengamatan dari dataset asli, dengan penggantian. Teknik Bagging menggunakan himpunan bagian (*bags*) untuk mendapatkan gambaran yang seimbang tentang distribusi (set lengkap). Ukuran subset yang dibuat untuk Bagging lebih kecil dari set aslinya (Ghojogh & Crowley, 2019). Tahapan bagging secara umum dapat dilihat di bawah ini dengan ilustrasi pada Gambar 2.7.

1) Beberapa *subset* dibuat dari dataset asli, memilih pengamatan dengan penggantian.
2) Model dasar (model lemah) dibuat pada masing-masing himpunan bagian ini.
3) Model berjalan secara paralel dan independen satu sama lain.
4) Prediksi akhir ditentukan dengan menggabungkan prediksi dari semua model.

2. *Random Forest*

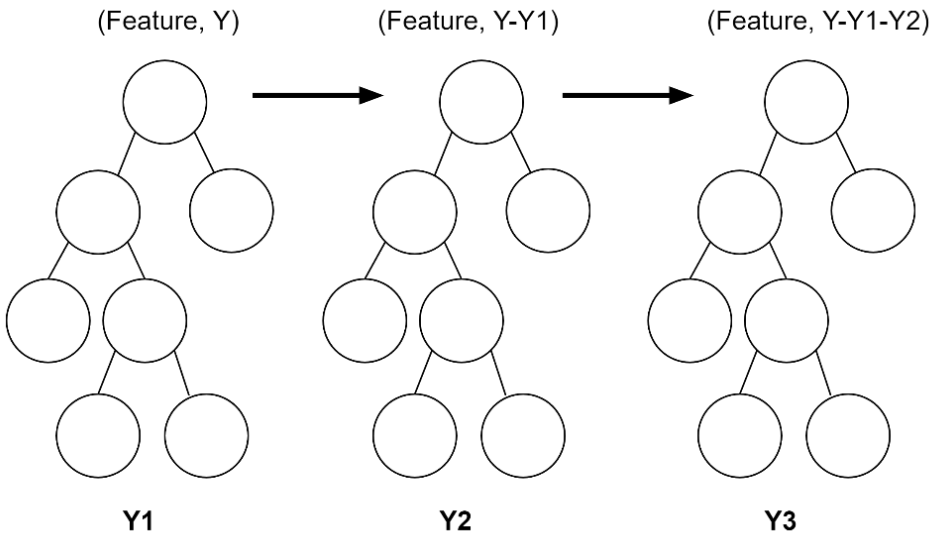
Algoritma *Random Forest* merupakan salah satu algoritma *machine learning* yang menggunakan teknik bagging. Algoritma ini dilakukan dengan mengembangkan berapa *decision tree* berdasarkan seleksi data dan variabel yang dilakukan secara acak. Algoritma ini dapat menyediakan variabel dependen pada kelas sejumlah *tree* yang dibentuk. Skema ini pertama kali dicetuskan oleh Leo Breiman untuk membangun prediktor dengan sekumpulan *Decision Tree* yang berkembang secara acak pada sub ruang data (Jackins et al., 2021). Kelas yang dihasilkan dari proses klasifikasi dipilih dari kelas yang paling banyak yang dihasilkan oleh pohon keputusan yang ada. Berikut adalah langkah untuk melakukan *random forest* :

1) Subset acak dibuat dari dataset asli (*bootstrapping*).
2) Pada setiap *node* dalam *decision tree*, hanya sekumpulan fitur acak yang dipertimbangkan untuk memutuskan *split* terbaik.
3) Sebuah model *decision tree* dipasang pada masing-masing *subsets*.

4) Prediksi akhir dihitung dengan merata-ratakan prediksi dari semua *decision tree*.

3. XGBoost

Metode *extreme gradient boosting* atau XGBoost merupakan sebuah algoritma *boosting* berbasis pohon keputusan atau pohon regresi. Gambaran umum algoritma *boosting* berbasis pohon regresi ditampilkan pada gambar 2.8 (Jiang et al., 2019). Proses pembelajaran pohon pertama dari data latih *(feature, Y)* memperoleh hasil estimasi pertama *(Y1)*. Pohon ke dua melakukan proses pembelajaran dari data latih *(feature, Y-Y1)*, dimana nilai *Y-Y1* merupakan selisih antara label sebenarnya dengan label prediksi pada tahap sebelumnya. Pohon ketiga melakukan proses pembelajaran dari data *(feature, Y-Y1-Y2)* dan menghasilkan estimasi *Y3*. Dengan cara tersebut, nilai *error* dapat direduksi dengan efektif.



Gambar 2.8 Ilustrasi Pohon Regresi XGBoost

4. Adaboost

Adaptive Boosting (Adaboost) adalah salah satu algoritma *boosting* yang ditemukan oleh Freund and Schapire pada tahun 1997. *Boosting* sendiri merupakan konsep pembelajaran mesin dengan mengombinasikan beberapa *classifier* yang lemah untuk membentuk classifier yang kuat. Sedangkan *Adaptive boosting* merupakan algoritma *boosting* yang mampu menyesuaikan secara adaptif nilai *error* yang dihasilkan oleh *classifier* lemah untuk dijadikan acuan pada proses pelatihan *classifier* berikutnya (Ghojogh & Crowley, 2019). Berikut adalah langkah-langkah untuk melakukan algoritma AdaBoost:

1) Awalnya, semua pengamatan dalam dataset diberi bobot yang sama.
2) Sebuah model dibangun di atas *subset* data.
3) Dengan menggunakan model tersebut, prediksi diterapkan pada seluruh dataset.
4) *Error* dihitung dengan membandingkan antara hasil prediksi dan nilai aktual.
5) Saat membuat model berikutnya, bobot yang lebih tinggi diberikan pada titik data yang diprediksi salah.
6) Bobot dapat ditentukan dengan menggunakan *error value*. Misalnya, semakin tinggi kesalahan, semakin banyak bobot yang diberikan untuk pengamatan.
7) Proses ini diulang sampai fungsi *error* tidak berubah, atau batas maksimum jumlah *estimator* tercapai.

2.2.6 Metode Evaluasi pada Klasifikasi Data Tidak Seimbang

1. *Confusion Matrix*

Metode pengukuran performa memiliki peranan yang sangat penting untuk mengevaluasi kinerja suatu metode klasifikasi. *Confusion matrix* merupakan alat yang paling populer dalam mengevaluasi performa klasifikasi. Pada Tabel 2.1 ditampilkan *confusion matrix*.

Tabel 2.1 *Confusion Matrix*

		Kelas Prediksi	
		Negative	***Positive***
Kelas Sebenarnya	***Negative***	*True Negative* (TN)	*False Positive* (FP)
	Positive	*False Negative* (FN)	*True Positive* (TP)

True Positive (TP) dan *True Negative* (TN) merupakan data kelas positif dan negatif yang diklasifikasikan dengan tepat. *False Positive* (FP) merupakan data kelas negatif yang salah terprediksi menjadi kelas positif atau biasa disebut sebagai *Type* I *Error*, sedangkan *False Negative* (FN) merupakan data kelas positif salah terprediksi menjadi kelas negatif atau biasa disebut sebagai *Type II Error*. Berdasarkan *confusion matrix* pada Tabel 2.1, dapat ditentukan kriteria performa seperti *Accuracy, Precision, Recall, specificity,* dan yang lainnya.

Akurasi (*accuracy*) merupakan kriteria yang paling umum untuk mengukur kinerja klasifikasi, tetapi apabila bekerja pada kelas tidak seimbang, kriteria ini kurang tepat karena kelas minoritas akan memiliki sumbangsih yang kecil. Presisi merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif. *Recall* atau sensitivitas merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Nilai *F1-score* meningkat secara proporsional dengan peningkatan presisi dan *recall*. Semakin tinggi nilai *F-measure*, semakin baik pengklasifikasi memprediksi sampel positif. *F1-score* menampilkan keseimbangan antara presisi dan sensitivitas. Di mana nilai yang dihasilkan akan 0 jika presisi atau sensitivitasnya 0, begitu pula sebaliknya (Tharwat, 2018).

$$Presisi = \frac{TP}{TP + FP} \tag{2.6}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

$$F1 - Score = 2 \times \frac{Presisi \times Recall}{Presisi + Recall} \tag{2.8}$$

Persamaan presisi diperoleh dengan cara membandingkan *True Positif* (TP) dengan keseluruhan hasil yang diprediksi positif (TP+FP), dengan formula yang dapat dilihat pada persamaan 2.6. Untuk *recall* merupakan rasio prediksi benar positif (TP) dibandingkan dengan keseluruhan data yang benar positif (TP+FN) dengan formula yang dapat dilihat pada persamaan 2.7. Lalu untuk *F1-Score* merupakan perbandingan rata-rata presisi dan *recall* yang dibobotkan, dengan formula yang dapat dilihat pada persamaan 2.8.

Pada penelitian ini, nilai *recall* digunakan sebagai pengukuran performa untuk dataset *binary*. Sedangkan untuk dataset *multiclass*, pengukuran performa dilakukan menggunakan *F1-Score*. Pemilihan pengukuran performa *recall* untuk dataset *binary* karena *recall* menghitung banyaknya data kelas minoritas yang diprediksi benar *(True Positive)* dibandingkan dengan banyaknya data kelas minoritas sebenarnya *(True Positive + False Negative)*. Di mana pada

studi kasus yang digunakan, nilai *False Negative* merupakan *Type II Error* yang harus dihindari atau memberikan dampak yang lebih berbahaya dibandingkan dengan *Type I Error*. Dengan pengukuran *recall*, semakin sedikit data yang terprediksi sebagai *False Negative,* maka semakin tinggi nilai *recall* yang dihasilkan, dengan begitu *Type II Error* dapat dihindari dengan memilih performa yang menghasilkan nilai *recall* yang tinggi.

Untuk dataset *multiclass,* digunakan pengukuran performa menggunakan *F1-Score.* Hal ini dikarenakan, semua kelas pada dataset tersebut diaggap penting, sehingga perlu menghasilkan klasifikasi yang memberikan nilai terbaik untuk semua kelas. Nilai terbaik tersebut diperoleh dengan memilih model yang mementingkan mementingkan aspek *recall* serta *presisi* yang dihasilkan melalui pengukuran *F1-Score.*

(Halaman ini sengaja dikosongkan)

BAB 3 METODOLOGI

Pada bab ini menjelaskan mengenai studi literatur, dataset yang digunakan, rancangan sistem, dan implementasi. Studi literatur merupakan bahan literatur yang digunakan sebagai acuan utama pada pembuatan penelitian ini. Selain itu, dijelaskan pula dataset yang digunakan pada penelitian ini, beserta karakteristik dari setiap dataset tersebut. Terdapat pula rancangan sistem yang berisi tahapan pengerjaan sistem pada penelitian ini. Serta terdapat implementasi yang menerangkan pembuatan implementasi sistem pada penelitian ini.

3.1 Studi Literatur

Studi literatur yang digunakan di sini yakni mengacu pada penelitian berjudul *An in-depth performance analysis of the oversampling techniques for high-class imbalanced dataset* (Wibowo & Fatichah, 2021). Pada penelitian tersebut, digunakan dataset *European cardholder transaction* dengan metode *oversampling Random Oversampling* (ROS), ADASYN, SMOTE, dan *Borderline*-SMOTE serta metode klasifikasi *Random Forest, Logistic Regression,* dan *k-Nearest Neighbor* (KNN).

Pada Penelitian tersebut, menjelaskan bahwa model dengan menggunakan dataset tidak seimbang akan menghasilkan klasifikasi yang bias, di mana model cenderung memprediksi data sebagai kelas mayoritas dan mengabaikan kelas minoritas. Untuk itu, perlu dilakukan *oversampling* untuk menangani permasalahan tersebut. Pada penelitian tersebut digunakan empat metode *oversampling*, sebuah metode *ensemble learning* dan menggunakan sebuah dataset saja. Dengan menggunakan acuan penelitian tersebut, maka melalui penelitian ini dikembangkan dengan menambahkan dataset yang diujikan menggunakan beberapa dataset *imbalanced* yang memiliki karakteristik berbeda-beda, serta digunakan beberapa metode *ensemble learning* sebagai pembanding dari penggunaan metode klasifikasi *single learning.*

3.2 Dataset yang digunakan

Dataset yang digunakan di sini terdiri dari beberapa kategori, yakni *binary class* dan *multiclass*. Pada *binary class,* digunakan tiga dataset yang memiliki karakteristik berbeda-beda untuk mendapatkan hasil perbandingan analisis terhadap metode yang digunakan, serta sebuah dataset *multiclass* yang dijelaskan pada subbab berikut.

3.2.1 Heberman's Survival Dataset

Dataset ini berisi kasus-kasus dari sebuah penelitian yang dilakukan antara tahun 1958 dan 1970 di Rumah Sakit Billings Universitas Chicago tentang kelangsungan hidup pasien yang telah menjalani operasi untuk kanker payudara. Dataset ini terdiri dari tiga fitur dengan tipe data numerik yakni *patient_age* (usia pasien pada saat operasi), *operation_year* (tahun pada saat pasien dioperasi yang dikurangi dengan 1900), *axillary_nodes* (Jumlah node aksila positif yang terdeteksi), dan target kelas *survival status*. Target kelas pada dataset ini terdiri dari kelas 1 untuk pasien yang bertahan lebih dari 5 tahun dengan 225 data (73,5%) dan kelas 2 untuk pasien yang tidak bisa bertahan kurang dari 5 tahun dengan 81 data (26,5%). Karakteristik umum dari dataset ini dapat dilihat pada Tabel 3.1.

Tabel 3.1 Karakteristik Dataset *Haberman's Survival*

Karakteristik	**Nilai**
Total Data	306
Jumlah Fitur	3
Perbandingan Data negatif : Positif	73.5 : 26.5
Frekuensi data *null* pada fitur	0

3.2.2 COVID-19 Dataset

Kumpulan data ini berisi data anonim dari pasien yang terlihat di Rumah Sakit Israelta Albert Einstein, di São Paulo, Brasil, dan yang memiliki sampel yang dikumpulkan untuk melakukan RT-PCR SARS-CoV-2 dan tes laboratorium tambahan selama kunjungan ke rumah sakit. Semua data dianonimkan mengikuti praktik dan rekomendasi internasional terbaik. Semua data klinis di standarisasi untuk memiliki rata-rata nol dan standar deviasi unit. Karakteristik dari dataset tersebut dapat dilihat pada Tabel 3.2.

Tabel 3.2 Karakteristik Dataset COVID-19 Sebelum dilakukan Praproses

Karakteristik	**Nilai**
Total Data	5644
Jumlah Fitur	111
Perbandingan Data negatif : Positif	90:10
Frekuensi data *null* pada fitur	65-100%

Seperti terlihat pada Tabel 3.2, banyak data yang hilang karena pada pengambilan keputusan tenaga medis membutuhkan proses yang kompleks seperti pengambilan rekam medis setiap pasien, pengaduan menderita, dan hasil laboratorium. Hal ini karena setiap pasien memiliki penanganan yang berbeda, sehingga setiap pasien tidak selalu memiliki aspek yang sama untuk semua fitur tersebut. Aspek terpenting dari dataset ini adalah ketidakseimbangan fitur dataset dengan perbandingan 10% untuk positif COVID-19 dan 90% untuk pasien negatif COVID-19. Total data 5.644 dengan jumlah fitur 111 dengan frekuensi data null pada fitur 65% 100%. Untuk itu, pada penelitian ini digunakan dataset yang telah dilakukan praproses sebelumnya, yakni *data cleaning, data reduction* dan *data transformation* sehingga dihasilkan dataset dengan karakteristik pada Tabel 3.3.

Tabel 3.3 Karakteristik Dataset COVID-19 Setelah dilakukan Praproses

Karakteristik	**Nilai**
Total Data	602
Jumlah Fitur	13
Perbandingan Data negatif : Positif	86:14
Frekuensi data *null* pada fitur	0

3.2.3 Credit Card Fraud Dataset

Dataset *Credit Card Fraud* berisi transaksi yang dilakukan dengan kartu kredit pada bulan September 2013 oleh pemegang kartu Eropa. Dataset ini menyajikan transaksi yang terjadi dalam dua hari, dengan 492 data penipuan dari 284.807 transaksi yang memiliki rasio ketidakseimbangan antar kelas yang cukup tinggi, di mana kelas positif (penipuan) menyumbang 0.172% dari semua transaksi. Deteksi penipuan kartu kredit penting untuk dilakukan agar perusahaan kartu kredit dapat mengenali transaksi kartu kredit palsu sehingga pelanggan tidak dikenakan biaya untuk barang yang tidak mereka beli.

Dataset ini terdapat variabel numerik yang merupakan hasil dari transformasi PCA. Karena masalah kerahasiaan, penyedia dataset tidak dapat menyediakan fitur asli dan lebih banyak mengenai informasi latar belakang data tersebut. Dataset terdiri dari 28 fitur yakni V1, V2, … V28 yang merupakan komponen utama yang diperoleh dengan PCA, fitur yang belum diubah dengan PCA adalah 'Waktu' dan 'Jumlah'. Fitur 'Waktu' berisi detik yang berlalu antara setiap transaksi dan transaksi pertama dalam dataset. Fitur 'Jumlah' adalah Jumlah dari transaksi yang dilakukan. Fitur 'Kelas' di sini terdiri dari variabel respon dengan nilai 1 apabila terjadi penipuan*(fraud)* dan 0 sebaliknya*(no fraud)*. Pada Tabel 3.4 ditampilkan karakterikstik umum dari dataset *Credit Card Fraud*.

Tabel 3.4 Karateristik Dataset *Credit Card Fraud*

Karakteristik	Nilai
Total Data	284.807
Jumlah Fitur	30
Perbandingan Data negatif : Positif	99.83 : 0.17
Frekuensi data null pada fitur	0

3.2.4 Car Evaluation Dataset

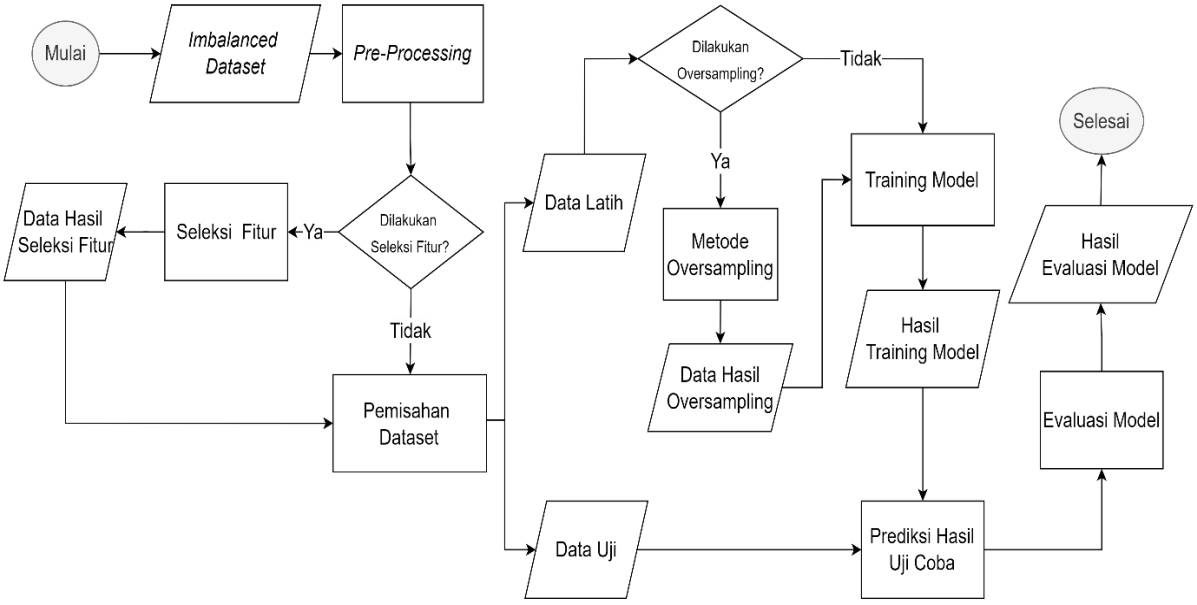
Dataset ini merupakan *multiclass* dataset yang berisi sejumlah informasi mengenai kualifikasi fisik mobil yang terdiri dari enam atribut, yakni *buying*, *maint*, *doors*, *persons*, *lug_boot*, dan *safety*, serta kelas target yang diklasfikasikan menjadi empat kelas dengan presentase jumlah datanya sebagai berikut, *unacc* 1210 data (70.023%), *acc* 384 data (22.222 %), *good* 69 data (3.993%), *v-good* 65 data (3.762%). Dengan tingginya ketimpangan jumlah kelas mayoritas dengan minoritas, maka diperlukan adanya penanganan untuk menyeimbangkan dataset agar mendapatkan model yang baik dalam membantu pengambilan keputusan. Karakteristik dari *Car Evaluation dataset* dapat dilihat pada Tabel 3.5.

Tabel 3.5 Karakteristik Dataset *Car Evaluation*

Karakteristik	Nilai
Total Data	1728
Jumlah Fitur	6
Perbandingan Data negatif : Positif	70 : 22 : 0.39 : 0.37
Frekuensi data null pada fitur	0

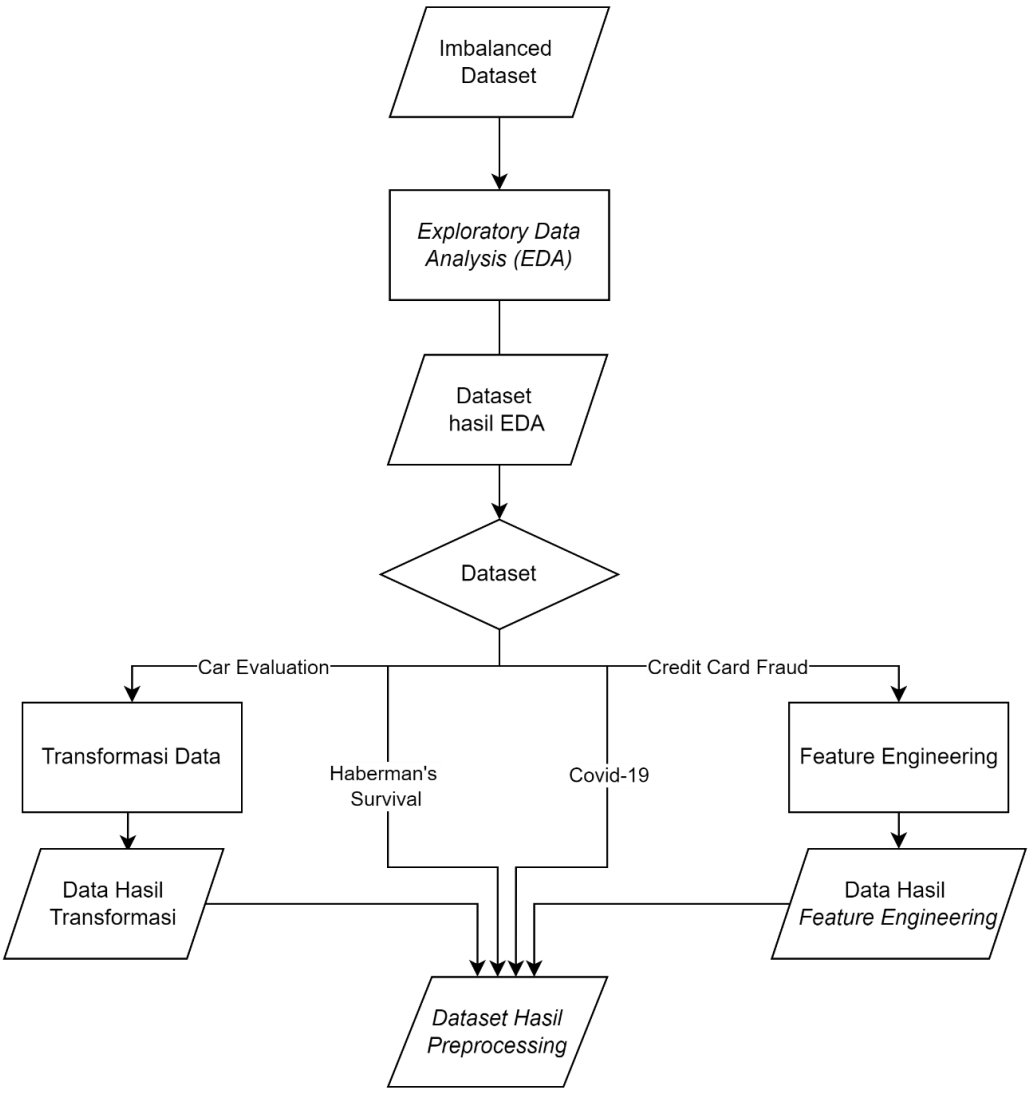
3.3 Rancangan Sistem

Pada subbab ini dijelaskan mengenai rancangan sistem yang terdiri dari diagram alir sistem yang berisi rincian dari proses pembuatan sistem. Pada diagram alir sistem, ditampilkan alur dari implementasi penelitian klasifikasi pada data tidak seimbang mulai dari tahap pengambilan dataset, tahap *preprocessing*, pembuatan model, prediksi model hingga evaluasi model yang dapat dilihat melalui Gambar 3.1. Tahapan dari implementasi penelitian ini, secara rinci dijelaskan melalui subbab ini.



Gambar 3.1 Diagram Alir Sistem

3.3.1 Pre-processing



Gambar 3.2 Tahap *preprocessing*

Berdasarkan pada Gambar 3.2, *imbalanced dataset* yang telah diinputkan kemudian dilakukan tahap *prepocessing* pertama, yakni *Exploratory Data Analysis* (EDA). EDA dilakukan untuk mengetahui karakteristik dari dataset sehingga dapat menghasilkan analisis lebih mendalam melalui visualisasi dataset yang dihasilkan. Informasi yang diperoleh dari hasil dari EDA, kemudian dapat dijadikan sebagai acuan untuk melakukan tahap *preprocessing* berikutnya. Tahap *preprocessing* berikutnya dilakukan bergantung dengan hasil EDA, di mana untuk dataset *Haberman's Survival* dan COVID-19 langsung dilakukan tahap normalisasi data. Sedangkan untuk dataset *Car Evaluation* perlu dilakukan transformasi data terlebih dahulu, karena dataset memiliki fitur dengan tipe data kategorikal, sehingga perlu diubah menjadi numerik agar lebih memudahkan proses analisis. Pada dataset *Credit Card Fraud,* dilakukan tahap *feature engineering* untuk mengekstrak informasi dari sebuah fitur agar memperoleh informasi lainnya yang diperlukan. Tahap *preprocessing* dasar berikutnya adalah *scaling* data dengan input berupa dataset hasil *preprocessing* sebelumnya. Tahap *scaling* data diperlukan agar setiap fitur pada dataset memiliki rentang nilai yang sama sehingga dapat memudahkan proses analisis berikutnya.

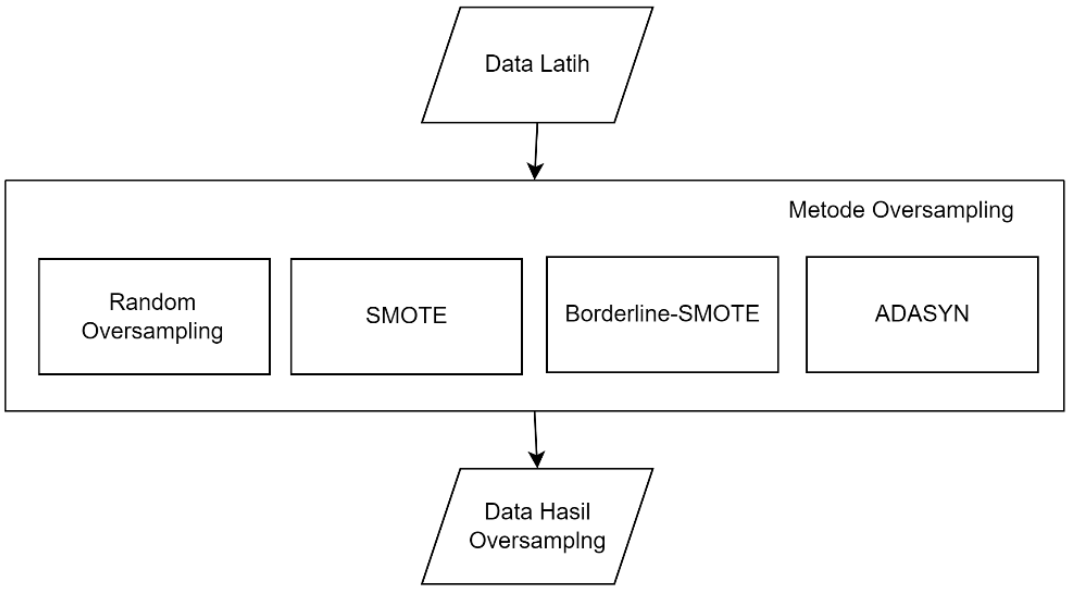
3.3.2 Seleksi Fitur

Seleksi fitur di sini dilakukan sebagai salah satu skenario pengujian. Setelah dilakukan EDA, diketahui bahwa pada dataset terdapat fitur yang kurang relevan dengan target. Untuk itu, agar lebih mempercepat proses komputasi, dilakukan seleksi bagi fitur yang kurang relevan dengan data target. Input dari tahap ini adalah dataset yang telah dilakukan tahap *preprosesing* sebelumnya. Setelah dilakukan seleksi fitur, dihasilkan dataset dengan fitur yang memiliki pengaruh tinggi terhadap target.

3.3.3 Pemisahan Dataset

Tahap *preprocessing* berikutnya adalah pemisahan dataset. Dataset yang telah diakukan tahap seleksi fitur kemudian dilakukan pemisahan menjadi data latih dan data uji dengan proporsi tertentu. Pemisahan diperlukan karena pada data latih akan dilakukan tahap *preprocessing* berupa *resampling* pada kelas minoritas dengan menggunakan beberapa metode *oversampling. Resampling* data ini diperlukan agar kuantitas datanya seimbang dengan kelas mayoritas. Sedangkan pada data uji tidak dilakukan *oversampling* karena fungsi dari data uji merupakan data yang dijadikan evaluasi model yang telah dibuat menggunakan data latih, sehingga hasil evaluasi yang diperoleh dapat merepresentasikan data yang sebenarnya.

3.3.4 Oversampling Data Latih

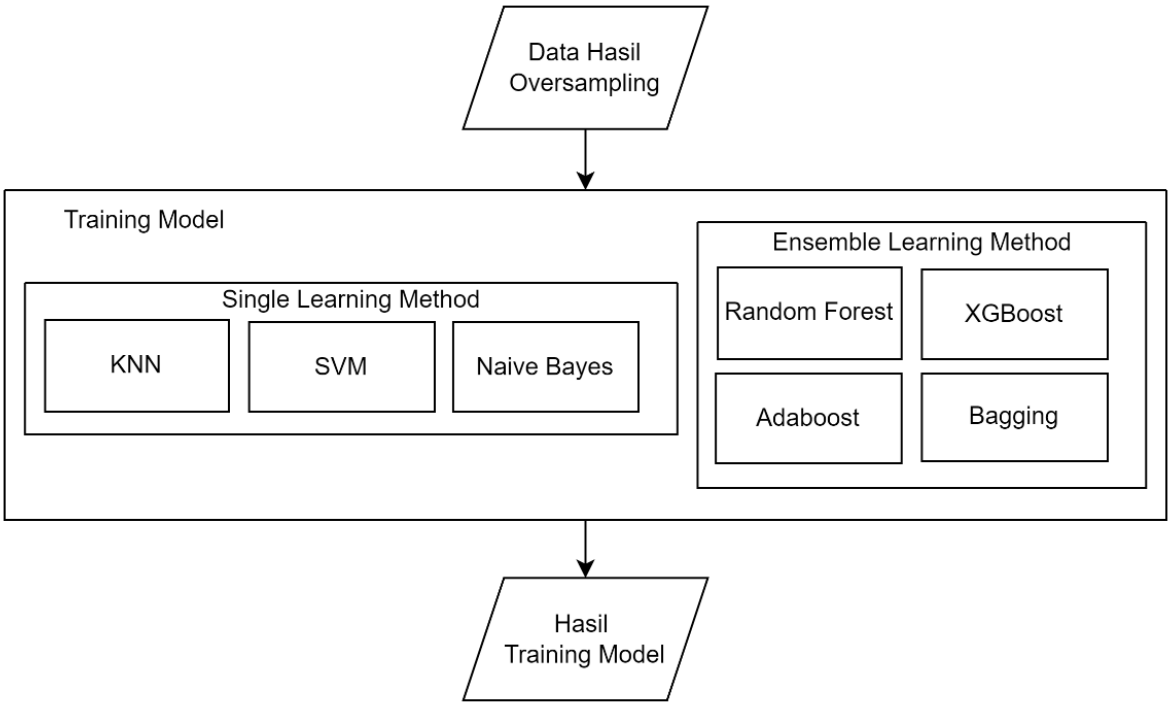


Gambar 3.3 Tahap Oversampling

Tahap selanjutnya dilakukan dua skenario, yakni data latih yang dilakukan *oversampling* dan data latih tanpa *oversampling*. Input dari tahap ini menggunakan data latih dari hasil pemisahan dataset sebelumnya. Data latih yang tidak dilakukan *oversampling*, nantinya akan dijadikan sebagai model pembanding dari data latih hasil *oversampling*. *Oversampling* data latih diperlukan agar data menjadi seimbang dan ketika dilakukan *training* model dan hasilnya tidak bias ke salah satu kelas. Setiap metode *oversampling* memiliki karakteristiknya masing-masing, sehingga data hasil dari setiap metode *oversampling* dapat berbeda satu dengan yang lainnya. Beberapa metode *oversampling* yang digunakan pada Tugas Akhir ini seperti yang terlihat pada Gambar 3.3, yakni *Random Oversampling* (ROS), SMOTE, Borderline-SMOTE, dan ADASYN. Hasil yang diperoleh pada tahap ini yakni dataset dengan persebaran kelas data yang seimbang dari hasil *oversampling.*

3.3.5 Training Model

Setelah melalui tahap *oversampling*, persebaran data antar kelas pada data latih telah seimbang dan siap untuk dilakukan *training* model. *Input* dari tahap ini adalah data latih hasil *oversampling*, dan diperoleh hasil berupa model hasil *training*. Pembuatan model di sini menggunakan beberapa metode *machine learning*, yang secara garis besar seperti yang terlihat pada Gambar 3.4, yakni *ensemble learing* dan *single learning*. Untuk *ensemble learning*, metode yang digunakan antara lain, *Random Forest*, Bagging. XGBoost, dan Adaboost. Sedangkan untuk *single learning*, antara lain KNN, SVM, dan *Naive Bayes.*



Gambar 3.4 Tahap *Training* Model

3.3.6 Prediksi Hasil Uji Coba

Prediksi hasil uji coba dilakukan dengan input berupa data uji yang dilakukan validasi data menggunakan model yang telah dihasilkan menggunakan data latih sebelumnya. *Output* yang dihasilkan pada tahap ini berupa seberapa banyak data hasil prediksi dari model yang dapat memprediksi data uji dengan benar.

3.3.7 Evaluasi Model

Setelah dilakukan prediksi hasil uji coba, maka dilakukan evaluasi model dan analisis dari hasil prediksi untuk mengetahui perbandingan kinerja dari setiap metode *oversampling* dan *ensemble learning* yang paling optimal dalam mengatasi data tidak seimbang. Evaluasi model yang digunakan di sini adalah *Confusion Matix* dengan pengukuran performa yang digunakan adalah *recall dan f1-score.*

3.4 Implementasi

3.4.1 Secara Umum

Pada subbab ini akan dijelaskan mengenai tahapan implementasi klasifikasi data tidak seimbang menggunakan metode *oversampling* secara umum.

1. *Import Library* dan Menampilkan dataset

Import library disesuaikan dengan kebutuhan masing-masing dataset. Untuk beberapa *library* dasar dan implementasi untuk menampilkan dataset dapat dilakukan pada Kode Sumber 3.1.

```
1. import pandas as pd
2. import numpy as np
3. import matplotlib.pyplot as plt
4. import seaborn as sns
5. df = pd.read_csv(dataset.csv')
6. df.head()
```

Kode Sumber 3.1 *Import Library* dan Menampilkan Dataset

Baris 1-3 pada Kode Sumber 3.1 digunakan untuk mengimpor *library* yang digunakan, di sini diimpor *library pandas*, *numpy* dan *matpolotlib*. *Pandas* tepat digunakan saat bekerja dengan data tabular, seperti data yang disimpan dalam *spreadsheet*. *Library Pandas* akan membantu dalam mengeksplorasi, membersihkan, dan memproses data tabular tersebut. Dalam *pandas*, data tabular disebut dengan *DataFrame*. *Library NumPy* adalah *library Python* yang mendukung pengolahan data pada *array* dan matriks multidimensi yang besar. *NumPy* menyediakan kumpulan fungsi matematika, seperti aljabar linear, transformasi *fourier*, pembuatan angka acak, dan lain-lain. Selanjutnya, *Matplotlib* adalah *library* Python yang mendukung pembuatan grafik dua dimensi dalam berbagai format dan dari berbagai jenis data. *Matplotlib* dapat membuat plot, histogram, spektrum, diagram batang, diagram kesalahan, plot pencar, dan lain-lain. Baris ke-5 digunakan untuk membaca dataset dengan format *csv*. Selanjutnya, baris ke-6 digunakan untuk menampilkan lima data pertama dari dataset.

2. *Preprocessing*

a. Exploratory Data Analysis (EDA)

Setiap dataset memiliki karakteristik yang berbeda-beda. Untuk mengetahui karakteristik dari setiap dataset, perlu dilakukan analisis data eksploratif (EDA). Setelah diketahui karakteristik dari masing-masing dataset, maka akan diperoleh pengetahuan baru yang digunakan untuk memproses dataset berikutnya.

```
1. df.shape
2. df.info()
3. df.describe()
4.
5. plt.subplots(figsize=’size’)
6. features_correlation=df.corr()
7. sns.heatmap(features_correlation,annot=True,cmap='RdPu')
8. plt.title('Correlation between the variables')
9. plt.xticks(rotation=45)
10.
11. df.drop('kelas_target', axis=1).corrwith(df.survival_status).plot(kind
   ='bar', grid=True, figsize=(12, 8), title="Correlation with target")
12.
13. df.target.value_counts().plot(kind='bar', title='Count (target)');
```

Kode Sumber 3. 2 *Exploratory Data Analysis*

Pada Kode Sumber 3.2 baris 1 digunakan untuk menampilkan bentuk dari dataset. Bentuk yang dimaksudkan di sini adalah dimensi dari *DataFrame* mencakup banyaknya baris data dan kolom yang dimiliki oleh dataset. Melalui implementasi tersebut, akan dihasilkan *tuple (x,y)* di mana *x* merupakan jumlah baris dan *y* merupakan jumlah kolom. Untuk baris 2 digunakan untuk

menampilkan informasi dataset berisi tentang *DataFrame* termasuk indeks, tipe data, kolom, nilai *non-null* dan memori yang digunakan. Baris ke-3 digunakan untuk mengetahui deskripsi stastistik dari dataset yang merangkum tendensi sentral, dispersi, dan bentuk distribusi kumpulan data, yang tidak termasuk nilai *NaN*. Untuk kode baris 5-9 digunakan untuk mengetahui korelasi antar fitur pada dataset, dengan dilakukan visualisasi menggunakan *heatmap* dari *library seaborn*. Implementasi baris 11 digunakan untuk menghitung korelasi berpasangan antara baris atau kolom dari dua objek *DataFrame* atau korelasi antara fitur dengan target dataset. Lalu baris 13 digunakan untuk menampilkan distribusi antar kelas. Implementasi tersebut berfungsi dengan baik apabila digunakan pada fitur dengan tipe numerik. Apabila berupa data kategorikal, maka data perlu diubah menjadi data numerik terlebih dahulu. Dengan mengetahui distribusi kelas target, akan diketahui jumlah data pada kelas target.

b. Seleksi Fitur

Pada dataset ini dilakukan dua implementasi, yakni dengan seleksi fitur dan tanpa seleksi fitur. Untuk implementasi dengan seleksi fitur, maka dilakukan penghapusan fitur yang memiliki korelasi cukup rendah dengan data target. Pada Kode Sumber 3.3 baris 1, dibuat variable baru untuk menyimpan *dataframe* dengan fitur-fitur terpilih yang memiliki korelasi tinggi dengan target.

```
1. feature_df=df[['Fitur1','Fitur2', … ]]
```

Kode Sumber 3.3 Seleksi Fitur

c. Train-test split

Tahap pra-proses berikutnya yang dilakukan adalah dengan melakukan pemisahan dataset menjadi data latih dan data uji. Baris pertama pada Kode Sumber 3.4 digunakan untuk mengimpor fungsi *train_test_split* dari *library sklearn*. Baris 2, dibuat variabel *y* untuk menyimpan data berisi kolom data target. Baris ke-3, variabel *X* digunakan untuk menyimpan data tanpa data target. Pada baris 4, dibuat variabel untuk menyimpan data hasil pemisahan dataset, yakni *X_train, y_train* untuk data latih, *X_test* dan *y_test* untuk data uji.

```
1. from sklearn.model_selection import train_test_split
2. y=df.iloc[:,-1]
3. X=df.iloc[:,:-1]
4. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2
   5, random_state=42,stratify=y)
```

Kode Sumber 3.4 *Train-test Split*

d. *Scaling* Data

```
1. from sklearn.preprocessing import StandardScaler, MinMaxScaler
2. scaler = StandardScaler()
3. X_train_scaled = scaler.fit_transform(X_train)
4. X_test_scaled = scaler.transform(X_test)
```

Kode Sumber 3.5 *Scaling* Data

Setelah dilakukan pemisahan dataset, data tersebut selanjutnya dilakukan *scaling* agar data setiap fitur memiliki rentang nilai yang sama, sehingga analisis statistik dapat dilakukan lebih mudah. Melalui Kode Sumber 3.5 baris 1, dilakukan impor fungsi *StandardScaler* dan *MinMaxScaler* dari *library sklearn*. Pada baris 2, digunakan untuk memanggil metode *scaling* yang akan digunakan dan menyimpannya pada sebuah variabel. Baris ke-3 dan ke-4 dilakukan *scaling* menggunakan *StandardScaler*, dimana *library* ini melakukan *scaling* berdasarkan standar deviasi. Pemilihan metode *scaling data* menggunakan *StandardScaler,* baik digunakan untuk dataset yang memiliki *range* data yang sangat beragam atau tidak ada batas minimal dan

maksimal tertentu, seperti data pada dataset *Haberman's Survival,* COVID-19, dan *Credit Card Fraud.* Sedangkan untuk dataset yang memiliki *range* data dengan nilai minimal dan maksimal tertentu, seperti data dengan tipe kategorikal pada dataset *Car Evaluation,* maka metode *scaling data* yang lebih baik digunakan adalah *MinMaxScaler*.

e. Resampling Data Train

```
1. from imblearn.over_sampling import SMOTE, ADASYN, BorderlineSMOTE, Rand
   omOverSampler
2. X_train_smote, y_train_smote = SMOTE(random_state=42).fit_resample(X_tr
   ain_scaled, y_train)
3. X_train_ADASYN, y_train_ADASYN = ADASYN().fit_resample(X_train_scaled,
   y_train)
4. X_train_ros, y_train_ros = RandomOverSampler().fit_resample(X_train_sca
   led, y_train)
5. X_train_borderline, y_train_borderline = BorderlineSMOTE(random_state=4
   2).fit_resample(X_train_scaled, y_train)
```

Kode Sumber 3.6 *Resampling* Data Latih

Setelah dilakukan *scaling*, langkah berikutnya adalah membuat sampel dataset menggunakan metode *resampling* dari data *train*. Untuk implementasi *resampling*, digunakan *library imblearn oversampling*. Baris pertama Kode Sumber 3.6 digunakan untuk mengimpor *library* setiap metode *oversampling* yang digunakan. Baris ke-2 hingga ke-4 dilakukan *resampling* data *train* menggunakan metode *oversampling* secara berturut-turut SMOTE, ADASYN, ROS, dan Borderline-SMOTE dengan parameter yang digunakan metode SMOTE dan Borderline-SMOTE yakni *random_state* 42.

3. Pembuatan Model

Pada Kode Sumber 3.7 dilakukan pembuatan fungsi model menggunakan beberapa metode klasifikasi, antara lain metode KNN, SVM, *Naïve Bayes, Random Forest,* Bagging, XGBoost, dan Adaboost. Di mana untuk setiap metode klasifikasi tersebut, digunakan dilakukan uji coba menggunakan beberapa *data train* yang sudah dilakukan *resample* data menggunakan metode *oversampling*.

```
1. def fit_model(metode, X_train_scaled, y_train):
2.   metode.fit(X_train_scaled, y_train)
3.   y_pred = metode.predict(X_test_scaled)
4.   pred_test = metode.predict_proba(X_test_scaled)[::,1]
5.   accuracy = metode.best_score_ *100
6.   test_accuracy=f1_score(y_test,y_pred)*100
```

Kode Sumber 3.7 Fungsi untuk *Fit Model*

Untuk implementasi metode klasifikasi, dibuat fungsi *fit_model* pada Kode Sumber 3.7. Pada baris pertama dilakukan inisiasi langsung dengan tiga parameter yang harus diinputkan, yakni metode yang digunakan, data *train* yang sudah dilakukan *scaling*, serta data target. Pada baris ke-2 dilakukan fit model sesuai dengan metode yang diinputkan pada parameter. Baris ke-3 dan ke-4 digunakan untuk memprediksi model yang dibuat berdasarkan data *test* yang sudah dilakukan *scaling*. Baris ke-5 digunakan untuk mendapatkan model dengan nilai akurasi terbaik. Baris ke-6 digunakan untuk menghitung akurasi berdasarkan metrik performansi yang digunakan. Sebelum dilakukan *fit model*, setiap metode dilakukan *grid search* terlebih dahulu guna menghasilkan model dengan parameter terbaik. Implementasi *grid search* secara umum dapat dilihat pada Kode Sumber 3.8.

Implementasi *gridsearch* untuk setiap model dilakukan dengan mengimpor fungsi_model dari *library sklearn* seperti baris ke-1 pada Kode Sumber 3.8, kemudian model dipanggil dan

disimpan pada variabel pada baris ke-2. Baris ke-3 digunakan untuk mendefinisikan parameter yang akan dilakukan *gridsearch*. Baris ke-4 dilakukan implementasi *GridsearchCV* dengan parameter berupa model yang digunakan, param_model, *cross validation*, dan *scoring*.

```
1. from sklearn.model import fungsi_model
2. model = fungsi_model()
3. param_model = {'parameter1' : (isi parameter), ... }
4. Model_Gridsearch = GridSearchCV(model, param_model, cv=n, scoring='f1',
    return_train_score=False)
```

Kode Sumber 3.8 Implementasi *GridSearchCV*

4. Evaluasi Model

Setelah dilakukan pembuatan model dan diperoleh model menggunakan parameter terbaik dari beberapa metode yang diuji cobakan, langkah berikutnya adalah mengevaluasi model tersebut menggunakan *confusion matrix*. Pada Kode Sumber 3.9, baris pertama digunakan untuk mencetak laporan klasifikasi dari model yang dihasilkan dengan memasukkan parameter data target sebenarnya dan data target hasil prediksi. Baris ke 2-4 digunakan untuk menampilkan *confusion matrix* dari hasil klasifikasi menggunakan *heatmap*.

```
1. print(classification_report(y_test, y_pred))
2. conf=confusion_matrix(y_test, y_pred)
3. sns.heatmap(conf, annot=True, xticklabels=label, yticklabels=label, cma
   p="YlGnBu", fmt = 'd')
4. plt.show()
```

Kode Sumber 3.9 Evaluasi Model

3.4.2 Hasil Implementasi *Haberman's Survival*

	patient_age	operation_year	axillary_nodes	survival_status
285	70	58	0	2
77	44	63	1	1
91	46	62	5	2
133	50	64	0	1
282	69	60	0	1

Gambar 3.5 Sampel Dataset *Haberman's Survival*

```
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   patient_age      306 non-null    int64
 1   operation_year   306 non-null    int64
 2   axillary_nodes   306 non-null    int64
 3   survival_status  306 non-null    int64
dtypes: int64(4)
memory usage: 9.7 KB
```

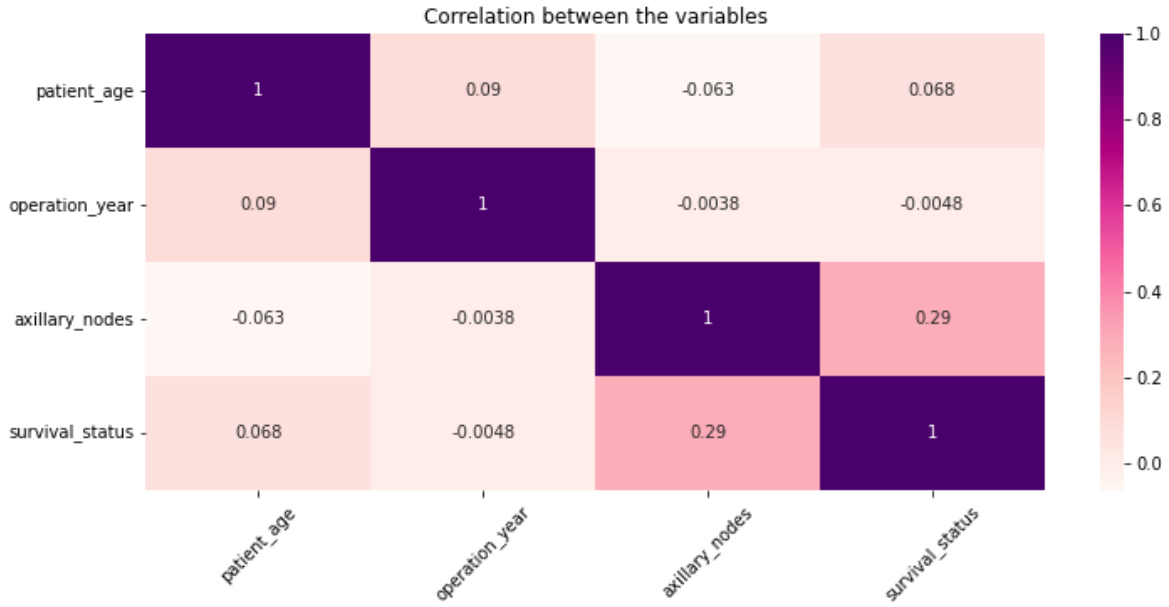
Gambar 3.6 Informasi Dataset *Haberman's Survival*

Pada Gambar 3.5 ditampilkan sampel dari dataset *Haberman's Survival* yang terdiri dari tiga fitur dan satu kelas target. Data untuk setiap fitur terdiri dari bilangan numerik dengan tipe data integer. Dengan menggunakan Kode Sumber 3.2, maka dihasilkan bentuk dari dataset *Haberman's Survival* adalah (306,4) yakni dengan 306 data dan 4 kolom. Gambar 3.6

menampilkan informasi mengenai dataset *Haberman's Survival* yang menampilkan nama setiap fitur, dengan jumlah data yang tidak *null* sebanyak 306 atau sesuai dengan keseluruhan data, yang menandakan tidak terdapat *null value* pada dataset ini. Diketahui pula tipe data dari setiap fitur ini adalah *integer*, serta memori yang digunakan oleh dataset ini yakni 9.7 KB. Pada deskripsi statistik dari dataset yang ditampilkan pada Gambar 3.7, dihasilkan jumlah data dari setiap fitur adalah sama, sehingga tidak terdapat *null value*. Sedangkan antar fitur memiliki rata-rata *range* data yang beragam, sehingga perlu dilakukan *scaling* data pada tahap berikutnya.

	patient_age	operation_year	axillary_nodes	survival_status
count	306.000000	306.000000	306.000000	306.000000
mean	52.457516	62.852941	4.026144	1.264706
std	10.803452	3.249405	7.189654	0.441899
min	30.000000	58.000000	0.000000	1.000000
25%	44.000000	60.000000	0.000000	1.000000
50%	52.000000	63.000000	1.000000	1.000000
75%	60.750000	65.750000	4.000000	2.000000
max	83.000000	69.000000	52.000000	2.000000

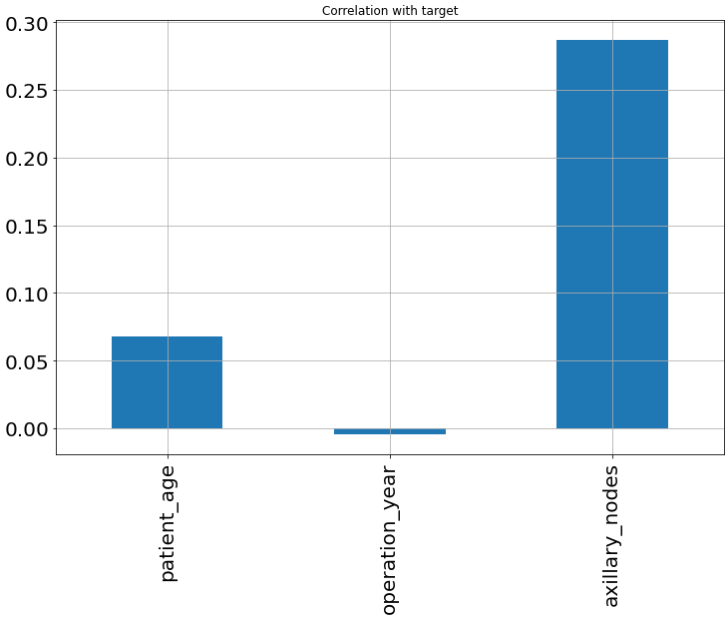
Gambar 3.7 Deskripsi Statistik Dataset *Haberman's Survival*



Gambar 3.8 Korelasi antar Fitur pada Dataset *Haberman's Survival*

Untuk korelasi antar fitur yang ditampilkan pada Gambar 3.8, warna semakin gelap menandakan antar fitur memiliki korelasi positif yang tinggi dan warna semakin terang menandakan antar fitur semakin berkorelasi negatif. Semakin tinggi korelasi antar fitur, maka salah satu fitur tersebut dapat dihilangkan karena dapat diwakilkan menggunakan salah satu fitur saja. Berdasarkan visualisasi yang dihasilkan, tidak terlihat fitur yang memiliki korelasi yang cukup tinggi.

Selain memeriksa korelasi antar fitur, dilakukan juga pemeriksaan korelasi antar fitur dengan data target seperti pada Gambar 3.9. Pengecekan korelasi fitur dengan data target diperlukan untuk mengetahui apakah fitur tersebut memiliki pengaruh yang besar pada penentuan hasil klasifikasi pada target. Dari hasil tersebut diketahui bahwa fitur *axillary_nodes* memiliki korelasi yang cukup tinggi dengan data target, sehingga merupakan salah satu fitur yang penting. Sedangkan *operation_year* memiliki korelasi paling rendah dengan data target, sehingga merupakan fitur yang memiliki pengaruh yang cukup rendah pada hasil klasifikasi. Berdasarkan hasil visualisasi korelasi fitur dengan data target tersebut, dapat dilakukan skenario uji coba dengan melakukan seleksi fitur antara penggunaan keseluruhan fitur dan dengan melakukan *drop* fitur yang memiliki korelasi cukup rendah dengan data target yakni *operation_year.*



Gambar 3.9 Korelasi antara Fitur dengan Data Target Dataset *Haberman's Survival*

Untuk mengetahui perbandingan jumlah data pada kelas target, ditampilkan pada Gambar 3.10. Diketahui bahwa pasien yang dapat bertahan hidup lebih lama memiliki jumlah data yang lebih banyak sebesar 225 data dibandingkan dengan pasien yang dapat bertahan hidup lebih singkat sebesar 81 data.

Gambar 3.10 Perbandingan Jumlah Data pada Kelas Target Dataset *Haberman's Survival*

Selanjutnya dilakukan eksplorasi mengenai persebaran data untuk setiap fitur berdasarkan *survival status*-nya. Pada Kode Sumber 3.11, baris pertama digunakan untuk mendefinisikan *style* dari visualisasi. Baris kedua digunakan untuk membuat perbandingan antar fitur menggunakan fungsi *pairplot* dari library *seaborn*. Selanjutnya, baris ke-3 digunakan untuk menampilkan hasil visualisasi.

```
1. sns.set_style('whitegrid')
2. sns.pairplot(df, hue = 'survival_status',palette='bright', size = 4)
3. plt.show()
```

Kode Sumber 3.10 Visualisasi Distribusi Dataset

Gambar 3.11 Visualisasi Distribusi Dataset *Haberman's Survival*

Gambar 3.11 menampilkan visualisasi distribusi dari dataset. Warna biru menandakan pasien yang memiliki tingkat bertahan hidup yang tinggi (*long survived)* sedangkan warna *orange* merupakan pasien yang memiliki tingkat bertahan hidup yang rendah *(short survived).* Berdasarkan hasil *plotting*, diketahui bahwa :

- Usia pasien : pasien dengan usia 30-40 tahun memiliki kemungkinan untuk *survive* yang lebih tinggi.
- Tahun operasi : pasien yang menjalani operasi pada tahun 1960-1968 memiliki kemungkinan *survived* yang lebih tinggi.

- *Axillary node* : pasien yang memiliki axillary node 0 memiliki kemungkinan *survive* yang lebih tinggi.

Setelah dilakukan proses EDA, dapat diketahui beberapa fitur yang memiliki korelasi cukup rendah dengan data target. Pada dataset ini dilakukan dua implementasi, yakni dengan seleksi fitur dan tanpa seleksi fitur. Untuk implementasi dengan seleksi fitur, maka dilakukan penghapusan fitur yang memiliki korelasi cukup rendah dengan data target, yakni *operation_year*. Selanjutnya untuk tahap pemisahan dataset digunakan *test size* sebesar 0.25 dari keseluruhan dataset, sehingga dihasilkan 229 data latih dan 77 data uji. Serta dilakukan *scaling* data, karena dataset ini memiliki range data yang cukup beragam, untuk itu metode *scaling* yang digunakan adalah *Standardscaler*. Setalah dilakukan *scaling*, tahap berikutnya dilakukan *resampling*. Untuk hasil *resampling* menggunakan data *train*, dihasilkan jumlah data seperti pada Tabel 3.6.

Tabel 3.6 Perbandingan Jumlah *Oversampling* Dataset *Haberman's Survival*

Metode *Oversampling*	**Jumlah Data Latih**	
	Long Survived	**Short Survived**
Tanpa Oversampling	168	61
SMOTE	168	168
ADASYN	168	166
ROS	168	168
Borderline-SMOTE	168	168

3.4.3 Hasil Implementasi COVID-19

Pada Gambar 3.12 menampilkan sampel dari dataset COVID-19. Dengan menggunakan Kode Sumber 3.2, dihasilkan bentuk dari dataset COVID-19 adalah (602,14) di mana dataset ini sudah pernah dilakukan praproses sebelumnya, sehingga dihasilkan 602 data dan 14 kolom.

	Patient age quantile	result	Platelets	Mean platelet volume	Red blood Cells	...	Mean corpuscular volume (MCV)	Monocytes	Red blood cell distribution width (RDW)	has_disease
0	17	0	-0.517413	0.010677	0.102004	...	0.166192	0.357547	-0.625073	1.0
1	1	0	1.429667	-1.672222	-0.850035	...	-1.336024	0.068652	-0.978899	0.0
2	9	0	-0.429480	-0.213711	-1.361315	...	1.668409	1.276759	-1.067355	1.0
3	11	0	0.072992	-0.550290	0.542763	...	0.606842	-0.220244	0.171035	1.0
4	9	0	-0.668155	1.020415	-0.127191	...	0.566783	2.012129	0.613318	0.0
...	...	...	...	...	...	...	...	...	...	...
597	19	0	-0.102873	0.908221	0.384090	...	-0.474754	1.066653	0.347948	0.0
598	19	0	0.663397	-0.774677	0.754327	...	-1.976971	1.670707	0.967144	0.0
599	15	0	-0.492289	-0.213711	0.613284	...	0.005955	0.909074	-1.155812	0.0
600	17	0	-1.773594	-0.550290	-3.318285	...	1.408024	1.381812	-0.448160	0.0
601	19	1	-0.906829	-0.325903	0.578024	...	0.025985	0.567652	-0.182790	0.0

602 rows × 14 columns

Gambar 3.12 Sampel Dataset COVID-19

Gambar 3.13 menampilkan informasi mengenai dataset COVID-19 yang berisi nama setiap fitur, jumlah data yang tidak *null*, tipe data untuk setiap fitur, serta memori yang digunakan oleh dataset ini yakni 66.0 KB. Untuk deskripsi statistik dari dataset COVID-19 ditampilkan melalui Gambar 3.14, dihasilkan nilai rata-rata antar fitur memiliki *range* yang cukup jauh, sehingga untuk pengolahan selanjutnya perlu dilakukan *scaling* agar *range* data antar fitur tidak terpaut jauh.

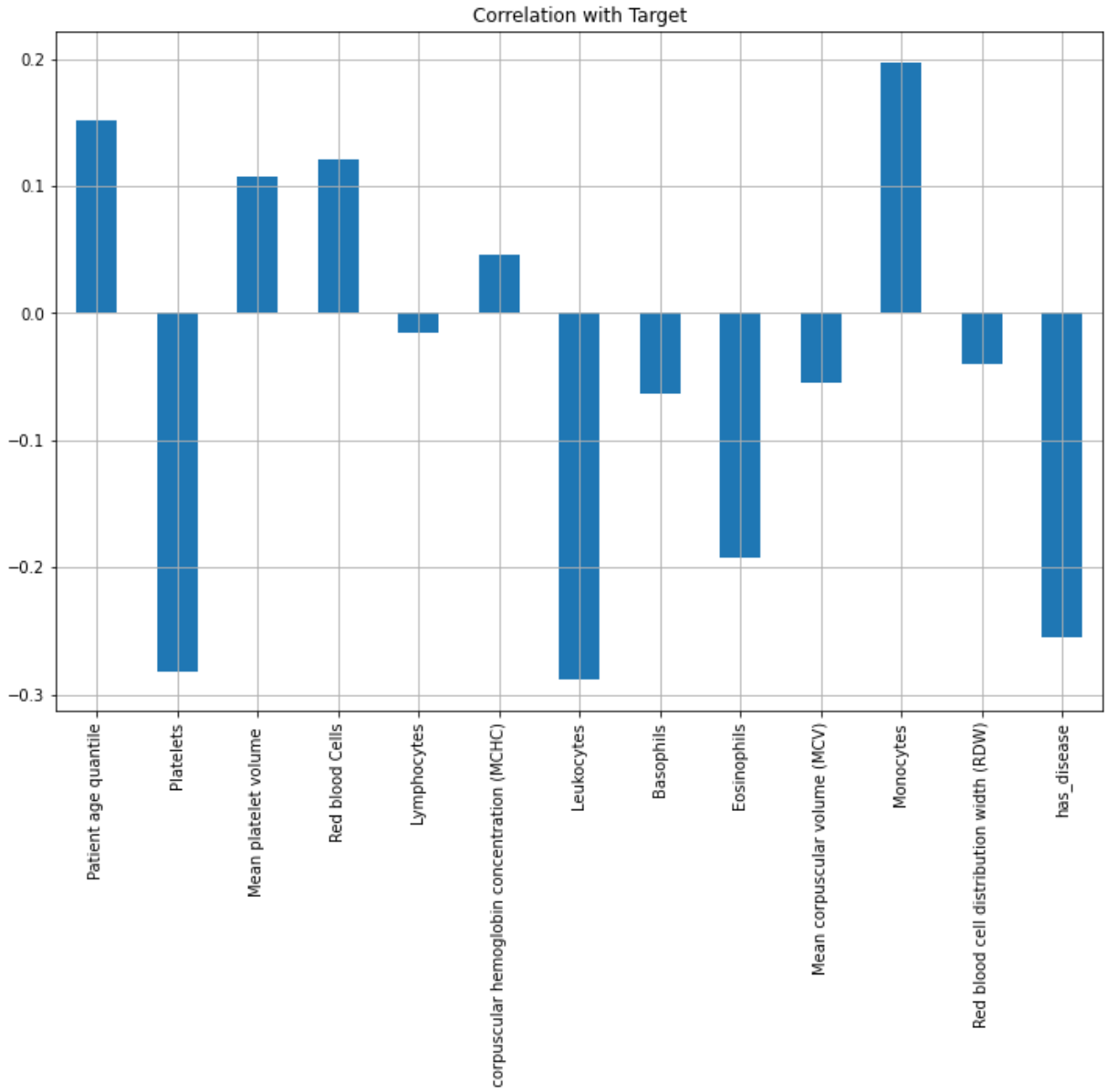
Dari hasil implementasi korelasi fitur dengan data target ditampilkan melalui Gambar 3.15, diketahui bahwa fitur *Lymphocytes*, *Mean corpuscular hemoglobin concentration (MCHC)*, *Basophils*, *Mean corpuscular volume (MCV)*, dan *Red blood cell distribution width (RDW)* memiliki korelasi dengan data target kurang dari 0.1 yang dapat dikategorikan sebagai korelasi yang rendah dan kurang berpengaruh dengan target. Gambar 3.16 ditampilkan distribusi data pada kelas target di mana kelas 0 untuk pasien dengan negatif COVID-19 sebanyak 518 dan kelas 1 untuk pasien dengan positif COVID-19 sebanyak 83, sehingga diketahui bahwa distribusi data pada kelas target tidak seimbang atau cenderung ke kelas 0 atau negatif.

```
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Patient age quantile                            602 non-null    int64
 1   result                                          602 non-null    int64
 2   Platelets                                       602 non-null    float64
 3   Mean platelet volume                            602 non-null    float64
 4   Red blood Cells                                 602 non-null    float64
 5   Lymphocytes                                     602 non-null    float64
 6   Mean corpuscular hemoglobin concentration (MCHC) 602 non-null   float64
 7   Leukocytes                                      602 non-null    float64
 8   Basophils                                       602 non-null    float64
 9   Eosinophils                                     602 non-null    float64
 10  Mean corpuscular volume (MCV)                   602 non-null    float64
 11  Monocytes                                       602 non-null    float64
 12  Red blood cell distribution width (RDW)         602 non-null    float64
 13  has_disease                                     602 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 66.0 KB
```

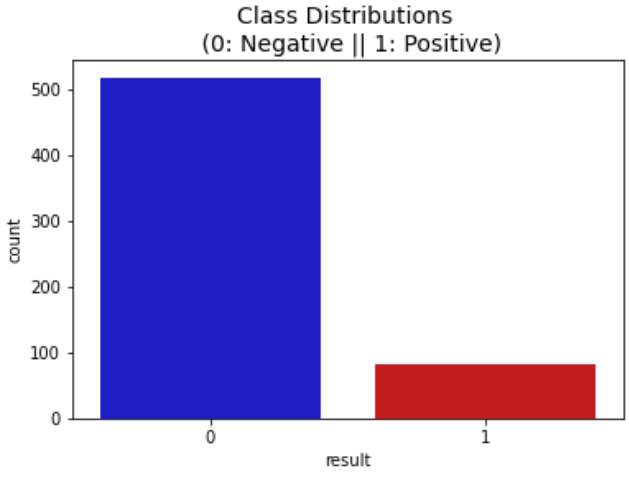
Gambar 3.13 Informasi Dataset COVID-19

	Patient age quantile	result	Platelets	Mean platelet volume	Red blood Cells	Lymphocytes
count	602.000000	602.000000	6.020000e+02	6.020000e+02	6.020000e+02	6.020000e+02
mean	11.156146	0.137874	-3.535004e-10	7.438142e-09	8.424447e-09	-7.866736e-09
std	5.710620	0.345054	1.000832e+00	9.983347e-01	1.000832e+00	1.000832e+00
min	0.000000	0.000000	-2.552426e+00	-2.457575e+00	-3.970608e+00	-1.865070e+00
25%	7.000000	0.000000	-6.053457e-01	-6.624832e-01	-5.679496e-01	-7.307069e-01
50%	12.000000	0.000000	-1.217160e-01	-1.015171e-01	1.385207e-02	-1.426696e-02
75%	16.000000	0.000000	5.314981e-01	6.838353e-01	6.661759e-01	5.976919e-01
max	19.000000	1.000000	9.532034e+00	3.713052e+00	3.645706e+00	3.764100e+00

Gambar 3.14 Deskripsi Statistik Dataset COVID-19



Gambar 3.15 Korelasi Fitur dengan Kelas Target Dataset COVID-19



Gambar 3.16 Persebaran Data pada Kelas Target Dataset COVID-19

Berdasarkan hasil EDA, diketahui beberapa fitur memiliki korelasi dengan data target yang cukup rendah sehingga perlu dilakukan penghapusan yang menghasilkan 8 fitur tersisa, yakni *Patient age quantile, result, Platelets, Mean platelet volume, Red blood Cells, Leukocytes, Eosinophils, Monocytes, has_disease.* Setelah dilakukan seleksi fitur, dilakukan pemisahan dataset dengan *test size* sebesar 0.2 dari keseluruhan dataset, sehingga dihasilkan 481 data *training* dan 121 data *testing*. Berdasarkan hasil EDA sebelumnya, diketahui bahwa data antar fitur memiliki *range* yang beragam, untuk itu perlu dilakukan *scaling*. Pada dataset ini digunakan *scaling* menggunakan *StandardScaler*. Untuk hasil *resampling* dari *data train* dari setiap metode *oversampling* dihasilkan seperti pada Tabel 3.7.

Tabel 3.7 Perbandingan Jumlah Hasil *Oversampling* Dataset COVID-19

Metode Oversampling	**Jumlah Data Latih**	
	0 (Negatif COVID)	**1 (Positif COVID)**
Tanpa Oversampling	415	66
SMOTE	415	415
ADASYN	415	403
ROS	415	415
Borderline-SMOTE	415	415

3.4.4 Hasil Implementasi *Credit Card Fraud*

Gambar 3.17 menampilkan sampel dari dataset *Credit Card Fraud*. Dengan menggunakan Kode Sumber 3.2, maka dihasilkan bentuk dari dataset *Credit Card Fraud* adalah (284807, 31) yakni dengan 284807 baris data dan 31 kolom.

	Time	V1	V2	V3	...	V26	V27	V28	Amount	result
0	0.0	-1.359807	-0.072781	2.536347	...	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	...	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	...	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	...	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	...	0.502292	0.219422	0.215153	69.99	0
5	2.0	-0.425966	0.960523	1.141109	...	0.105915	0.253844	0.081080	3.67	0
6	4.0	1.229658	0.141004	0.045371	...	-0.257237	0.034507	0.005168	4.99	0
7	7.0	-0.644269	1.417964	1.074380	...	-0.051634	-1.206921	-1.085339	40.80	0
8	7.0	-0.894286	0.286157	-0.113192	...	-0.384157	0.011747	0.142404	93.20	0
9	9.0	-0.338262	1.119593	1.044367	...	0.094199	0.246219	0.083076	3.68	0

10 rows × 31 columns

Gambar 3.17 Sampel Dataset *Credit Card Fraud*

Gambar 3.18 menampilkan informasi mengenai dataset *Credit Card Fraud* yang berisi nama setiap fitur dengan jumlah data yang tidak *null* sebanyak 284807 atau sesuai dengan keseluruhan data yang menandakan bahwa tidak terdapat *null value* pada dataset ini. Diketahui pula tipe data dari setiap fitur ini adalah *float*, serta memori yang digunakan oleh dataset ini yakni 67.4 MB. Deskripsi statistik dari dataset *Credit Card Fraud* ditampilkan pada Gambar 3.19 dengan fitur V1-V28 merupakan fitur hasil dari *Principal Component Analysis* (PCA), sehingga data yang dihasilkan tidak memiliki rentang data yang terlalu tinggi antar fitur.

Sedangkan untuk *Time*, *Amount* dan *Result* memiliki rentang yang sangat berbeda dengan yang lainnya, sehingga perlu dilakukan analisis lebih lanjut.

Fitur *Time* merupakan jumlah detik berlalu antara transaksi sekarang dengan transaksi pertama akun. Apabila hanya diketahui jumlah detik, maka kurang memperoleh gambaran kapan transaksi tersebut terjadi. Untuk itu perlu dilakukan *feture engineering* agar memperoleh informasi lebih melalui fitur *Time* tersebut. Hal tersebut dilakukan melalui implementasi Kode Sumber 3.11.

```
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  result  284807 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

Gambar 3.18 Informasi Dataset *Credit Card Fraud*

Baris pertama digunakan untuk mengimpor *library math* untuk menjalankan fungsi matematika lebih lanjut pada *pyhon*. Baris 2-3 digunakan untuk mengonversi waktu dari detik menjadi jam, hal ini diperlukan karena dengan diketahui jumlah jam, maka akan lebih mudah dipahami waktu terjadinya transaksi daripada menampilkan jumlah detik. Baris ke 4-5 digunakan untuk menghitung jumlah jam menjadi hari. Transaksi pada dataset ini dilakukan dalam dua hari, untuk itu implementasi tersebut diperlukan untuk mengetahui jam terjadinya transaksi dalam dua hari tersebut. Baris ke 6 digunakan untuk menentukan hari terjadinya transaksi dengan mengonversi jam menjadi hari terjadinya transaksi. Dataset ini diambil dalam selang waktu dua hari, sehingga akan dihasilkan kemungkinan terjadinya transaksi pada hari pertama atau kedua. Baris ke-7 digunakan untuk menampilkan hasil dari *feature engineering* yang telah dilakukan berupa kolom *time*, *day*, *hour*, *amout* dan *result* yang dihasilkan seperti pada Gambar 3.20.

	Time	V1	V2	V28	Amount	result
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	284807.000000	284807.000000
mean	94813.859575	3.918649e-15	5.682686e-16	-1.217809e-16	88.349619	0.001727
std	47488.145955	1.958696e+00	1.651309e+00	3.300833e-01	250.120109	0.041527
min	0.000000	-5.640751e+01	-7.271573e+01	-1.543008e+01	0.000000	0.000000
25%	54201.500000	-9.203734e-01	-5.985499e-01	-5.295979e-02	5.600000	0.000000
50%	84692.000000	1.810880e-02	6.548556e-02	1.124383e-02	22.000000	0.000000
75%	139320.500000	1.315642e+00	8.037239e-01	7.827995e-02	77.165000	0.000000
max	172792.000000	2.454930e+00	2.205773e+01	3.384781e+01	25691.160000	1.000000

8 rows × 31 columns

Gambar 3.19 Deskripsi Statistik Dataset *Credit Card Fraud*

```
1. import math
2. df['Time'] = df['Time'].apply(lambda sec : (sec/3600))
3. df['hour'] = df['Time']%24   # 2 days of data
4. df['hour'] = df['hour'].apply(lambda x : math.floor(x))
5. df['day'] = df['Time']/24   # 2 days of data
6. df['day'] = df['day'].apply(lambda x : 1 if(x==0) else math.ceil(x))
7. df[['Time','day','hour','Amount','result']]
```

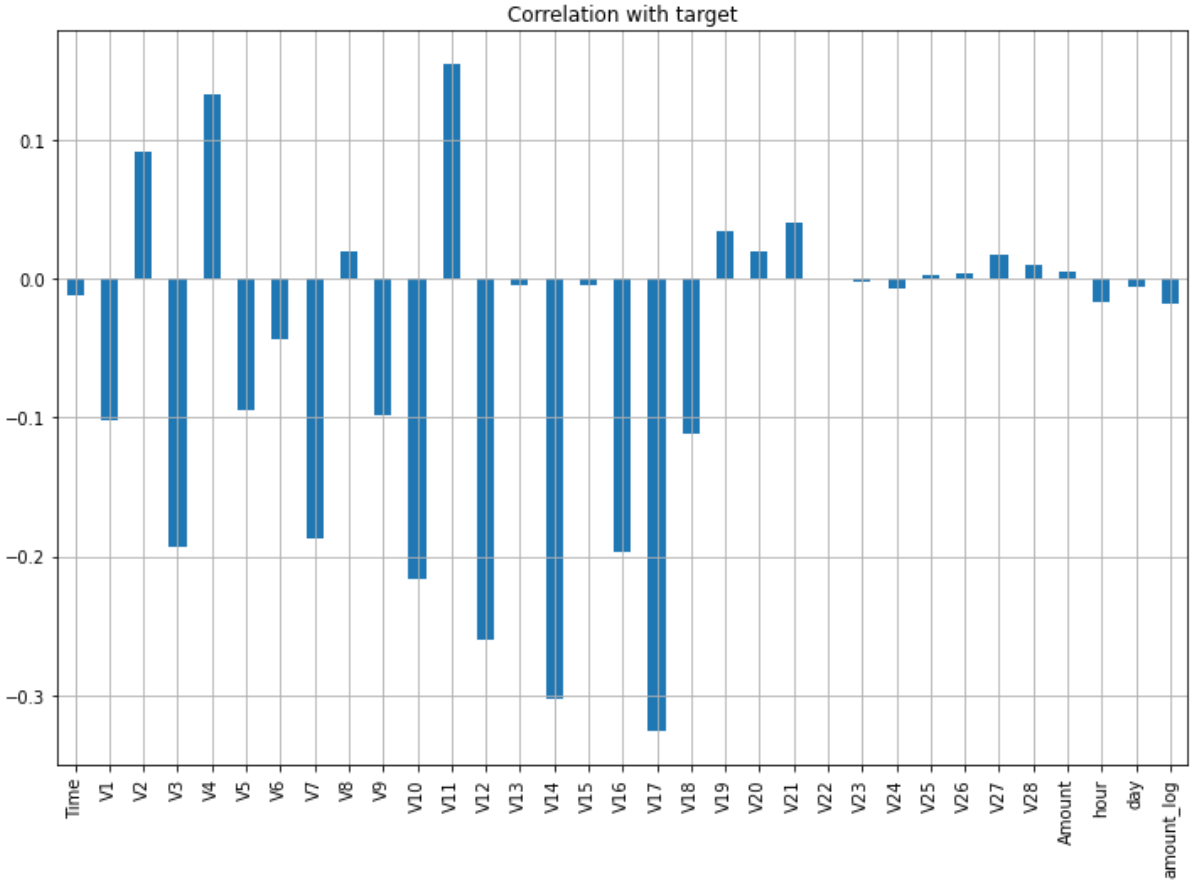
Kode Sumber 3.11 *Feature Engineering Time* Dataset *Credit Card Fraud*

	Time	day	hour	Amount	result
0	0.000000	1	0	149.62	0
1	0.000000	1	0	2.69	0
2	0.000278	1	0	378.66	0
3	0.000278	1	0	123.50	0
4	0.000556	1	0	69.99	0
...	...	...	...	...	...
284802	47.996111	2	23	0.77	0
284803	47.996389	2	23	24.79	0
284804	47.996667	2	23	67.88	0
284805	47.996667	2	23	10.00	0
284806	47.997778	2	23	217.00	0

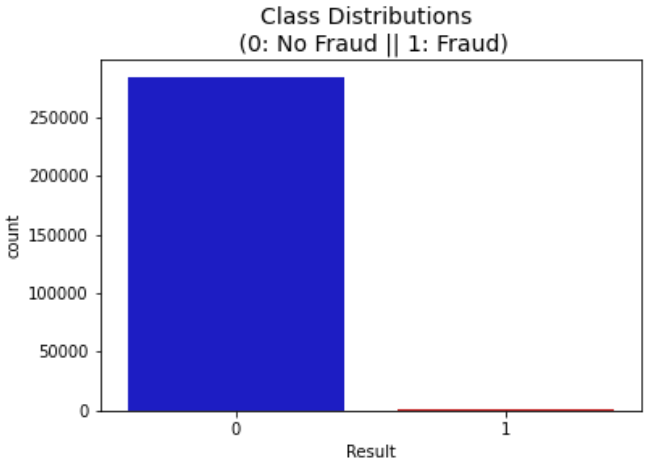
Gambar 3.20 Hasil *Feature Engineering* Dataset *Credit Card Fraud*

Setelah dilakukan *feature engineering*, dilakukan analisis mengenai korelasi antar fitur dengan target. Melalui Gambar 3.21 terlihat bahwa beberapa fitur memiliki korelasi yang sangat rendah terhadap data target. Selanjutnya dilakukan visualisasi distribusi data kelas target dapat dilihat melalui Gambar 3.22, yakni 284315 data untuk transaksi normal (*no fraud*) dan 492

untuk transaksi *fraud*, sehingga diketahui bahwa distribusi data pada kelas target tidak seimbang atau cenderung ke transaksi normal. Berdasarkan hasil korelasi fitur dengan kelas target pada Gambar 2.21, terdapat 16 fitur yang memiliki korelasi cukup rendah dengan kelas target, sehingga fitur tersebut dilakukan penghapusan hingga tersisa 15 fitur. Pada tahap pemisahan dataset, digunakan *test size* sebesar 0.25 dari keseluruhan dataset. Sehingga dihasilkan 213605 data *training* dan 71202 data *testing*. Hasil dari *resampling* diperoleh jumlah *data train* untuk setiap metode seperti pada Tabel 3.8.



Gambar 3.21 Korelasi Fitur dengan Kelas Target Dataset *Credit Card Fraud*



Gambar 3.22 Distribusi Data pada Kelas Target *Credit Card Fraud*

Tabel 3.8 Perbandingan Jumlah *Oversampling* Dataset *Credit Card Fraud*

Metode Oversampling	Jumlah Data Latih	
	0 (No Fraud)	1 (Fraud)
Tanpa Oversampling	213236	369
SMOTE	213236	213236
ADASYN	213236	213228
ROS	213236	213236
Borderline-SMOTE	213236	213236

3.4.5 Hasil Implementasi *Car Evaluation*

Gambar 3.18 menunjukkan sampel dari dataset *Car Evaluation.* Dengan menggunakan implementasi Kode Sumber 3.2, maka dihasilkan bentuk dari dataset *Car Evaluation* adalah (1728, 7) yakni dengan 1728 baris data dan 7 kolom.

	buying	maint	doors	persons	lug_boot	safety	value
247	vhigh	med	3	2	med	med	unacc
1321	low	vhigh	2	more	big	med	acc
868	med	vhigh	2	2	med	med	unacc
1486	low	high	5more	2	small	med	unacc
1556	low	med	3	4	big	high	vgood

Gambar 3.23 Sampel Dataset *Car Evaluation*

Pada Gambar 3.24 menampilkan informasi mengenai dataset *Car Evaluation* yang berisi nama setiap fitur dengan jumlah data yang tidak *null* sebanyak 1728 atau sesuai dengan keseluruhan data yang menandakan tidak terdapat *null value* pada dataset ini. Diketahui pula tipe data dari setiap fitur ini masih berupa *object* yang perlu dilakukan pengolahan lebuh lanjut, serta memori yang digunakan oleh dataset ini yakni 94.6 KB.

```
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   buying    1728 non-null   object
 1   maint     1728 non-null   object
 2   doors     1728 non-null   object
 3   persons   1728 non-null   object
 4   lug_boot  1728 non-null   object
 5   safety    1728 non-null   object
 6   value     1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

Gambar 3.24 Informasi Dataset *Car Evaluation*

```
1. col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety'
   , 'value']
2. for col in col_names:
3.     print(df1[col].value_counts())
```

Kode Sumber 3.12 Distribusi Data Setiap Fitur Dataset *Car Evaluation*

Kode Sumber 3.12 digunakan untuk mengetahui distribusi jumlah data setiap kategori pada fitur dataset. Baris pertama digunakan untuk membuat list dari nama kolom. Pada Baris ke 2-3 dibuat iterasi untuk menampilkan jumlah data dari setiap kategori pada kolom. Untuk

dataset *Car Evaluation* memiliki tujuh fitur dengan tipe kategorikal. Melalui Gambar 3.25 dapat dilihat bahwa setiap fitur memiliki 3-4 kategori data saja, dan setiap kategori memiliki jumlah data yang sama kecuali pada data target yang memiliki jumlah data paling banyak pada kategori *unacc*.

```
vhigh    432
high     432
med      432
low      432
Name: buying, dtype: int64
vhigh    432
high     432
med      432
low      432
Name: maint, dtype: int64
2        432
3        432
4        432
5more    432
Name: doors, dtype: int64
2       576
4       576
more    576
Name: persons, dtype: int64
small    576
med      576
big      576
Name: lug_boot, dtype: int64
low     576
med     576
high    576
Name: safety, dtype: int64
unacc    1210
acc       384
good       69
vgood      65
Name: value, dtype: int64
```

Gambar 3.25 Jumlah Kategori Data pada Fitur Dataset *Car Evaluation*

```
1. df = df1.copy()
2. df['buying'] = df.buying.replace({'low':1,'med': 2, 'high': 3, 'vhigh':
    4})
3. df['maint'] = df.maint.replace({'low':1,'med': 2, 'high': 3, 'vhigh': 4
   })
4. df['doors'] = df.doors.replace({'2': 1, '3': 2, '4': 3, '5more':4})
5. df['persons'] = df.persons.replace({'2': 1, '4': 2, 'more': 3})
6. df['lug_boot'] = df.lug_boot.replace({'small': 1, 'med': 2, 'big': 3})
7. df['safety'] = df.safety.replace({'low': 1, 'med': 2, 'high': 3})
8. df['value'] = df.value.replace({'unacc':1,'acc':2,'good':3,'vgood':4})
9. df.head()
```

Kode Sumber 3.13 Transformasi Data Kategorial menjadi Numerik

Karena dataset ini merupakan kategorikal, untuk itu perlu dilakukan transformasi menjadi data numerik. Transformasi dataset diperlukan untuk mengubah tipe dataset agar lebih mudah

dalam melakukan pengolahan data. Melalui Kode Sumber 3.13, baris pertama dibuat salinan dari dataframe sebelumnya. Baris 2-8 digunakan untuk mengubah setiap kategori pada fitur dataset menjadi data numerik menggunakan fungsi replace. Baris 9 digunakan untuk menampilkan data hasil encoding. Hasil dari transformasi dataset ditampilkan pada Gambar 3.26.

	buying	maint	doors	persons	lug_boot	safety	value
0	4	4	1	1	1	1	1
1	4	4	1	1	1	2	1
2	4	4	1	1	1	3	1
3	4	4	1	1	2	1	1
4	4	4	1	1	2	2	1

Gambar 3.26 Hasil Transformasi Data Kategorikal menjadi Numerik

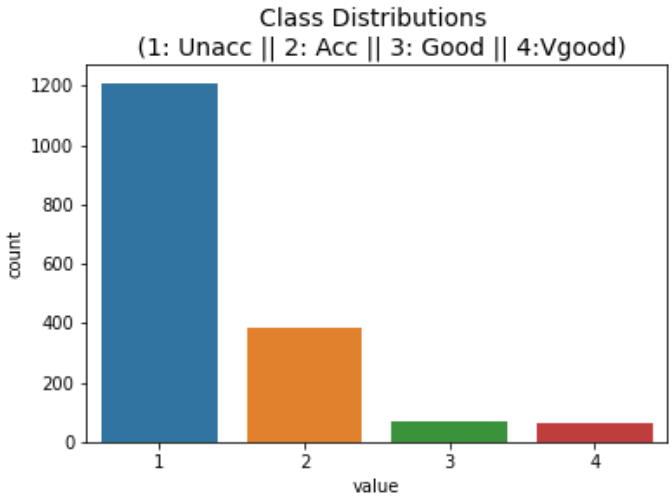
Setelah dilakukan transformasi, selanjutnya dapat dilakukan beberapa eksploarasi dataset lebih lanjut, seperti implementasi deskripsi statistik dari dataset *Car Evaluation* yang dihasilkan seperti pada Gambar 3.27.

	buying	maint	doors	persons	lug_boot	safety	value
count	1728.000000	1728.000000	1728.000000	1728.000000	1728.000000	1728.000000	1728.000000
mean	2.500000	2.500000	2.500000	2.000000	2.000000	2.000000	1.414931
std	1.118358	1.118358	1.118358	0.816733	0.816733	0.816733	0.740700
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	1.750000	1.750000	1.750000	1.000000	1.000000	1.000000	1.000000
50%	2.500000	2.500000	2.500000	2.000000	2.000000	2.000000	1.000000
75%	3.250000	3.250000	3.250000	3.000000	3.000000	3.000000	2.000000
max	4.000000	4.000000	4.000000	3.000000	3.000000	3.000000	4.000000

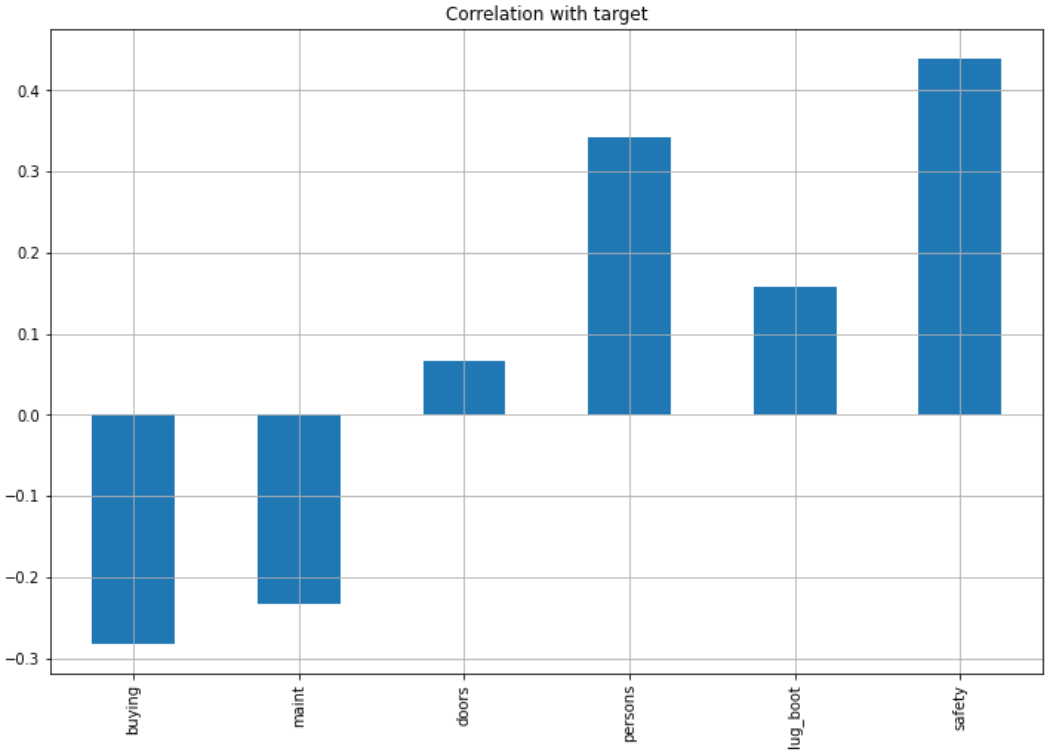
Gambar 3.27 Deskripsi Dataset *Car Evaluation*

Serta dilakukan visualisasi distribusi data pada kelas target yang dihasikan seperti pada Gambar 3.29, di mana diketahui nilai 1 (*unacc*) sebesar 1210, 2(*acc)* sebanyak 384, 3(*good*) sebanyak 69, 4(*vgood*) sebanyak 65. Melalui hasil distribusi tersebut diketahui bahwa dataset ini memiliki ketimpangan jumlah kelas atau *imbalanced dataset.*

Selain itu, dilakukan analisis pada korelasi fitur dengan kelas target. Pada Gambar 3.28 diketahui bahwa terdapat fitur yang mempunyai korelasi rendah dengan kelas target, yakni *doors* kurang dari 0.1. Untuk fitur yang memiliki korelasi dengan data target yang rendah, maka fitur tersebut dilakukan penghapusan pada tahap seleksi fitur. Untuk tahap selanjutnya yakni pemisahan dataset dengan menggunakan *test size* sebesar 0.25 dari keseluruhan dataset, sehingga dihasilkan 1296 *data train* dan 432 *data test.* Pada dataset ini digunakan *scaling* berupa *MinMaxScaler*. Lalu dengan menggunakan metode *oversampling*, dihasilkan jumlah *data train* masing-masing metode yang dihasilkan menjadi seperti Tabel 3.9.



Gambar 3.28 Distribusi Data dengan Kelas Target *Car Evaluation*



Gambar 3.29 Korelasi Fitur dengan Kelas Target Dataset *Car Evaluation*

Tabel 3.9 Perbandingan Jumlah *Oversampling* Dataset *Car Evaluation*

Metode Oversampling	**Jumlah Data Latih**			
	Unacc	**Acc**	**Good**	**Vgood**
Tanpa *Oversampling*	907	288	52	49
SMOTE	907	907	907	907
ADASYN	907	903	900	900
ROS	907	907	907	907
Borderline-SMOTE	907	907	907	907

BAB 4 Hasil Uji Coba dan Evaluasi

Bab ini akan membahas mengenai hasil uji coba sistem yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

4.1 Lingkungan Pengujian

Lingkungan uji coba pada Tugas Akhir ini adalah menggunakan sebuah notebook Asus A412FL. Sistem operasi yang digunakan adalah Windows 10 Home 64-bit. Notebook yang digunakan memiliki spesifikasi Intel® Core™ i5-8265U dengan kecepatan hingga 3.9 GHz, *Random Access Memory* (RAM) sebesar 8 GB, dan mempunyai *Video Graphics Adapter (VGA)* yaitu NVIDIA GeForce MX250. Pada sisi perangkat lunak, uji coba pada Tugas Akhir ini dilakukan dengan menggunakan bahasa pemograman Python 3.6, dilengkapi dengan *library* antara lain *Pandas, NumPy, Scikit-learn, SciPy, Imbalanced-learn*, dan *Matplotlib*.

4.2 Skenario Uji Coba

Skenario uji coba pada penelitian ini dilakukan untuk setiap dataset yang digunakan. Uji coba tersebut antara lain :

1) Uji Coba Penggunaan Seleksi Fitur
   Pada uji coba ini dilakukan perbandingan performa antara penggunaan seleksi fitur dan tanpa seleksi fitur pada dataset. Uji coba ini bertujuan untuk mengetahui pengaruh dari penggunaan fitur terhadap performa yang dihasilkan. Hal ini karena tidak semua fitur yang dimiliki setiap dataset merupakan fitur yang penting. Untuk itu, penggunaan seleksi fitur ini akan mengeliminasi fitur yang kurang berpengaruh terhadap data target.
2) Uji Coba Penggunaan Metode *Oversampling*
   Pada uji coba ini dilakukan perbandingan performa penggunaan setiap metode *oversampling* dan tanpa *oversampling*. Uji coba ini bertujuan untuk mengetahui pengaruh dari penggunaan *oversampling* terhadap performa klasifikasi dalam mengatasi data tidak seimbang. Selain itu, dilakukan pula perbandingan metode *oversampling* yang paling optimal untuk setiap dataset yang diujikan.
3) Uji Coba Penggunaan Metode Klasifikasi
   Pada uji coba ini dilakukan perbandingan performa dari setiap metode klasifikasi yang digunakan. Uji coba ini bertujuan untuk mengetahui metode klasifikasi terbaik yang digunakan untuk mengatasi data tidak seimbang pada setiap dataset dengan menggunakan metode pengukuran performa tertentu.
   Berikut adalah hasil uji coba yang dilakukan untuk setiap dataset :

4.2.1 Uji Coba Dataset *Haberman's Survival*

Pada studi kasus ini, model lebih difokuskan untuk mendeteksi pasien yang memiliki status bertahan hidup kurang dari 5 tahun. Di mana lebih baik pasien yang memiliki status bertahan hidup lebih dari 5 tahun terprediksi menjadi pasien yang dapat bertahan hidup kurang dari 5 tahun, daripada pasien yang sebenarnya memiliki status bertahan hidup kurang dari 5 tahun tetapi terprediksi menjadi pasien yang dapat bertahan hidup lebih dari 5 tahun. Untuk itu, pengukuran performa yang cocok digunakan adalah *recall.* Hal ini dikarenakan *recall* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. *Recall* menjawab persentase hasil prediksi pasien yang memiliki status bertahan hidup kurang dari 5 tahun dibandingkan dengan keseluruhan pasien yang memiliki status bertahan hidup kurang dari 5 tahun yang sebenarnya. Pada dataset *Haberman's Survival*, dilakukan beberapa skenario yang dijelaskan pada subbab berikut.

1. Uji Coba Penggunaan Seleksi Fitur

a. Tanpa Seleksi Fitur

Uji coba pertama dilakukan menggunakan keseluruhan fitur yang tersedia, yakni *patient_age, operation_ye, axillary_nodes*. Dari dataset ini kemudian dilakukan pengujian menggunakan beberapa metode *oversampling* dan metode klasifikasi yakni *ensemble learning* dan *single learning* dengan hasil uji coba yang dapat dilihat pada Tabel 4.1.

Tabel 4.1 Hasil Uji Dataset *Haberman's Survival* Tanpa Seleksi Fitur

Model	Recall					
	Tanpa Oversampling	ADASYN	Borderline	ROS	SMOTE	Rata-rata Oversampling
KNN	0.7403	0.5844	0.6494	0.6234	0.6234	0.6202
Naive Bayes	0.7143	0.7013	0.7013	0.7013	0.7013	0.7013
SVM	0.7013	0.6494	0.6104	0.6364	0.7143	0.6526
Adaboost	0.7013	0.6104	0.5844	0.5844	0.5844	0.5909
Bagging	0.6364	0.6234	0.6234	0.6364	0.6104	0.6234
Random Forest	0.6623	0.6364	0.6753	0.6364	0.6494	0.6494
XGboost	0.7013	0.6364	0.6623	0.6494	0.6364	0.6461
Rata-rata Klasifikasi	**0.6939**	**0.6345**	**0.6438**	**0.6382**	**0.6457**	

Dari hasil uji coba tanpa menggunakan seleksi fitur, dihasilkan metode yang memiliki performa terbaik adalah KNN tanpa dilakukan *oversampling* dengan nilai *recall* sebesar 0.7403. Berdasarkan Tabel 4.1, dapat terlihat bahwa penggunaan metode *oversampling* tidak memiliki pengaruh yang signifikan pada hasil klasifikasi, bahkan cenderung menurun daripada sebelum dilakukan *oversampling*.

b. Menggunakan Seleksi Fitur

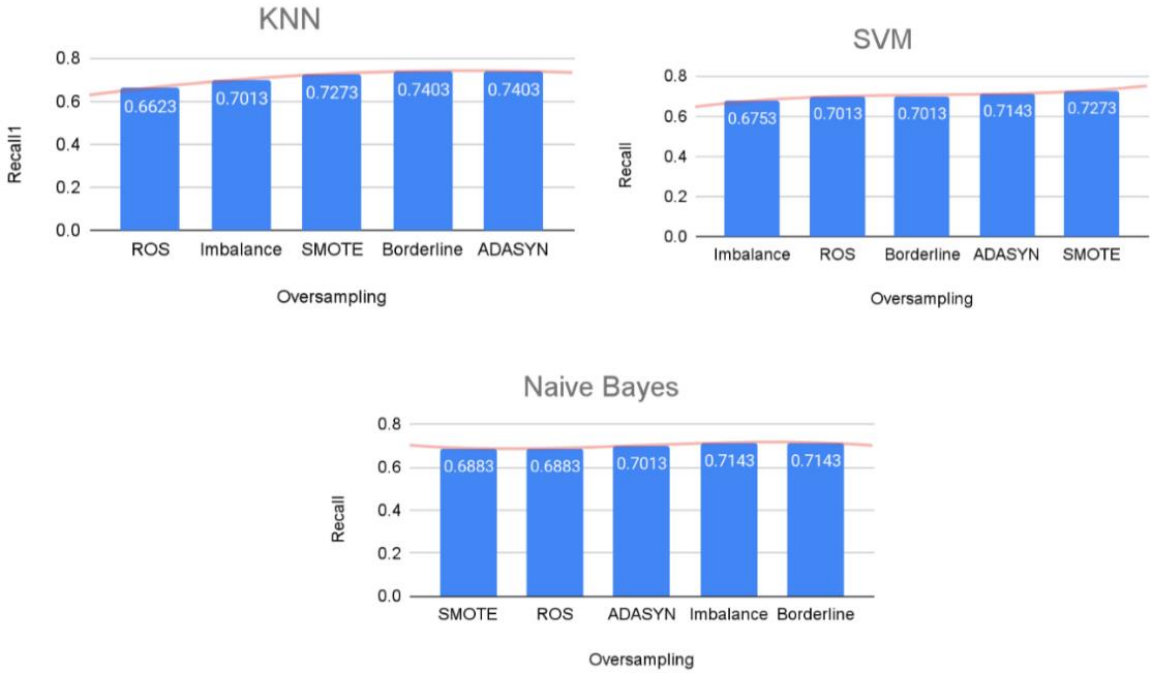
Setelah dilakukan analisis data eksplorasi pada tahap sebelumnya, diketahui bahwa terdapat fitur yang memiliki korelasi rendah dengan data target. Untuk itu, dilakukan pengujian menggunakan dataset yang telah dilakukan seleksi fitur. Di sini digunakan 2 dari 3 fitur yang tersedia, yakni *patient_age* dan*'axillary_nodes* dengan hasil pengujian yang dapat dilihat pada Tabel 4.2.

Tabel 4.2 Hasil Uji Dataset *Haberman's Survival* dengan Seleksi Fitur

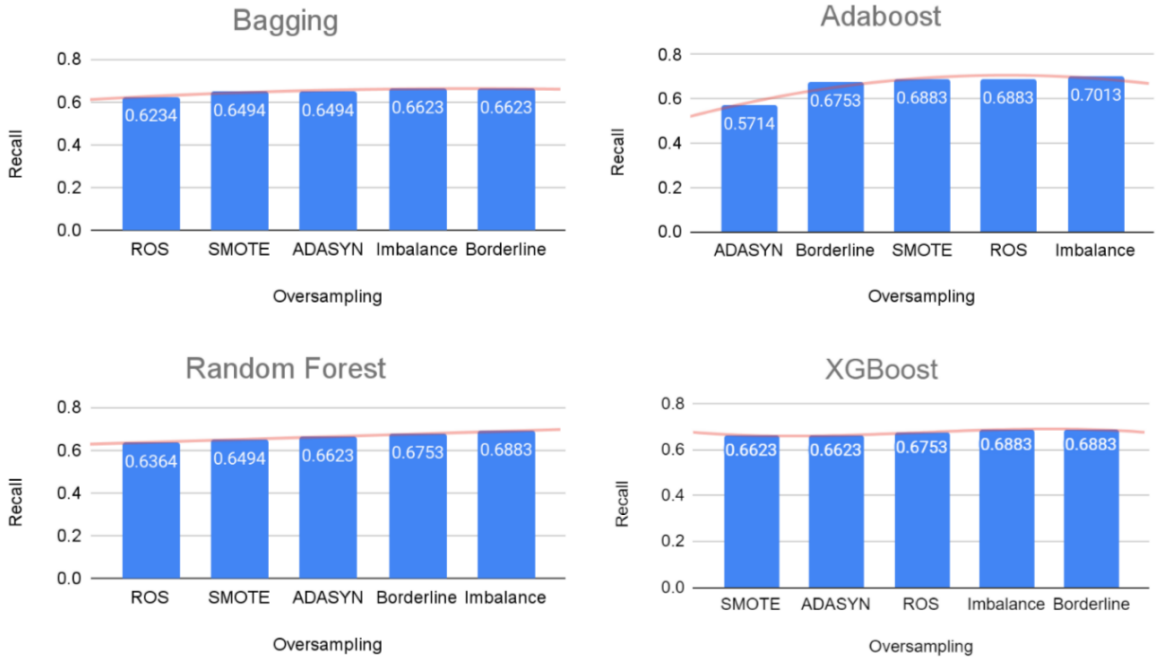
Model	Recall					
	Tanpa Oversampling	ADASYN	Borderline	ROS	SMOTE	Rata-rata Oversampling
KNN	0.7013	0.7403	0.7403	0.6623	0.7273	0.7370
Naive Bayes	0.7143	0.7013	0.7143	0.6883	0.6883	0.7013
SVM	0.6753	0.7143	0.7013	0.7013	0.7273	0.7078
Adaboost	0.7013	0.5714	0.6753	0.6883	0.6883	0.6364
Bagging	0.6623	0.6494	0.6623	0.6234	0.6494	0.6526
Random Forest	0.6883	0.6623	0.6753	0.6364	0.6494	0.6623
XGboost	0.6883	0.6623	0.6883	0.6753	0.6623	0.6688
Rata-rata Klasifikasi	**0.6902**	**0.6772**	**0.6883**	**0.6716**	**0.6865**	

Setelah dilakukan seleksi fitur, terjadi peningkatan performa pada hasil klasifikasi. Nilai *recall* tertinggi yang diperoleh adalah 0.7403 menggunakan metode KNN dengan *oversampling*

ADASYN dan *Borderline*-SMOTE. Melalui seleksi fitur, terlihat pula pengaruh dari penggunaan metode *oversampling* dalam mengatasi data tidak seimbang, di mana terjadi kenaikan nilai *recall* dibandingkan tanpa menggunakan metode *oversampling*.



Gambar 4.1 Haberman's - Grafik Nilai *Recall* pada Klasifier *Single Learning*

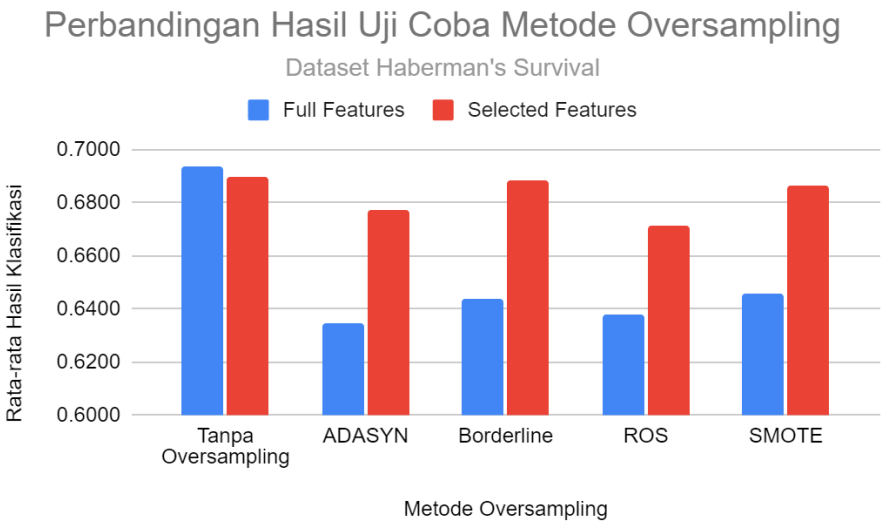


Gambar 4.2 Haberman's - Grafik Nilai *Recall* pada Klasifier *Ensemble Learning*

Pada *single learning*, nilai *recall* tertinggi berikutnya diperoleh dengan metode SVM dan *oversampling* SMOTE yang menghasilkan nilai *recall* sebesar 0.7273. Kemudian *Naïve Bayes* dengan *oversampling Borderline*-SMOTE yang menghasilkan nilai *recall* 0.7143. Dari ketiga *classifier* tersebut, nilai *recall* yang dihasilkan cenderung mengalami kenaikan daripada sebelum dilakukan *oversampling*. Grafik visualisasi dari peningkatan nilai *recall* untuk setiap metode klasifikasi *single learning* dapat dilihat pada Gambar 4.2.

Untuk metode *ensemble learning*, nilai *recall* tertinggi diperoleh tanpa menggunakan metode *oversampling* dengan metode klasifikasi Adaboost yakni sebesar 0.7013. Disusul oleh XGBoost yang menghasilkan nilai *recall* sebesar 0.6883 dengan metode *oversampling Borderline*-SMOTE. Nilai *recall* yang sama juga diperoleh menggunakan metode *Random forest* tanpa *oversampling* sebesar 0.6883. Metode *ensemble* terakhir yakni Bagging dengan *oversampling borderline*-SMOTE yang menghasilkan nilai *recall* sebesar 0.6623. Grafik visualisasi dari peningkatan nilai *recall* untuk setiap metode klasifikasi *ensemble learning* dapat dilihat pada Gambar 4.2.

2. Uji Coba Penggunaan Metode *Oversampling*



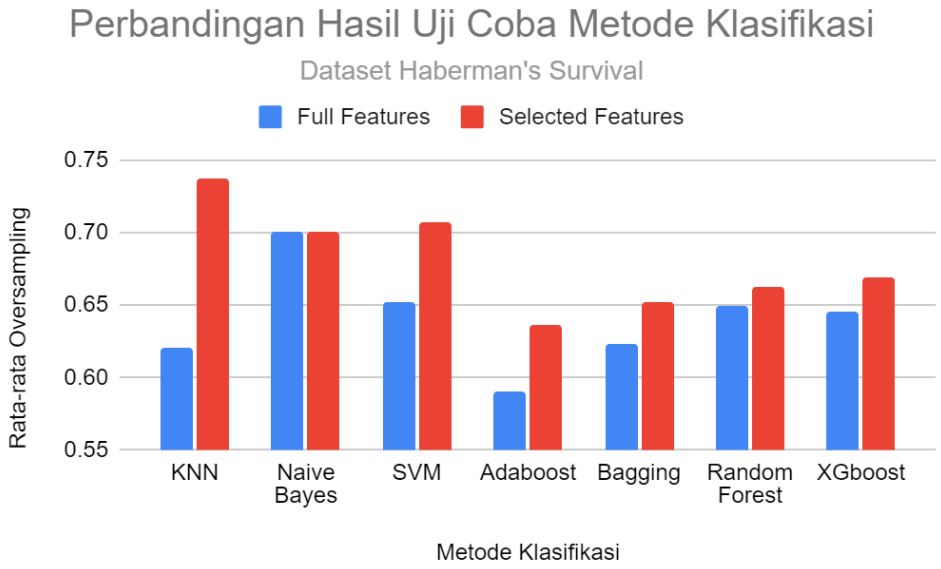
Gambar 4.3 Perbandingan Hasil Metode *Oversampling* Dataset *Haberman's Survival*

Untuk perbandingan penggunaan metode *oversampling*, penerapan seleksi fitur dapat meningkatkan hasil klasifikasi daripada tanpa seleksi fitur. Grafik visualisasi perbandingan nilai rata-rata hasil klasifikasi setiap metode *oversampling* dapat dilihat pada Gambar 4.3. Dari keseluruhan metode yang telah diuji cobakan, terlihat performa dari metode *oversampling* yang digunakan. Metode *oversampling* yang memiliki rata-rata nilai *recall* terbaik secara berurutan adalah *Borderline*-SMOTE, SMOTE, ADASYN, dan ROS.

Pada dataset ini, dihasilkan bahwa metode *oversampling* kurang memberikan pengaruh yang cukup signifikan terhadap hasil klasifikasi, baik dengan menggunakan seleksi fitur maupun tanpa seleksi fitur, bahkan hasil klasifikasi yang diperoleh cenderung menurun daripada tanpa dilakukan *oversampling*. Beberapa hal yang dapat mempengaruhi dari hasil klasifikasi tersebut, diantaranya adalah jumlah data dari dataset, persebaran dataset, serta perbandingan jumlah data pada kelas target yang tidak seimbang. Pada dataset *Haberman's Survival* ini, jumlah data yang dimiliki hanya 306 data dengan data latih sebanyak 229 data dan data uji sebanyak 77 data. Semakin sedikit data, maka semakin sedikit pula informasi yang dimiliki untuk melatih data dalam pembuatan model. Selain jumlah data yang sedikit, persebaran data juga dapat mempengaruhi hasil klasifikasi. Berdasarkan visualisasi distribusi dataset pada Gambar 3.8, terlihat persebaran kelas data yang saling tumpang tindih, sehingga klasifier tidak dapat mengklasifikasikan data dengan baik. Persebaran dataset yang kurang baik juga dapat mempengaruhi *resampling* data yang dihasilkan pada tahap *oversampling* data latih. Data *resampling* yang kurang baik akan membuat performa model dari data hasil *oversampling* tidak terlihat begitu signifikan dibandingkan tanpa menggunakan *oversampling*.

3. Uji Coba Penggunaan Metode Klasifikasi

Penggunaan seleksi fitur memiliki pengaruh lebih baik terhadap klasifikasi yang dihasilkan. Pada dataset *Haberman's Survival*, model tanpa seleksi fitur menghasilkan rata-rata nilai *recall* sebesar 0.6370 sedangkan model dengan seleksi fitur menghasilkan rata-rata nilai *recall* sebesar 0.6610, lebih tinggi daripada tanpa dilakukan seleksi fitur. Perbandingan rata-rata nilai *recall* yang dihasilkan setiap model dapat dilihat pada Gambar 4.4. Untuk metode klasifikasi *single learning*, rata-rata penggunaan metode *oversampling* terbaik adalah metode KNN, SVM, dan *Naïve Bayes*. Lalu untuk metode *ensemble learning*, nilai rata-rata *recall* terbaik adalah XGBoost, *Random Forest*, Bagging, dan Adaboost.



Gambar 4.4 Perbandingan Hasil Metode Klasifikasi Dataset *Haberman's Survival*

4.2.2 Uji Coba Dataset COVID-19

Pada kasus ini, model lebih difokuskan untuk mendeteksi pasien yang mengalami positif COVID-19 dibandingkan pasien yang negatif COVID-19, atau lebih baik memprediksi pasien sebagai positif COVID-19 walaupun sebenarnya negatif COVID-19 daripada memprediksi pasien menjadi negatif COVID-19 tetapi sebenarnya positif COVID-19. Untuk itu, pengukuran performa yang cocok digunakan adalah *recall*. Hal ini dikarenakan *recall* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. *Recall* menjawab persentase prediksi pasien positif COVID-19 dibandingkan keseluruhan pasien positif COVID-19 yang sebenarnya. Pada dataset COVID-19 ini, dilakukan beberapa skenario yang dapat dilihat pada subbab berikut.

1. Uji Coba Penggunaan Seleksi Fitur

a. Tanpa Seleksi Fitur

Uji coba pertama dilakukan menggunakan keseluruhan fitur yang tersedia, yakni 13 fitur dan 1 kelas target. Pada uji coba ini dilakukan skenario dengan menggunakan beberapa metode klasifikasi *ensemble learning* dan *single learning*, serta penggunaan metode *oversampling* pada data latih. Hasil dari uji coba ini dapat dilihat pada Tabel 4.3.

Tabel 4.3 Hasil Uji Dataset COVID-19 Tanpa Seleksi Fitur

Model	Recall

	Tanpa Oversampling	ADASYN	Borderline	ROS	SMOTE	Rata-rata Oversampling
KNN	0.6963	0.8071	0.8264	0.8258	0.8012	0.8151
Naive Bayes	0.7952	0.8167	0.8264	0.8066	0.7766	0.8066
SVM	0.5000	0.8295	0.8835	0.8001	0.8247	0.8345
Adaboost	0.6818	0.7118	0.7412	0.7557	0.7610	0.7424
Bagging	0.6867	0.8097	0.7557	0.7359	0.8343	0.7839
Random Forest	0.5294	0.7797	0.7407	0.6620	0.7407	0.7308
XGboost	0.6963	0.7803	0.7803	0.7605	0.8145	0.7839
Rata-rata Klasifikasi	**0.6551**	**0.7907**	**0.7935**	**0.7638**	**0.7933**	

Dari hasil uji coba tanpa seleksi fitur, metode yang memiliki performa terbaik adalah *single learning* dengan klasifier SVM dan metode *oversampling Borderline*-SMOTE dengan nilai *recall* sebesar 0.8835. Sedangkan untuk *ensemble learning,* nilai *recall* tertinggi diperoleh menggunakan Bagging dan SMOTE dengan nilai *recall* sebesar 0.8343. Berdasarkan Tabel 4.3, seluruh model menghasilkan nilai *recall* yang lebih tinggi setelah dilakukan metode *oversampling*.

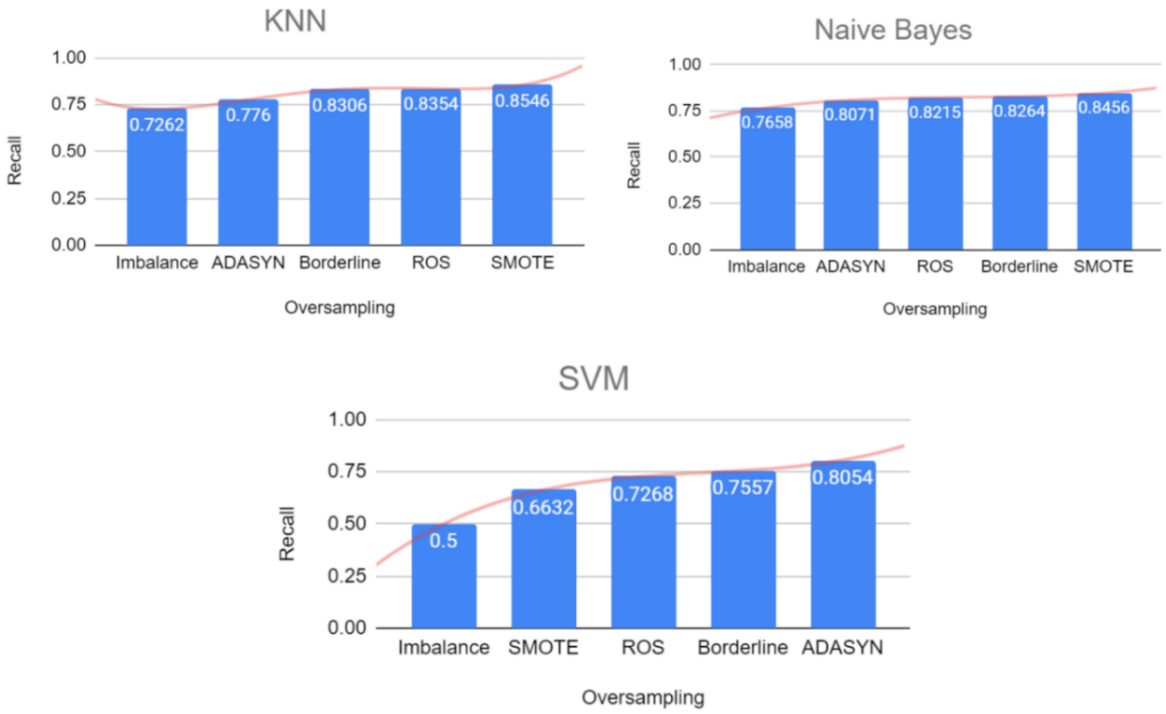
b. Menggunakan Seleksi Fitur

Pada uji coba ini dilakukan seleksi fitur yang memiliki korelasi cukup rendah terhadap target berdasarkan hasil analisis yang dilakukan sebelumnya. Terdapat 8 fitur yang digunakan disini memiliki korelasi lebih dari 0.1 dari target, fitur tersebut antara lain *Patient age quantile, Platelets, Mean platelet volume, Red blood Cells, Leukocytes, Eosinophils, Monocytes, dan has_disease* Hasil dari uji coba menggunakan seleksi fitur dapat dilihat pada Tabel 4.4.

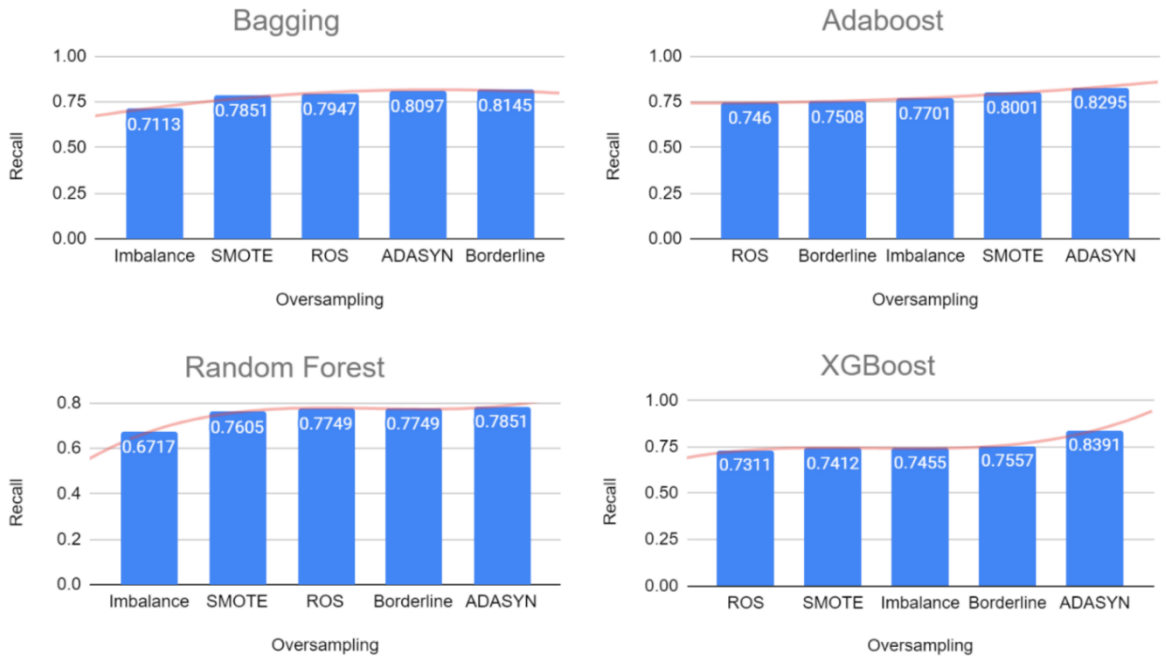
Tabel 4.4 Hasil Uji Dataset COVID-19 dengan Seleksi Fitur

Model	Recall					
	Tanpa Oversampling	ADASYN	Borderline	ROS	SMOTE	Rata-rata Oversampling
KNN	0.7262	0.7760	0.8306	0.8354	0.8546	0.8242
Naive Bayes	0.7658	0.8071	0.8264	0.8215	0.8456	0.8252
SVM	0.5000	0.8054	0.7557	0.7268	0.6632	0.7378
Adaboost	0.7701	0.8295	0.7508	0.7460	0.8001	0.7816
Bagging	0.7113	0.8097	0.8145	0.7947	0.7851	0.8010
Random Forest	0.6717	0.7851	0.7749	0.7749	0.7605	0.7739
XGboost	0.7455	0.8391	0.7557	0.7311	0.7412	0.7668
Rata-rata Klasifikasi	**0.6987**	**0.8074**	**0.7869**	**0.7758**	**0.7786**	

Pada uji coba menggunakan seleksi fitur, metode *single learning* menghasilkan model dengan performa terbaik, yakni dengan klasifier KNN dan metode *oversampling* SMOTE, yang menghasilkan nilai *recall* sebesar 0.8546. Selanjutnya terdapat metode *Naïve Bayes* dengan metode *oversampling* yang sama, yakni SMOTE dengan nilai *recall* sebesar 0.8456. Kemudian SVM dengan metode *oversampling* ADASYN yang menghasilkan nilai *recall* 0.8054. Grafik visualisasi dari peningkatan nilai *recall* untuk setiap metode klasifikasi *single learning* dapat dilihat pada Gambar 4.3. Pada grafik tersebut dapat dilihat bahwa penggunaan metode *oversampling* menghasilkan peningkatan performa cukup signifikan, terutama pada metode SVM.



Gambar 4.5 COVID-19 - Grafik Nilai *Recall* pada Klasifier *Single Learning*

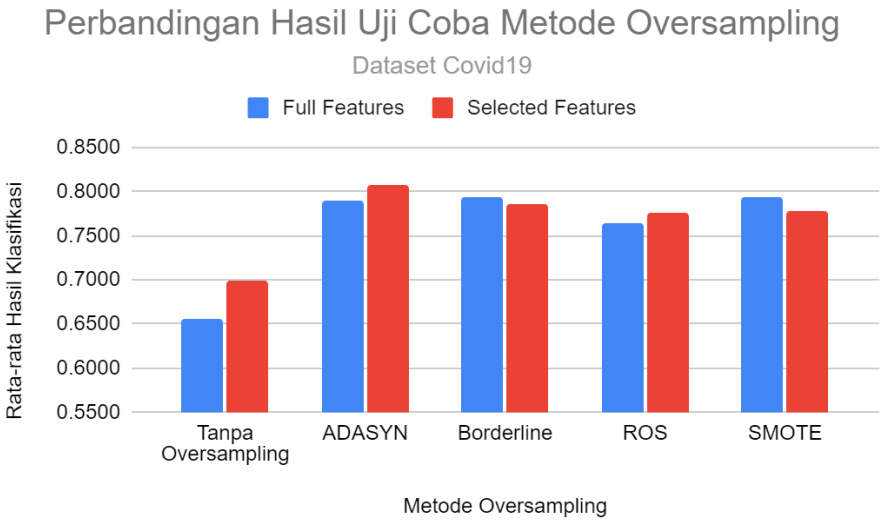


Gambar 4.6 COVID-19 - Grafik Nilai *Recall* pada Klasifier *Ensemble Learning*

Untuk metode *ensemble learning*, nilai *recall* tertinggi diperoleh dengan menggunakan metode *oversampling* ADASYN dan metode klasifikasi *XGBoost* yakni sebesar 0.8391. Disusul oleh *Adaboost* dengan metode *oversampling* yang sama, yakni ADASYN yang menghasilkan nilai *recall* sebesar 0.8295. Selanjutnya terdapat metode Bagging dengan *oversampling Borderline*-SMOTE yang menghasilkan nilai *recall* sebesar 0.8145. Model dengan performa terendah diperoleh pada metode *Random Forest* dengan *oversampling borderline*-SMOTE yang menghasilkan nilai *recall* sebesar 0.7851. Grafik visualisasi dari peningkatan nilai *recall* untuk setiap metode klasifikasi *ensemble learning* dapat dilihat pada Gambar 4.4.

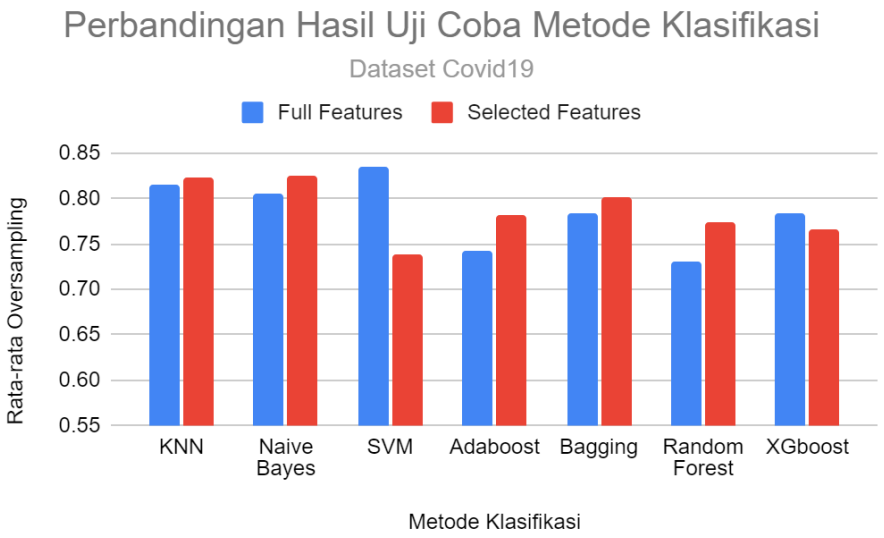
2. Uji Coba Penggunaan Metode *Oversampling*

Dari keseluruhan metode yang telah diuji coba pada dataset ini, dapat diketahui perbandingan performa setiap model yang dihasilkan antara penggunaan seluruh fitur dan seleksi fitur. Melalui grafik pada Gambar 4.7, ditampilkan perbandingan antara performa rata-rata dari metode klasifikasi yang digunakan. Pada grafik tersebut, terlihat bahwa pengaruh dari metode *oversampling* yang cukup signifikan dibandingkan tanpa menggunakan *oversampling*. Pada studi kasus ini, metode *oversampling* yang memiliki rata-rata nilai *recall* terbaik secara berurutan adalah ADASYN, *Borderline*-SMOTE, SMOTE, dan ROS.



Gambar 4.7 Perbandingan Hasil Metode *Oversampling* Dataset COVID-19

3. Uji Coba Penggunaan Metode Klasifikasi



Gambar 4.8 Perbandingan Hasil Uji Coba Seleksi Fitur Dataset COVID-19

Perbandingan hasil uji coba penggunaan metode klasifikasi dapat dilihat melalui grafik pada Gambar 4.8. Pada metode klasifikasi *single learning* dihasilkan rata-rata nilai *recall* oversampling secara berurutan dari yang terbaik adalah *Naïve Bayes*, KNN, dan SVM. Lalu untuk metode *ensemble learning*, nilai rata-rata *recall* dari yang terbaik adalah Bagging, Adaboost, *Random Forest*, dan XGBoost. Berdasarkan skenario yang telah diujikan, model

menghasilkan performa yang lebih baik ketika dilakukan *oversampling* serta penerapan seleksi fitur. Pada Gambar 4.8, sebagian besar model menghasilkan performa yang lebih baik ketika dilakukan seleksi fitur dibandingkan tanpa seleksi fitur. Pada dataset COVID-19 ini, model tanpa seleksi fitur menghasilkan rata-rata nilai *recall* sebesar 0.7592 sedangkan model dengan menggunakan seleksi fitur menghasilkan rata-rata nilai *recall* sebesar 0.7695, sedikit lebih tinggi daripada tanpa dilakukan seleksi fitur.

4.2.3 Uji Coba Dataset *Credit Card Fraud*

Pada studi kasus ini, model lebih difokuskan untuk mendeteksi terjadinya transaksi penipuan *(fraud)*, atau lebih baik memprediksi sebuah transaksi normal menjadi transaksi penipuan daripada memprediksi transaksi penipuan menjadi transaksi normal. Untuk itu, pengukuran performa yang cocok digunakan adalah *recall.* Hal ini dikarenakan *recall* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. *Recall* menjawab pertanyaan persentase prediksi transaksi penipuan dibandingkan keseluruhan transaksi penipuan yang sebenarnya. Pada dataset *Credit Card Fraud*, dilakukan beberapa skenario pengujian yang dijelaskan pada subbab berikut.

1. Uji Coba Penggunaan Seleksi Fitur

a. Tanpa Seleksi Fitur

Uji coba pertama, dilakukan menggunakan keseluruhan fitur yang tersedia. Serta dilakukan pengujian menggunakan beberapa metode *oversampling* dan metode klasifikasi yakni *ensemble learning* dan *single learning* seperti yang terdapat pada Tabel 4.5.

Tabel 4.5 Hasil Uji Dataset *Credit Card Fraud* Tanpa Seleksi Fitur

Model	Recall					
	Tanpa Oversampling	ADASYN	Borderline	ROS	SMOTE	Rata-rata Oversampling
KNN	0.8821	0.9300	0.9225	0.9224	0.9300	0.9262
Naive Bayes	0.8805	0.9068	0.9116	0.9096	0.9105	0.9096
SVM	0.8130	0.8551	0.9046	0.9128	0.9070	0.8949
Adaboost	0.8453	0.9251	0.9245	0.9305	0.9281	0.9271
Bagging	0.8699	0.8613	0.8331	0.8739	0.8818	0.8625
Random Forest	0.9024	0.8942	0.8739	0.8699	0.8942	0.8831
XGboost	0.8861	0.9359	0.9138	0.9365	0.9327	0.9297
Rata-rata Klasifikasi	**0.8685**	**0.9012**	**0.8977**	**0.9079**	**0.9121**	

Dari hasil uji coba tanpa menggunakan seleksi fitur, model yang memiliki performa terbaik diperoleh metode *ensemble learning* dengan klasifier XGBoost dan metode *oversampling* ROS dengan nilai *recall* sebesar 0.9364. Sedangkan untuk metode *single learning*, model terbaik dihasilkan dengan menggunakan klasifier KNN dan metode *oversampling* SMOTE dan ADASYN yang menghasilkan nilai *recall* sebesar 0.9300. Berdasarkan Tabel 4.3, terlihat bahwa penggunaan metode *oversampling* memiliki pengaruh terhadap performa yang dihasilkan, di mana nilai *recall* cenderung meningkat ketika menggunakan *oversampling* dibandingkan tanpa menggunakan *oversampling*.

b. Menggunakan Seleksi Fitur

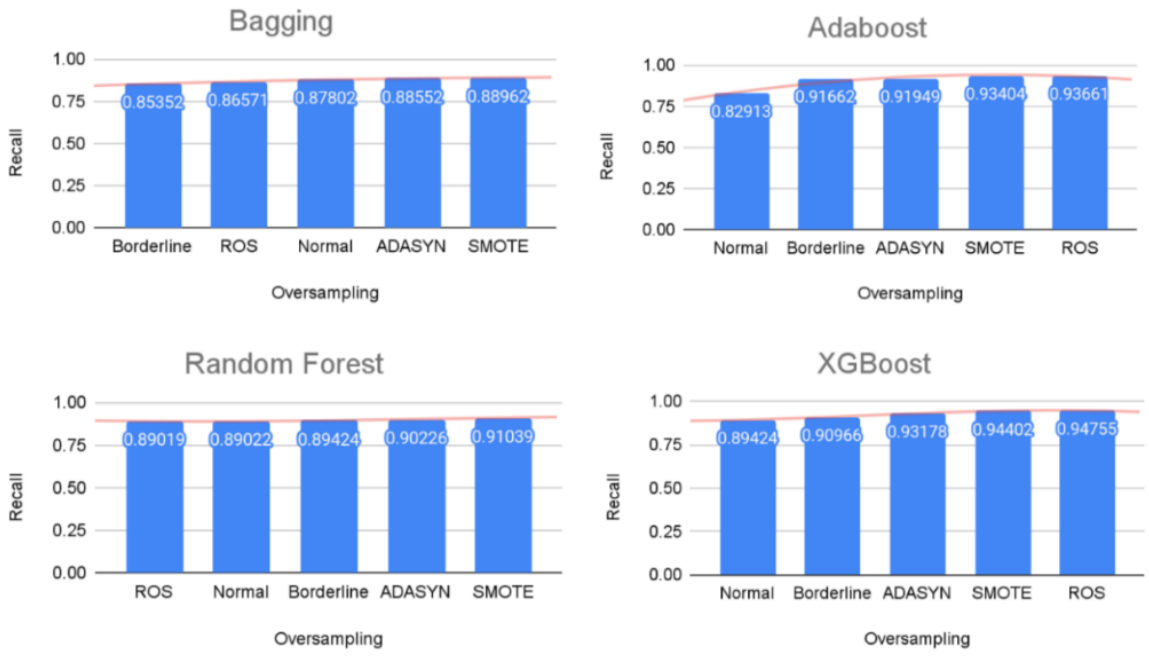
Setelah dilakukan analisis data eksplorasi, diketahui bahwa terdapat beberapa fitur yang memiliki korelasi rendah dengan data target seperti yang terlihat pada Gambar 3.21. Untuk itu, pada pengujian ini dilakukan dengan menggunakan dataset yang telah dilakukan seleksi fitur. Fitur yang digunakan di sini merupakan fitur yang memiliki nilai korelasi lebih dari 0.5 dengan

data target. Fitur yang digunakan tersebut antara lain, V1, V2, V3, V4, V5, V7, V9, V10, V11, V12, V14, V16, V17, V18. Hasil dari uji coba dataset *Credit Card Fraud* menggunakan seleksi fitur dapat dilihat pada Tabel 4.6.

Tabel 4.6 Hasil Uji Dataset *Credit Card Fraud* dengan Seleksi Fitur

Model	*Recall*					
	Tanpa *Oversampling*	ADASYN	Borderline	ROS	SMOTE	Rata-rata *Oversampling*
KNN	0.8942	0.9298	0.9185	0.9225	0.9298	0.9251
Naive Bayes	0.8995	0.9103	0.9143	0.9142	0.9157	0.9136
SVM	0.8252	0.8586	0.9175	0.9379	0.9261	0.9100
Adaboost	0.8291	0.9195	0.9166	0.9366	0.9340	0.9267
Bagging	0.8780	0.8855	0.8535	0.8657	0.8896	0.8736
Random Forest	0.8902	0.9023	0.8942	0.8902	0.9104	0.8993
XGboost	0.8942	0.9318	0.9097	0.9476	0.9440	0.9333
Rata-rata Klasifikasi	**0.8729**	**0.9054**	**0.9035**	**0.9164**	**0.9214**	

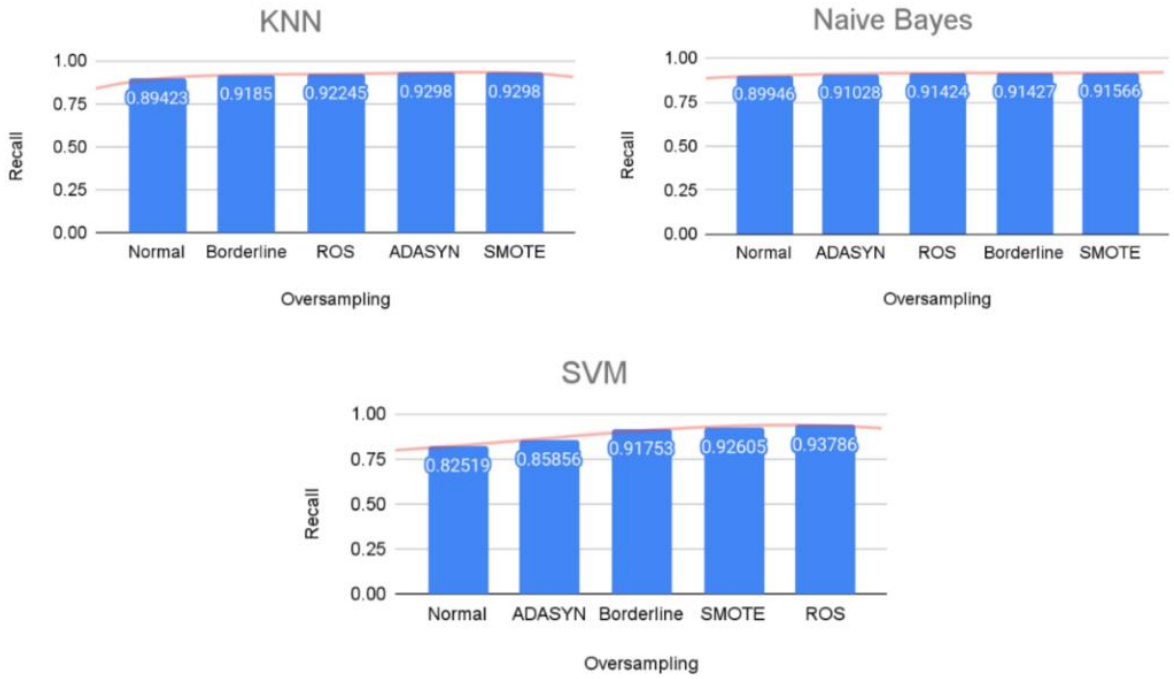
Pada uji coba menggunakan seleksi fitur, metode *ensemble learning* menghasilkan model dengan performa terbaik menggunakan klasifier XGBoost dan metode *oversampling* ROS dengan nilai *recall* sebesar 0.9476. Lalu Adaboost dengan *oversampling* ROS dengan hasil 0.9366. Kemudian *Random Forest* dengan metode *oversampling* SMOTE yang menghasilkan nilai *recall* 0.9104. Metode *ensemble learning* terbaik selanjutnya adalah metode Bagging dengan *oversampling* SMOTE yang menghasilkan *recall* sebesar 0.8896. Grafik visualisasi peningkatan nilai *recall* untuk metode klasifikasi *ensemble learning* dapat dilihat pada Gambar 4.9.



Gambar 4.9 *Credit Card Fraud* - Grafik Nilai *Recall* pada Klasifier *Ensemble Learning*

Untuk metode *single learning*, nilai *recall* tertinggi diperoleh dengan menggunakan metode *oversampling* ROS dan metode klasifikasi SVM yakni sebesar 0.9379. Diikuti oleh KNN dengan metode *oversampling* SMOTE dan ADASYN yang menghasilkan nilai *recall* sebesar 0.9298. Model dengan performa terendah diperoleh dengan metode *Naïve Bayes* dengan *oversampling* SMOTE yang menghasilkan nilai *recall* sebesar 0.9157. Grafik

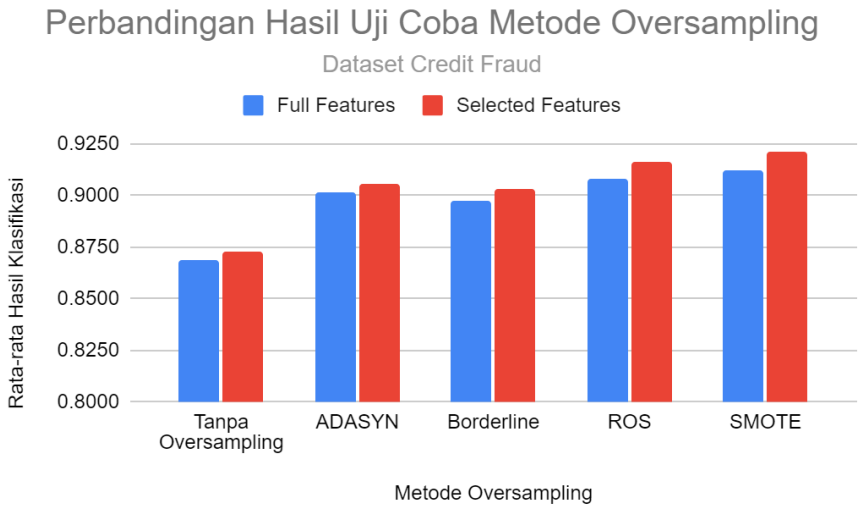
visualisasi peningkatan nilai *recall* untuk setiap metode klasifikasi *single learning* dapat dilihat pada Gambar 4.10.



Gambar 4.10 *Credit Card Fraud* - Grafik Nilai *Recall* pada Klasifier *Single Learning*

2. Uji Coba Penggunaan Metode *Oversampling*

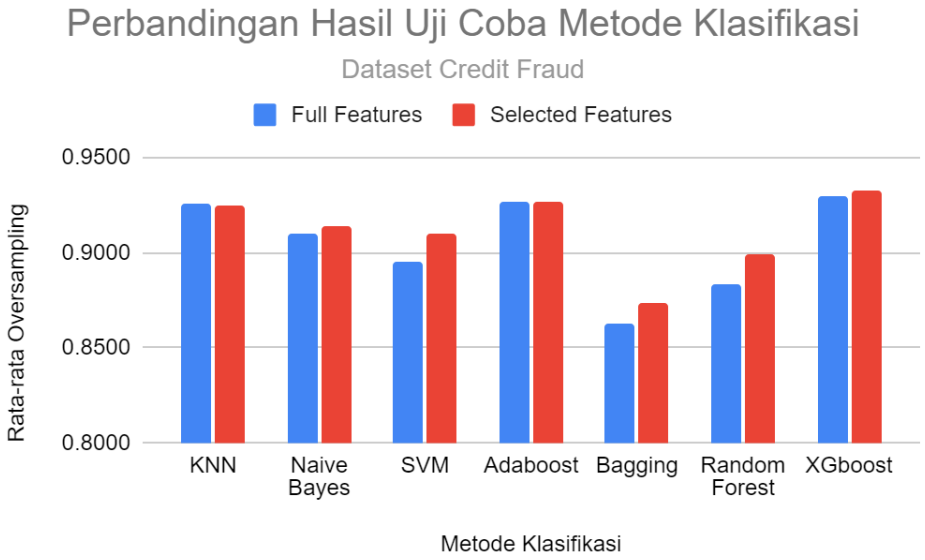
Melalui grafik pada Gambar 4.11, ditampilkan perbandingan performa metode *oversampling* antara menggunakan seleksi fitur dan tanpa seleksi fitur. Pada grafik tersebut, terlihat bahwa pengaruh dari metode *oversampling* yang cukup signifikan dibandingkan tanpa menggunakan *oversampling*. Pada studi kasus ini, metode *oversampling* yang memiliki rata-rata nilai *recall* hasil klasifikasi terbaik menggunakan seleksi fitur secara berurutan adalah SMOTE, ROS, ADASYN, dan *Borderline*-SMOTE.



Gambar 4.11 Perbandingan Hasil Metode *Oversampling* Dataset *Credit Card Fraud*

3. Uji Coba Penggunaan Metode Klasifikasi

Berdasarkan skenario yang telah diujikan, model menghasilkan performa yang lebih baik ketika dilakukan *oversampling* serta dilakukan seleksi fitur. Rata-rata nilai *recall* setiap model menghasilkan performa yang lebih baik ketika dilakukan seleksi fitur daripada tidak. Dari uji coba penggunaan metode klasifikasi, terlihat performa dari setiap metode klasifikasi yang digunakan. Perbandingan tersebut dapat dilihat melalui grafik pada Gambar 4.12. Untuk metode *ensemble learning*, nilai rata-rata *recall* mulai dari yang terbaik adalah XGBoost, Adaboost, *Random Forest* dan Bagging. Lalu untuk metode klasifikasi *single learning*, rata-rata nilai *recall* terbaik adalah KNN, *Naïve Bayes*, dan SVM.



Gambar 4.12 Perbandingan Model Hasil Uji Coba Seleksi Fitur Dataset *Credit Card Fraud*

4.2.4 Uji Coba Dataset *Car Evaluation*

Car Evaluation dataset merupakan dataset *multi-class* dengan persebaran data antar kelas tidak seimbang. Untuk itu, model lebih difokuskan untuk dapat memprediksi data setiap kelas seakurat mungkin dengan data yang sebenarnya. Untuk itu, pengukuran performa yang digunakan di sini adalah *F1-Score. F1-Score* merupakan perbandingan rata-rata presisi dan *recall*, sehingga hasil tidak akan condong ke salah satu kelas dan prediksi akan mempertimbangkan hasil yang paling optimal untuk setiap kelas. Pada subbab ini dijelaskan mengenai hasil uji coba dari dataset *Car Evaluation*.

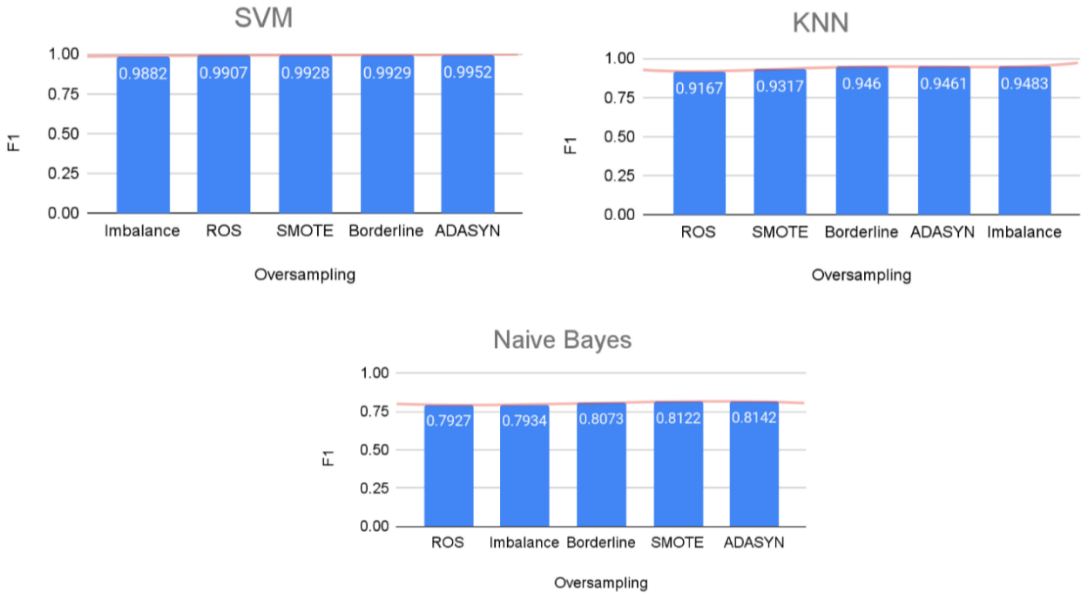
1. Uji Coba Penggunaan Seleksi Fitur

a. Tanpa Seleksi Fitur

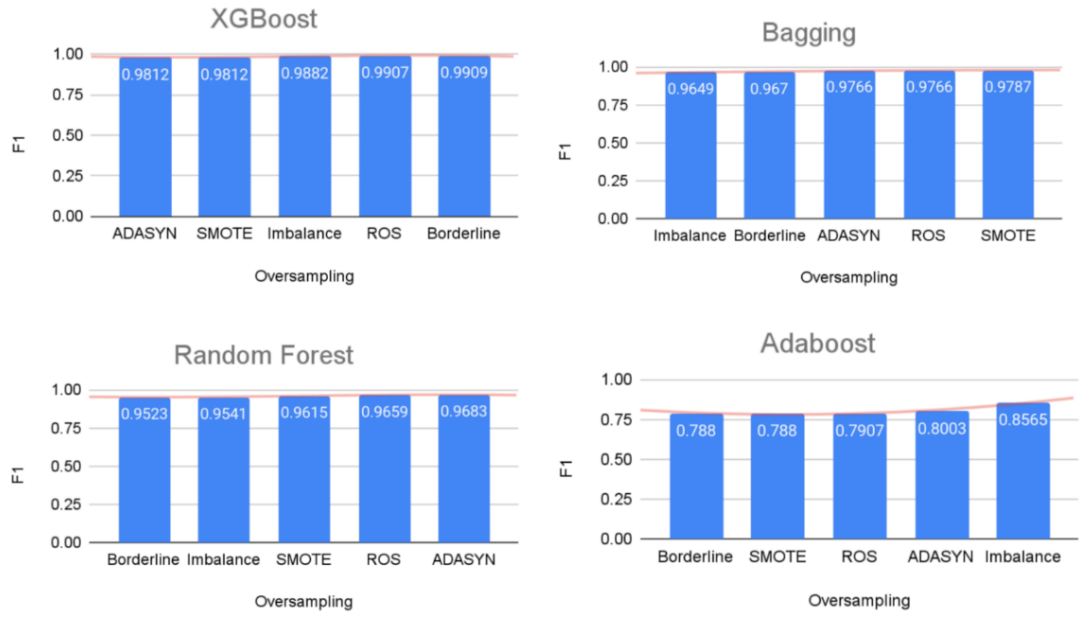
Uji coba pertama dilakukan menggunakan keseluruhan fitur yang ditampilkan pada Tabel 4.7. Dari hasil uji coba tanpa seleksi fitur, model menghasilkan performa terbaik yang diperoleh menggunakan metode *single learning* dengan klasifier SVM dan metode *oversampling* ADASYN dengan *F1-Score* sebesar 0.9952. Selanjutnya terdapat metode KNN tanpa menggunakan metode *oversampling* dengan nilai *F1-Score* sebesar 0.9483. Kemudian *Naïve Bayes* dengan metode *oversampling* ADASYN yang menghasilkan nilai *F1-Score* 0.8142. Grafik visualisasi dari peningkatan nilai *recall* untuk setiap metode klasifikasi *single learning* dapat dilihat pada Gambar 4.13.

Tabel 4.7 Hasil Uji Dataset *Car Evaluation* Tanpa Seleksi Fitur

Model	F1-Score					
	Tanpa Oversampling	ADASYN	Borderline	ROS	SMOTE	Rata-rata Oversampling
KNN	0.9483	0.9461	0.9460	0.9167	0.9317	0.9351
Naive Bayes	0.7934	0.8142	0.8073	0.7927	0.8122	0.8066
SVM	0.9882	0.9952	0.9929	0.9907	0.9928	0.9929
Adaboost	0.8565	0.8003	0.7880	0.7907	0.7880	0.7918
Bagging	0.9649	0.9766	0.9670	0.9766	0.9787	0.9747
Random Forest	0.9541	0.9683	0.9523	0.9659	0.9615	0.9620
XGboost	0.9882	0.9812	0.9909	0.9907	0.9812	0.9860
Rata-rata Klasifikasi	**0.9236**	**0.9260**	**0.9206**	**0.9177**	**0.9209**	



Gambar 4.13 *Car Evaluation* - Grafik *F1-Score* pada Klasifier *Single Learning*



Gambar 4.14 *Car Evaluation* - Hasil *F1-Score* pada Klasifier *Ensemble Learning*

Sedangkan untuk metode *ensemble learning*, model terbaik dihasilkan dengan menggunakan klasifier XGBoost dan metode *oversampling Borderline*-SMOTE yang menghasilkan *F1-Score* sebesar 0.9909. Disusul oleh Bagging dengan metode *oversampling SMOTE* yang menghasilkan *F1-Score* sebesar 0.9787. Selanjutnya metode *Random Forest* dengan *oversampling* ADASYN yang menghasilkan *F1-Score* sebesar 0.9683. Metode dengan performa terendah diperoleh dengan metode Adaboost tanpa menggunakan metode *oversampling* dengan *F1-Score* sebesar 0.8565. Grafik visualisasi dari peningkatan *F1-Score* untuk setiap metode klasifikasi *ensemble learning* dapat dilihat pada Gambar 4.14.

b. Menggunakan Seleksi Fitur

Setelah dilakukan analisis data eksplorasi pada tahap sebelumnya, diketahui bahwa terdapat beberapa fitur yang memiliki korelasi rendah dengan data target. Untuk itu, pada pengujian ini dilakukan dengan menggunakan dataset yang telah dilakukan seleksi fitur. Fitur yang digunakan di sini merupakan fitur yang memiliki nilai korelasi lebih dari 0.1 dengan data target, antara lain *buying, maint, persons, lug_boot,* dan *safety*. Hasil dari uji coba dataset *Car Evaluation* menggunakan seleksi fitur dapat dilihat pada Tabel 4.8.

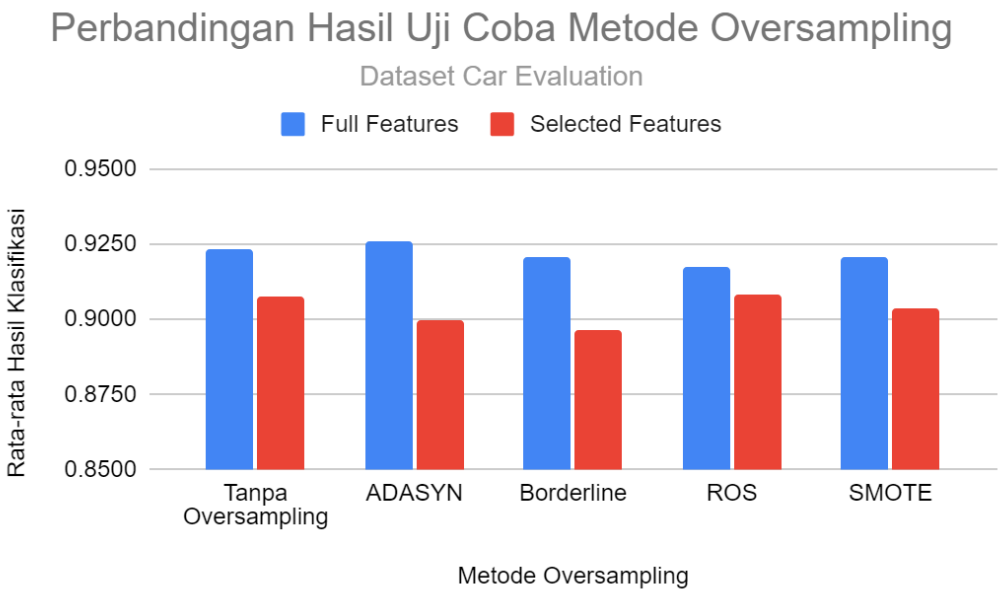
Tabel 4.8 Hasil Uji Dataset *Car Evaluation* dengan Seleksi Fitur

Model	F1-Score					
	Tanpa Oversampling	ADASYN	Borderline	ROS	SMOTE	Rata-rata Oversampling
KNN	0.9278	0.9315	0.9231	0.9263	0.9224	0.9258
Naive Bayes	0.7986	0.7942	0.7846	0.7990	0.7954	0.7933
SVM	0.9408	0.9632	0.9519	0.9566	0.9610	0.9582
Adaboost	0.8624	0.8023	0.7863	0.8023	0.8023	0.7983
Bagging	0.9320	0.9383	0.9269	0.9490	0.9372	0.9379
Random Forest	0.9505	0.9196	0.9575	0.9683	0.9610	0.9516
XGboost	0.9421	0.9489	0.9448	0.9566	0.9486	0.9497
Rata-rata Klasifikasi	**0.9077**	**0.8997**	**0.8964**	**0.9083**	**0.9040**	

Pada uji coba ini, nilai *F1-Score* tertinggi diperoleh dengan menggunakan *ensemble learning* yakni dengan klasifier *Random Forest* dan metode *oversampling* ROS dengan nilai sebesar 0.9683. Sedangkan untuk *single learning*, metode terbaik yang dihasilkan adalah XGBoost dan metode *oversampling* ADASYN dengan *F1-Score* sebesar 0.9632. Berdasarkan Tabel 4.8, pada uji coba ini seluruh model cenderung mengalami kenaikan *F1-Score* setelah dilakukan metode *oversampling*.

2. Uji Coba Penggunaan Metode *Oversampling*

Melalui grafik pada Gambar 4.15, ditampilkan perbandingan performa metode *oversampling* yang digunakan antara menggunakan seleksi fitur dan tanpa seleksi fitur. Apabila dilihat dari rata-rata metode klasifikasi yang dihasilkan pada setiap model, pengaruh metode *oversampling* tidak begitu terlihat signifikan. Hal ini dikarenakan terdapat dua dari tujuh metode klasifikasi menghasilkan *F1-Score* yang sedikit lebih tinggi ketika tidak dilakukan *oversampling*, yakni metode KNN dan Adaboost. Sedangkan untuk kelima metode *oversampling* lainnya, *F1-Score* yang dihasilkan lebih baik ketika dilakukan *oversampling*. Pada studi kasus ini, metode *oversampling* yang memiliki rata-rata nilai *F1-Score* terbaik pada hasil uji coba tanpa seleksi fitur, yang secara berurutan adalah ADASYN, SMOTE, *Borderline*-SMOTE dan ROS.

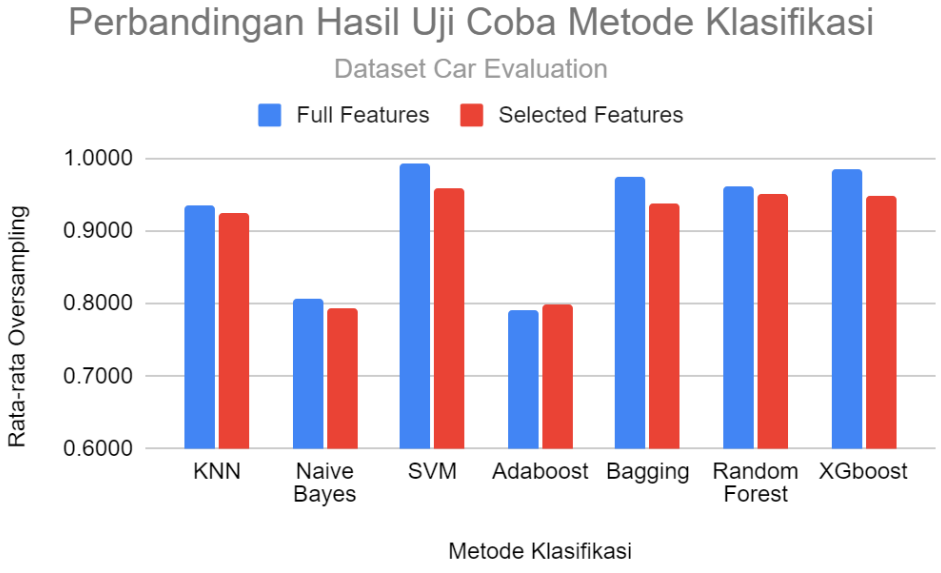


Gambar 4.15 Perbandingan Hasil Metode *Oversampling* Dataset *Car Evaluation*

3. Uji Coba Penggunaan Metode Klasifikasi

Berdasarkan skenario yang telah diujikan sebelumnya, model menghasilkan performa yang lebih baik ketika dilakukan *oversampling* dan tanpa seleksi fitur. Pada dataset *Car Evaluation* ini, model tanpa seleksi fitur menghasilkan rata-rata *F1-Score* sebesar 0.9297 sedangkan model dengan menggunakan seleksi fitur menghasilkan rata-rata lebih rendah yakni dengan *F1-Score* sebesar 0.91185.

Dari keseluruhan metode yang telah diuji coba pada dataset ini, terlihat performa dari setiap metode klasifikasi yang digunakan yang dapat dilihat melalui grafik pada Gambar 4.15. Untuk metode klasifikasi *single learning*, rata-rata nilai *F1-Score* terbaik secara berurutan adalah SVM, KNN, dan *Naïve Bayes*. Lalu untuk metode *ensemble learning*, nilai rata-rata *F1-Score* mulai dari yang terbaik adalah XGBoost, Bagging, *Random Forest* dan Adaboost.



Gambar 4.16 Perbandingan Model Hasil Uji Coba Seleksi Fitur Dataset *Car Evaluation*

4.3 Pembahasan

Pada subbab pembahasan ini, akan dibahas mengenai seluruh tahapan implementasi yang telah dilakukan pada penelitian ini, hasil dari uji coba, dan evaluasi pada setiap dataset. Pada tahap *preprocessing* dataset tidak seimbang, sebagian besar tahapannya sama dengan tahap *pre-processing* dasar *data mining* pada umumnya, antara lain *Exploratory Data Analysis* (EDA) untuk mengetahui karakteristik setiap dataset, seleksi fitur apabila terdapat fitur yang kurang relevan, pembagian data latih dan data uji, hingga *scaling* data. Hal yang membedakan antara *preprocessing* data seimbang dan tidak seimbang adalah adanya tahap *resampling*, yakni pembuatan sampel data pada kelas minoritas agar jumlah data setiap kelas memiliki jumlah yang seimbang.

Tabel 4.9 Perbandingan Karakteristik Dataset

Dataset	**Karakteristik**							
	Jumlah Data	**Jumlah Fitur**	**Rasio Kelas**	**Jenis Kelas**	**Jenis Data**	**Kategori *Imbalanced***	**Kategori Dataset**	**Tahap *Pre-Processing***
Haberman's Survival	306	4	73.5:26.5	*Binary*	Numerikal	Rendah	Kecil	Dasar
COVID-19	602	14	86:14	*Binary*	Numerikal	Sedang	Sedang	Dasar
Credit Card Fraud	284807	31	99.83:0.17	*Binary*	Numerikal	Tinggi	Besar	Dasar + Feature Engineering
Car Evaluation	1728	7	70:22:0.39:0.37	*Multi-class*	Kategorikal	Sedang	Sedang	Dasar + Data Transformation

Selain tahap *preprocessing* dasar tersebut, perlakuan *preprocessing* setiap dataset dapat berbeda antara satu dengan yang lainnya. Hal ini bergantung dengan karakteristik masing-masing dataset, di mana karakteristik tersebut dapat diketahui melalui tahap EDA yang telah dilakukan sebelumnya. Pada penelitian ini, digunakan beberapa dataset dengan karakteristik berbeda-beda, untuk itu tahapan *preprocessing* yang dilakukan dapat berbeda pula. Rangkuman karakteristik dan tahap *preprocessing* yang dilakukan pada setiap dataset, dapat dilihat pada Tabel 4.9.

Kategori dataset *imbalanced* di sini dibagi menjadi tiga jenis, yakni rendah, sedang, dan tinggi. *Imbalanced* rendah berarti memiliki rasio data tidak seimbang dengan proporsi data minoritas 20-40%. *Imbalanced* sedang berarti rasio data tidak seimbang dengan proporsi data minoritas sebesar 1-20%. Lalu *imbalanced* tinggi berarti proporsi data kelas minoritas kurang dari 1% dari data keseluruhan. Untuk jumlah data pada dataset dikategorikan menjadi tiga jenis, yakni dataset kecil, sedang, dan besar. Dataset kecil memiliki jumlah data kurang dari 1000 baris, dataset sedang memiliki jumlah data antara 1000-5000 baris, lalu dataset besar memiliki jumlah data lebih dari 5000 baris.

Berdasarkan Tabel 4.9, tahap *preprocessing* yang digunakan di sini terdiri dari *preprocessing* dasar dan *preprocessing* tambahan. *Preprocessing* dasar yakni *preprocessing* yang terdiri dari EDA dan *scaling* data. Sedangkan *preprocessing* tambahan yakni *preprocessing* lain yang dilakukan bergantung dengan karakteristik dari dataset yang digunakan. Pada dataset *Haberman's Survival* dan COVID-19 dilakukan menggunakan *preprocessing* dasar saja. Sedangkan untuk dataset *Credit Card Fraud*, dilakukan *preprocessing* tambahan berupa *feature engineering* pada fitur *Time* yang diperlukan untuk mendapatkan informasi lain dari fitur tersebut. Dataset terakhir, *Car Evaluation*, dilakukan tahap *preprocessing* tambahan berupa *data transformation*, yakni mengubah data kategorikal menjadi data numerik agar lebih memudahkan untuk dilakukan pengolahan berikutnya.

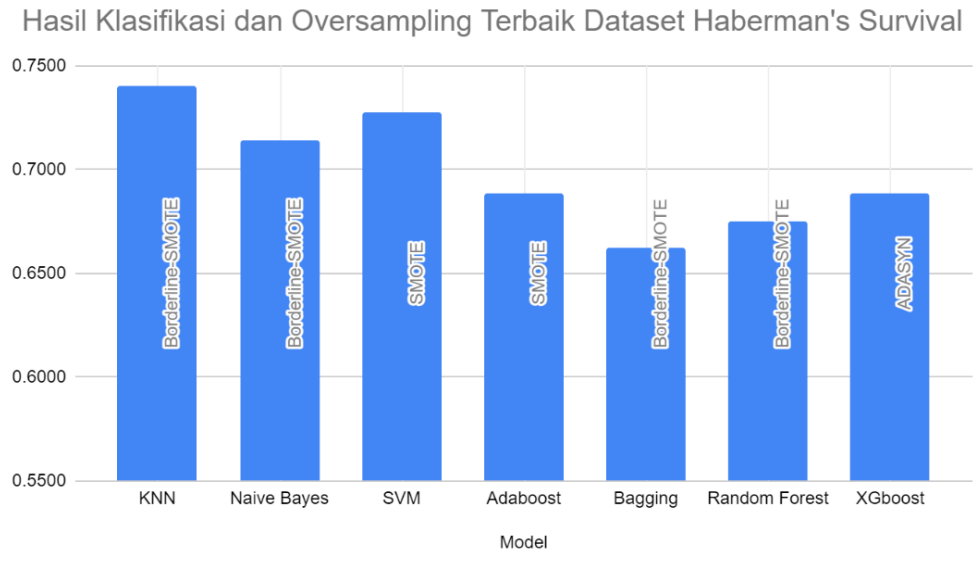
Pada tahap *resampling*, data kelas minoritas dibuat data sintetis sebanyak data kelas mayoritas. Semua metode *oversampling* menghasilkan data sintetis kelas minoritas sesuai

dengan jumlah data kelas mayoritas, kecuali metode *oversampling* ADASYN. Hal ini dikarenakan pada metode *oversampling* ADASYN, data minoritas disintetis berdasarkan densitas atau kepadatan dari data kelas minoritas tersebut, sehingga data sintetis yang dihasilkan tidak dapat sama persis dengan jumlah data kelas mayoritas, melainkan mendekati jumlah data kelas mayoritas.

Setelah dilakukan *preprocessing*, setiap dataset dilakukan tahap *training* model dengan menggunakan metode klasifikasi dan beberapa skenario uji coba. Hasil dari *training* model kemudian dilakukan validasi menggunakan data uji untuk mengetahui hasil prediksi dari model. Hasil validasi kemudian dilakukan evaluasi menggunakan pengukuran performa *Recall* dan *F1-Score* bergantung dengan karakteristik dataset.

Berdasarkan hasil skenario uji coba yang telah dilakukan, diperoleh hasil bahwa tiga dari empat dataset menghasilkan performa yang lebih tinggi ketika dilakukan seleksi fitur. Dataset tersebut yakni, *Haberman's Survival,* COVID-19, dan *Credit Card Fraud.* Sedangkan untuk dataset *Car Evaluation*, menghasilkan performa yang lebih baik ketika tidak dilakukan seleksi fitur. Dari uji coba penggunaan seleksi fitur, kemudian dilakukan analisa model yang menghasilkan performa terbaik berdasarkan metode *oversampling* dan metode klasifikasi yang digunakan.

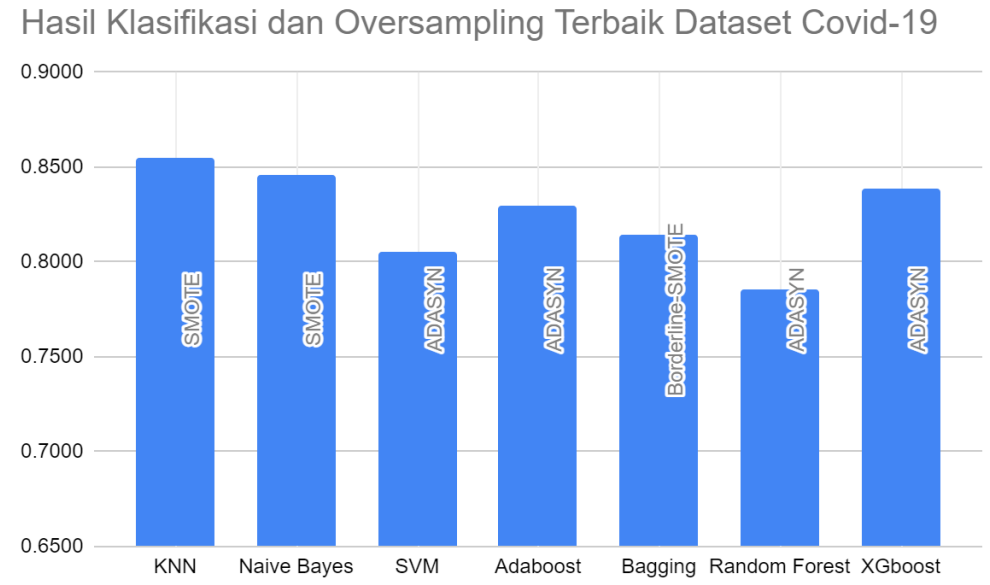
Dari hasil dari uji coba penggunaan metode *oversampling* dan tanpa *oversampling*, diperoleh hasil bahwa penggunaan metode *oversampling* dapat meningkatkan performa dari klasifikasi yang dihasilkan pada dataset yang memiliki rasio *imbalance* yang tinggi, sedangkan untuk dataset dengan rasio *imbalance* rendah kurang menghasilkan pengaruh yang signifikan. Selain metode *oversampling*, dilakukan uji coba metode klasifikasi untuk setiap dataset. Kinerja metode *oversampling* dan klasifikasi berbeda-beda untuk setiap dataset. Pada dataset *Haberman's Survival*, metode klasifikasi yang menghasilkan performa terbaik adalah KNN dengan *oversampling Borderline*-SMOTE dengan grafik perbandingan metode terbaik lainnya ditampilkan pada Gambar 4.17.



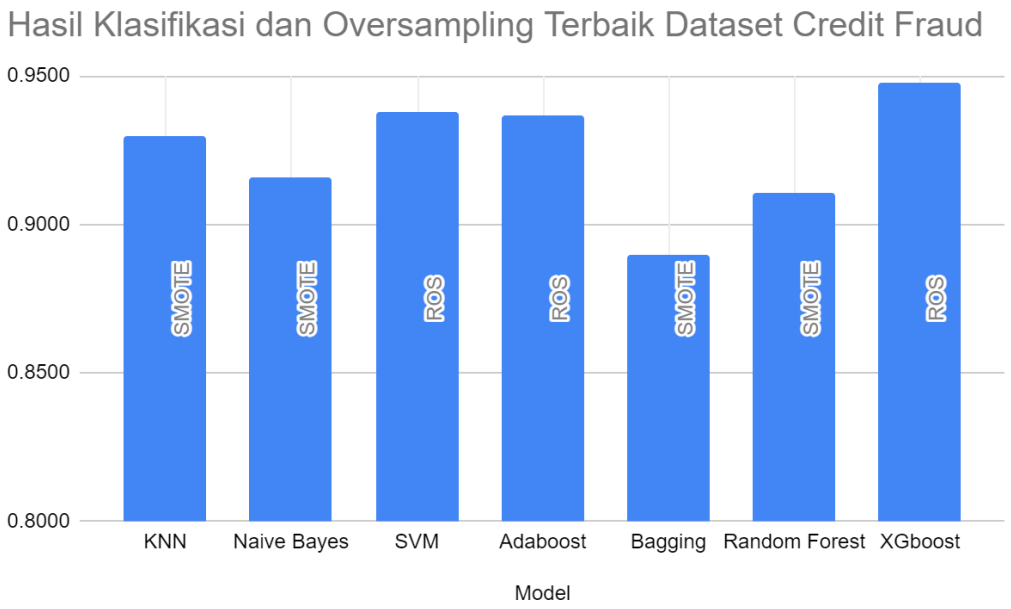
Gambar 4.17 Hasil Klasifikasi dan *Oversampling* Terbaik dataset *Haberman's Survival*

Untuk dataset COVID-19, metode klasifikasi yang menghasilkan performa terbaik adalah KNN dengan *oversampling* SMOTE, dan metode performa terbaik lainnya ditampilkan pada Gambar 4.18. Lalu pada dataset *Credit Card Fraud*, metode klasifikasi yang menghasilkan performa terbaik adalah XGBoost dengan *oversampling* ROS yang ditampilkan pada Gambar 4.19. Terakhir, untuk dataset *Car Evaluation*, metode klasifikasi yang menghasilkan performa

terbaik adalah XGBoost dengan *oversampling Borderline*-SMOTE yang ditampilkan pada Gambar 4.20.

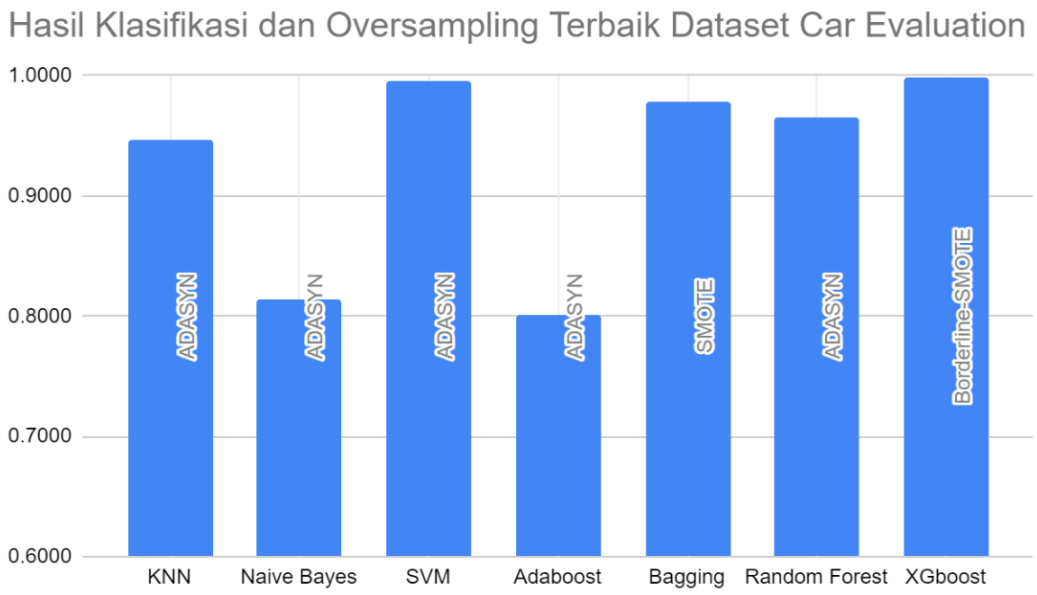


Gambar 4. 18 Hasil Klasifikasi dan *Oversampling* Terbaik dataset COVID-19



Gambar 4. 19 Hasil Klasifikasi dan *Oversampling* Terbaik dataset Credit Card Fraud

Untuk analisa hasil dari kinerja metode *single learning* dan *ensemble learning*, dapat dilihat melalui hasil klasifikasi terbaik yang telah diuji cobakan yang ditampilkan pada Tabel 4.10. Pada dataset *Haberman's Survival* dan COVID-19, metode *single learning* menghasilkan klasifikasi terbaik dengan menggunakan metode KNN, dan metode *ensemble learning* terbaik diperoleh menggunakan metode klasifikasi XGBoost namun dengan performa yang lebih rendah dibandingkan dengan metode *single learning*. Pada dataset *Credit Card* dan *Car Evaluation*, metode *ensemble learning* menghasilkan nilai lebih tinggi dibandingkan dengan *single learning*, yakni dengan menggunakan metode XGBoost, sedangkan untuk hasil metode *single learning* terbaik diperoleh dengan menggunakan metode SVM.



Gambar 4. 20 Hasil Klasifikasi dan *Oversampling* Terbaik dataset Car Evaluation

Tabel 4.10 Perbandingan Hasil Metode Klasifikasi Terbaik

Dataset	Single Learning		Ensemble Learning	
	Metode	**Nilai**	**Metode**	**Nilai**
Haberman's Survival	KNN	0.7403	XGBoost	0.6883
COVID-19	KNN	0.8546	XGBoost	0.8391
Credit Card Fraud	SVM	0.9379	XGBoost	0.9476
Car Evaluation	SVM	0.9952	XGBoost	0.9989

Apabila dilihat dari karakteristik dataset dan metode terbaik yang dihasilkan, metode *ensemble learning* baik digunakan untuk memprediksi data dengan tipe kategorikal, seperti pada dataset *Car Evaluation.* Pada dataset tersebut, metode *ensemble learning* cenderung memberikan hasil klasifikasi yang lebih baik dibandingkan dengan dataset lainnya. Hal ini dikarenakan *base learning* dari metode *ensemble* merupakan *decision tree,* di mana metode yang dapat bekerja dengan sangat baik pada data kategorikal dibandingkan pada numerikal. Sedangkan untuk dataset dengan tipe numerikal lainnya, tidak semua metode *ensemble learning* menghasilkan performa yang lebih baik daripada metode *single learning*. Hal ini dikarenakan pada metode *single learning,* digunakan optimasi untuk setiap keputusannya, sedangkan pada beberapa metode *ensemble learning,* digunakan *random sample* dari dataset sebagai inisiasi model. Selain itu, metode *single learning* yang digunakan di sini merupakan metode *strong learner* atau metode klasifikasi yang dapat menghasilkan performa lebih baik dibandingkan dengan metode klasifikasi *single learning* lainnya. Sehingga hasil klasifikasi yang diperoleh juga tidak terlalu jauh dengan hasil yang diperoleh menggunakan metode *ensemble learning.*

Selain dengan melihat hasil klasifikasi terbaik, pembahasan selanjutnya dilakukan berdasarkan hasil prediksi *True Positive* dan *True Negative* yang dihasilkan setiap model melalui metode evaluasi *confusion matix.* Pada penelitian ini difokuskan kepada pengaruh penggunaan metode *oversampling* dalam meningkatkan hasil klasifikasi pada kelas minoritas.

Untuk itu, akan dilakukan analisis lebih lanjut dengan melihat peningkatan hasil klasifikasi pada kelas minoritas melalui hasil pemetaan dari *confusion matrix.* Dengan menggunakan *confusion matrix*, persebaran data dari hasil prediksi setiap model yang diujikan dapat dilihat dan dapat diketahui seberapa banyak model dapat mengklasifikasikan data minoritas dengan tepat.

a. Haberman's Survival

Pada hasil prediksi *True Positive* dan *True Negative* dataset *Haberman's Survival,* hal yang diprioritaskan di sini adalah kelas minoritas yakni banyaknya model memprediksi pasien yang dapat bertahan hidup kurang dari 5 tahun dengan benar. Data yang digunakan di sini merupakan data *testing* yang terdiri dari 57 data pasien yang dapat bertahan hidup lebih dari 5 tahun, dan 20 data pasien yang dapat bertahan hidup kurang dari 5 tahun. Pada Tabel 4.11, dilakukan rekapitulasi hasil prediksi *True Positive* dan *True Negative* dari beberapa metode *oversampling* dan tanpa *oversampling*. Kolom *short (True Positive)* yakni banyaknya data dari kelas yang dapat bertahan kurang dari 5 tahun dapat diprediksi dengan benar, sedangkan kolom *long (True Negative)* yakni banyaknya data dari kelas yang dapat bertahan lebih dari 5 tahun diprediksi dengan benar.

Tabel 4.11 Hasil *True Positive* dan *True Negative* Dataset *Haberman's Survival*

Model	Tanpa Oversampling		SMOTE		ROS		ADASYN		Borderline		Rata-rata Oversampling	
	Short	**Long**	**Short**	**Long**	**Short**	**Long**	**Short**	**Long**	**Short**	**Long**	**Short**	**Long**
KNN	1	**53**	5	**51**	5	46	5	52	5	52	5	50
Naive Bayes	**3**	52	4	49	4	49	4	50	6	49	5	49
SVM	2	50	8	48	7	47	8	47	7	47	8	47
Adaboost	3	51	**12**	41	5	48	12	32	9	43	10	41
Bagging	2	49	5	45	3	45	3	47	5	46	4	46
Random Forest	2	51	5	45	4	45	7	44	6	46	6	45
XGboost	2	51	5	46	6	46	5	46	6	47	6	46

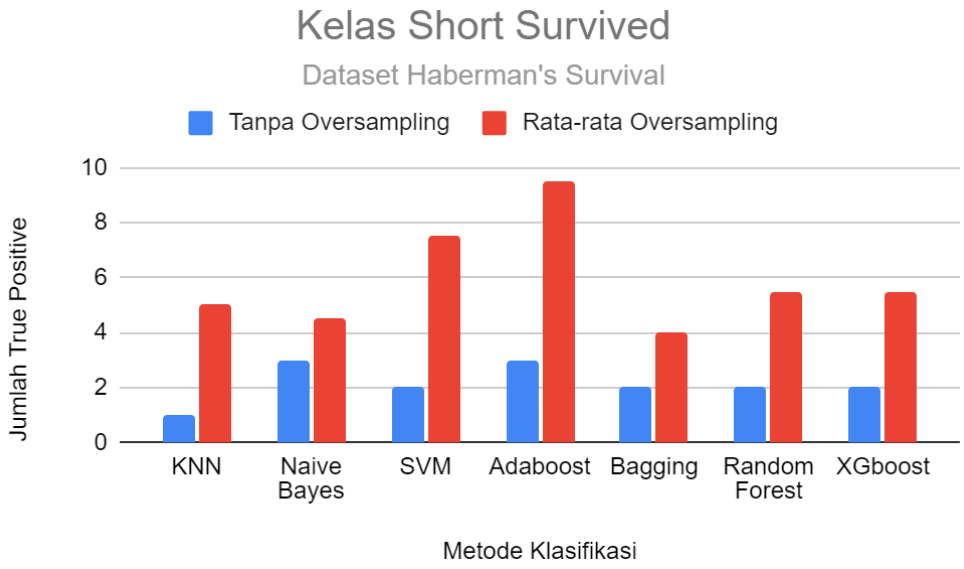
Berdasarkan Tabel 4.11, dapat dilihat bahwa model yang tidak dilakukan *oversampling*, hanya dapat memprediksi kelas *short* dengan benar paling banyak 3 dari 20 total data uji kelas *short*. Untuk kelas *long,* hasil prediksi tertinggi diperoleh sejumlah 53 dari 57 total data uji. Untuk dataset yang sudah dilakukan *oversampling*, model dapat memprediksi kelas *short* dengan benar jauh lebih tinggi dibandingkan tanpa *oversampling*, dengan hasil prediksi tertinggi sebesar 12 dari 20 data uji kelas *short.*

Melalui hasil prediksi kelas *True Positive,* terlihat pengaruh dari penggunaan metode *oversampling* dapat menghasilkan prediksi kelas minoritas dengan lebih baik. Akan tetapi, dengan kenaikan hasil prediksi kelas minoritas, diiringi dengan turunnya hasil prediksi dari kelas mayoritas, begitu pula sebaliknya. Dapat dilihat seperti hasil model KNN+Tanpa *Oversampling*, bahwa model tersebut dapat memprediksi kelas mayoritas tertinggi sebanyak 53 dari 57 data, akan tetapi kelas minoritas yang berhasil diprediksi hanya 1 dari 20 data. Hal ini dikarenakan sebelum dilakukan *oversampling,* model cenderung memprediksi data sebagai kelas mayoritas dan kurang memperhatikan kelas minoritas. Sedangkan apabila model dapat memprediksi kelas minoritas dengan lebih baik, seperti pada model Adaboost dan SMOTE, kelas mayoritas yang berhasil diprediksi juga jauh menurun dari 57 menjadi 41 data.

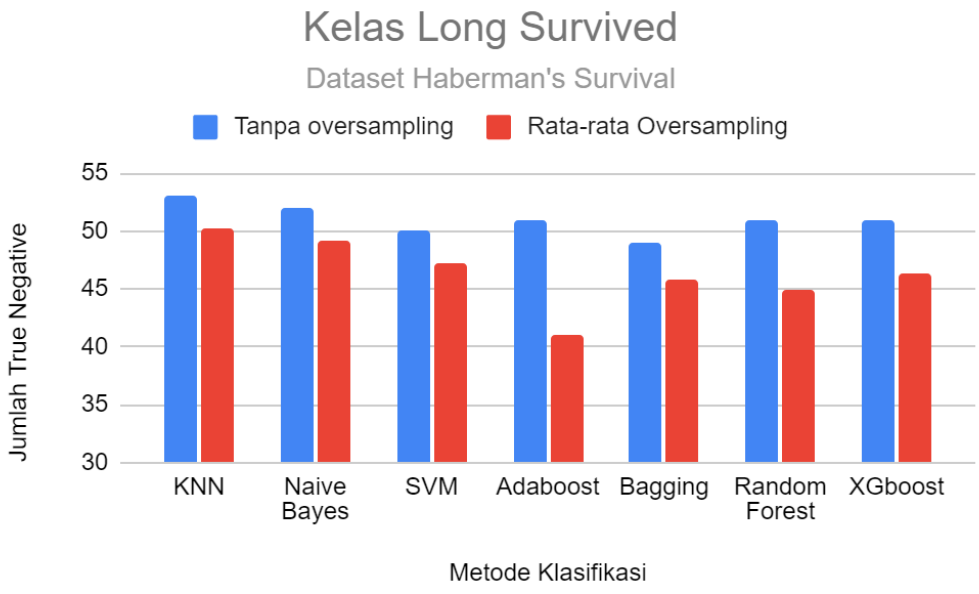
Perbandingan hasil prediksi *True Positive* tiap kelas sebelum *oversampling* dan setelah *oversampling* dapat dilihat pada Gambar 4.21 untuk kelas *Short Survived* (minoritas) dan Gambar 4.22 untuk kelas *long Survived* (mayoritas). Pada Gambar 4.21 terlihat bahwa hasil prediksi sebelum dilakukan *oversampling* yang cukup rendah dibandingkan dengan prediksi

hasil dari rata-rata keempat *oversampling*. Sedangkan untuk kelas mayoritas pada Gambar 4.22, terlihat bahwa penggunaan *oversampling* cenderung menurunkan tingkat prediksi dari sebelum menggunakan metode *oversampling*.

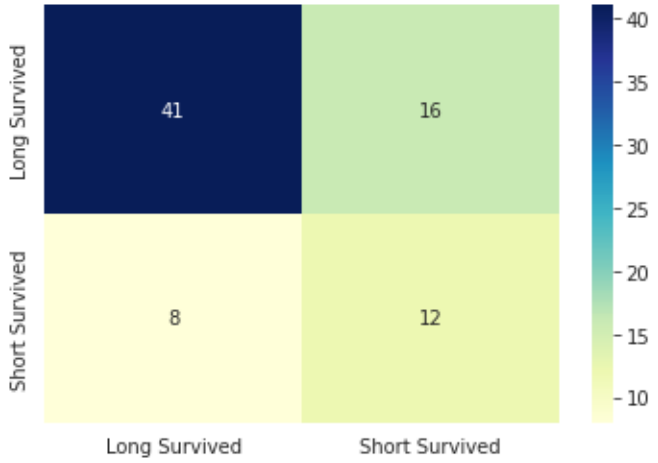
Apabila mengacu pada hasil skenario uji coba pada subbab 4.2.1, di mana pengujian dilakukan menggunakan pengukuran *recall*, maka metode yang memberikan nilai terbaik adalah KNN dengan *oversampling Borderline*-SMOTE. Apabila dilihat di sini, metode tersebut berhasil memprediksi data kelas *short* sebanyak 5 dan kelas *long* sebanyak 52, terlihat kurang baik apabila dilihat dari sisi hasil prediksi kelas minoritasnya. Sehingga baik tidaknya model sangat bergantung dengan pemilihan metode pengukuran performa yang dipilih serta kepentingan dari model tersebut.



Gambar 4.21 Perbandingan Hasil *Oversampling* pada Kelas *Short Survived*



Gambar 4.22 Perbandingan Hasil *Oversampling* pada Kelas *Long Survived*



Gambar 4.23 *Confusion Matrix* Adaboost – SMOTE Dataset *Haberman's Survival*

Sampel hasil *confusion matrix* dengan hasil prediksi kelas minoritas terbaik, yakni *Adaboost* dan SMOTE dapat dilihat pada Gambar 4.23. Baris pertama kolom pertama merupakan data kelas *long survived* yang berhasil diprediksi dengan benar, baris pertama kolom kedua merupakan data kelas *long survived* yang salah terprediksi menjadi kelas *short survived.* Lalu baris kedua kolom pertama merupakan data kelas *short survived* yang salah terprediksi menjadi kelas *long survived,* hasil prediksi ini merupakan hal yang cukup berbahaya apabila diterapkan pada dunia medis sebenarnya, hal ini dikarenakan apabila pasien yang memiliki tingkat bertahan hidup rendah *(short survived)* tetapi terprediksi menjadi memiliki tingkat bertahan hidup tinggi *(long survived)*, maka hasil prediksi tersebut dapat membahayakan keselamatan pasien. Terakhir, baris kedua kolom kedua merupakan hasil prediksi kelas *short survived* yang terprediksi dengan benar. Untuk hasil dari *confusion matrix* setiap model pada dataset ini, dapat dilihat pada Lampiran L.1.

b. Dataset COVID-19

Pada hasil prediksi *True Positive* dan *True Negative* dataset COVID-19, hal yang diprioritaskan di sini adalah kelas minoritas, yakni banyaknya pasien positif COVID-19 yang dapat diprediksi dengan benar. Data yang digunakan di sini merupakan data *testing* yang terdiri dari 107 data pasien negatif COVID-19, dan 17 data pasien positif COVID-19. Pada Tabel 4.12, dilakukan rekapitulasi hasil prediksi *True Positive* dan *True Negative* dari *confusion matrix* yang dihasilkan. Kolom positif *(True Positive)* yakni banyaknya data dari kelas pasien positif COVID-19 yang dapat diprediksi dengan benar, sedangkan kolom negatif *(True Negative)* yakni banyaknya data dari kelas pasien negatif COVID-19 yang dapat diprediksi dengan benar.

Berdasarkan Tabel 4.12, model yang tidak dilakukan *oversampling*, hanya dapat memprediksi kelas positif dengan benar paling banyak 11 dari 17 total data uji kelas positif, lalu untuk hasil prediksi kelas negatif tertinggi diperoleh sebanyak 104 dari 107 total data uji kelas negatif. Untuk dataset yang sudah dilakukan *oversampling*, model dapat memprediksi kelas positif dengan benar lebih tinggi dibandingkan tanpa menggunakan *oversampling*, dengan hasil prediksi paling tinggi sebesar 16 dari 17 data uji kelas positif, lalu hasil prediksi kelas negatif paling tinggi sebanyak 100 dari 107 data uji kelas negatif.

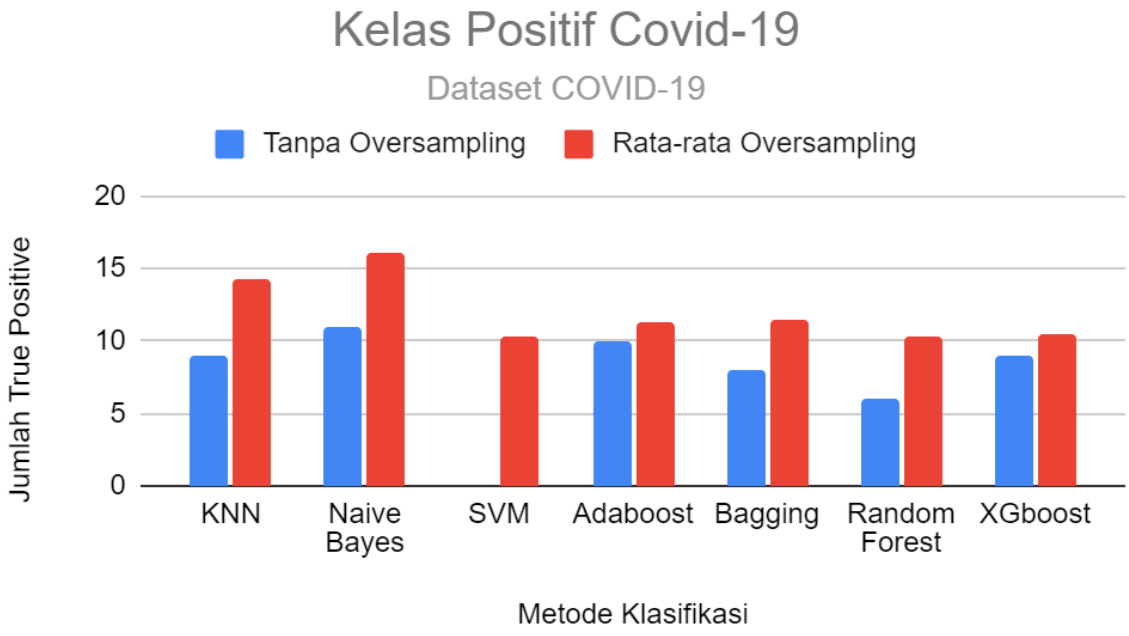
Hasil klasifikasi kelas minoritas tertinggi diperoleh sebesar 16 dengan menggunakan metode klasifikasi *Naïve Bayes* dan keempat metode *oversampling*. Melalui hasil prediksi *True Positive* dan *True Negatif*, terlihat pengaruh dari penggunaan *metode oversampling* yang dapat menghasilkan prediksi kelas minoritas dengan lebih baik, namun diiringi dengan turunnya hasil prediksi dari kelas mayoritas, begitu pula sebaliknya. Dapat dilihat seperti hasil model SVM, apabila tidak dilakukan *oversampling*, model tersebut dapat memprediksi kelas mayoritas

sebanyak 104 dari 107 data, akan tetapi kelas minoritas yang berhasil diprediksi adalah 0 dari 17 data. Hal ini dikarenakan sebelum dilakukan *oversampling*, model cenderung memprediksi data sebagai kelas mayoritas dan kurang memperhatikan kelas minoritas. Apabila dilakukan *oversampling*, model tersebut dapat memprediksi kelas minoritas jauh lebih baik, yakni menjadi sekitar 8-13 data, namun dengan hasil prediksi data mayoritas yang menurun, menjadi sekitar 88-96 dari yang sebelumnya terprediksi benar mencapai 104 data.

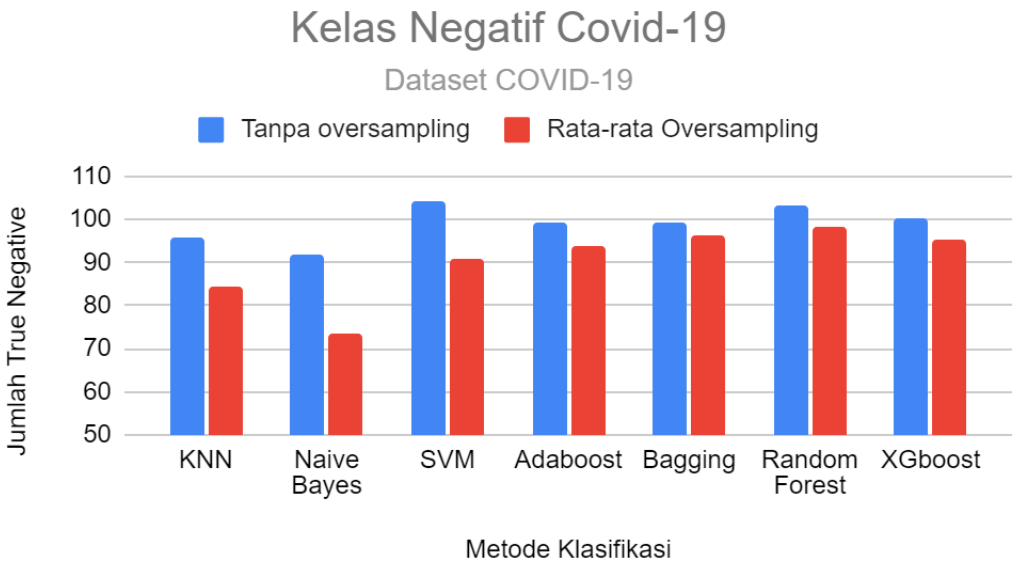
Tabel 4.12 Hasil *True Positive* dan *True Negative* Dataset COVID-19

Model	Tanpa Oversampling		SMOTE		ROS		ADASYN		Borderline		Rata-rata Oversampling	
	Positif	Negatif	Positif	Negatif	Positif	Negatif	Positif	Negatif	Positif	Negatif	Positif	Negatif
KNN	9	96	15	86	12	88	15	82	15	81	14	84
Naive Bayes	**11**	**92**	**16**	**78**	16	73	16	70	16	74	16	74
SVM	0	104	8	89	10	90	13	88	10	96	10	91
Adaboost	10	99	12	93	10	94	13	93	10	95	11	94
Bagging	8	99	11	96	11	98	12	95	12	96	12	96
Random Forest	6	103	10	97	10	100	11	96	10	100	10	98
XGboost	9	100	10	93	9	97	13	95	10	96	11	95

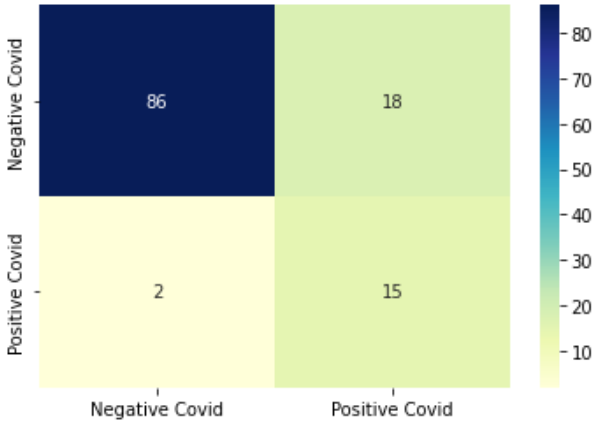
Perbandingan hasil prediksi *True Positive* dan *True Negatif* tiap kelas dari sebelum *oversampling* dan setelah *oversampling* dapat dilihat pada Gambar 4.24 untuk kelas Positif COVID-19 (minoritas), dan Gambar 4.25 untuk kelas negatif COVID-19 (mayoritas). Pada Gambar 4.24 terlihat bahwa hasil prediksi sebelum dilakukan *oversampling* cukup rendah dibandingkan dengan sesudah dilakukan *oversampling* dari rata-rata keempat metode *oversampling* yang digunakan. Sedangkan pada Gambar 4.25, terlihat bahwa penggunaan *oversampling* cenderung menurunkan tingkat prediksi kelas mayoritas daripada sebelum menggunakan metode *oversampling*.



Gambar 4.24 Perbandingan Hasil *Oversampling* pada Kelas Positif COVID-19



Gambar 4.25 Perbandingan Hasil *Oversampling* pada Kelas Negatif COVID-19



Gambar 4.26 Hasil *Confusion Matrix* Metode KNN – SMOTE Dataset COVID-19

Apabila mengacu pada hasil skenario uji coba pada bab 4.2.2, di mana pengujian dilakukan menggunakan pengukuran *recall*, maka metode yang memberikan hasil terbaik adalah KNN dengan *oversampling* SMOTE. Apabila dilihat di sini, metode tersebut berhasil memprediksi data kelas positif sebanyak 15 dan kelas negatif sebanyak 86. Pada Gambar 4.26 menampilkan hasil *confusion matrix* model dengan hasil prediksi terbaik, yakni KNN dengan *oversampling* SMOTE. Baris pertama-kolom pertama merupakan data kelas negatif COVID-19 yang berhasil diprediksi dengan benar sebanyak 86 dari 107 data, baris pertama kolom kedua merupakan data kelas negatif COVID-19 yang salah terprediksi menjadi kelas positif COVID-19. Lalu baris kedua kolom pertama merupakan data kelas positif COVID-19 yang salah terprediksi menjadi kelas negatif COVID-19, hasil prediksi ini merupakan hal yang krusial apabila diterapkan pada dunia nyata. Hal ini dikarenakan apabila pasien yang memiliki sebenarnya positif COVID-19, tetapi terprediksi menjadi negatif COVID-19, maka hasil prediksi tersebut dapat membahayakan keselamatan pasien dan orang sekitarnya, karena pasien tersebut dapat melakukan kontak erat dengan orang lain dan dapat dengan mudah menularkannya. Terakhir, baris kedua kolom kedua merupakan kelas positif COVID-19 yang dapat terprediksi dengan benar. Pada model ini dapat diprediksi pasien positif dengan benar sebanyak 15 dari 17 data, artinya dari 17 pasien yang menderita positif COVID-19, model dapat

mendeteksi 15 pasien tersebut positif COVID-19. Untuk hasil dari *confusion matrix* setiap model pada dataset COVID-19 ini, dapat dilihat pada Lampiran L.2.

c. Dataset *Credit Card Fraud*

Hasil evaluasi model menggunakan *confusion matrix* pada dataset *Credit Card Fraud* ini, hal yang diprioritaskan adalah banyaknya transaksi *fraud* yang dapat diprediksi dengan benar, di mana transaksi *fraud* merupakan kelas minoritas pada dataset ini. Data yang digunakan di sini merupakan data *testing* yang terdiri dari 71079 data transaksi normal, dan 123 data transaksi *fraud*. Pada Tabel 4.13, dilakukan rekapitulasi hasil evaluasi model menggunakan *confusion matrix* yang menampilkan prediksi *True Positive* dan *True Negatif* dari beberapa metode *oversampling* dan tanpa *oversampling*. Kolom *fraud* yakni banyaknya data dari kelas transaksi *fraud* yang dapat diprediksi dengan benar, sedangkan kolom normal yakni banyaknya data dari kelas transaksi normal yang dapat diprediksi dengan benar.

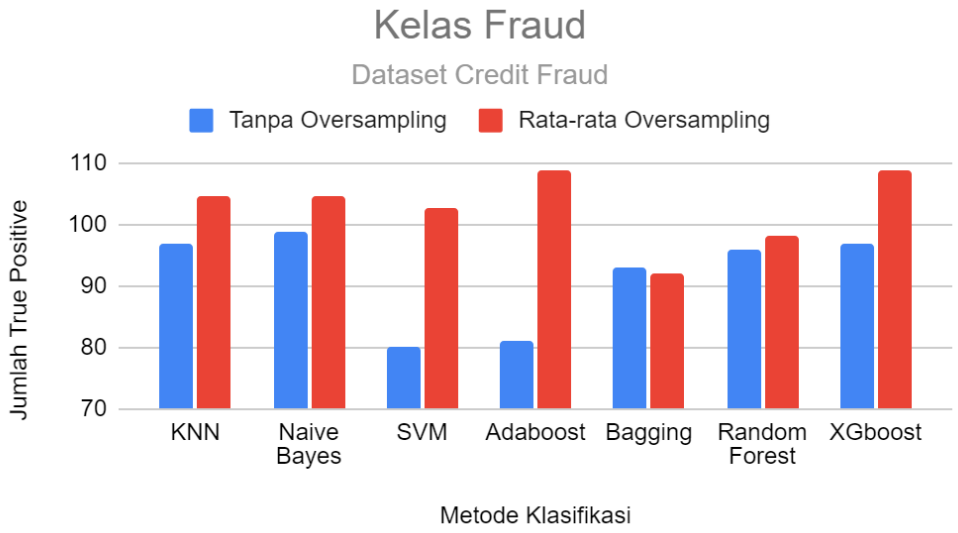
Tabel 4.13 Hasil *True Positive* dan *True Negative* Dataset *Credit Card Fraud*

Model	Imbalanced		SMOTE		ROS		ADASYN		Borderline		Rata-rata Oversampling	
	Fraud	**Normal**	**Fraud**	**Normal**	**Fraud**	**Normal**	**Fraud**	**Normal**	**Fraud**	**Normal**	**Fraud**	**Normal**
KNN	97	71068	106	70924	104	71035	106	70923	103	71050	105	70966
Naive Bayes	**99**	70656	104	70069	104	69867	106	68149	105	69294	105	69171
SVM	80	**71077**	107	69813	109	70970	92	68886	103	70913	103	69871
Adaboost	81	71059	110	69214	110	69580	**112**	65991	103	70784	109	68663
Bagging	93	71075	96	70990	90	71059	95	70985	87	71059	92	71011
Random Forest	96	71075	101	71053	96	**71071**	99	71053	97	71069	98	71058
XGboost	97	71069	111	70055	111	70557	112	67738	101	70950	109	69581

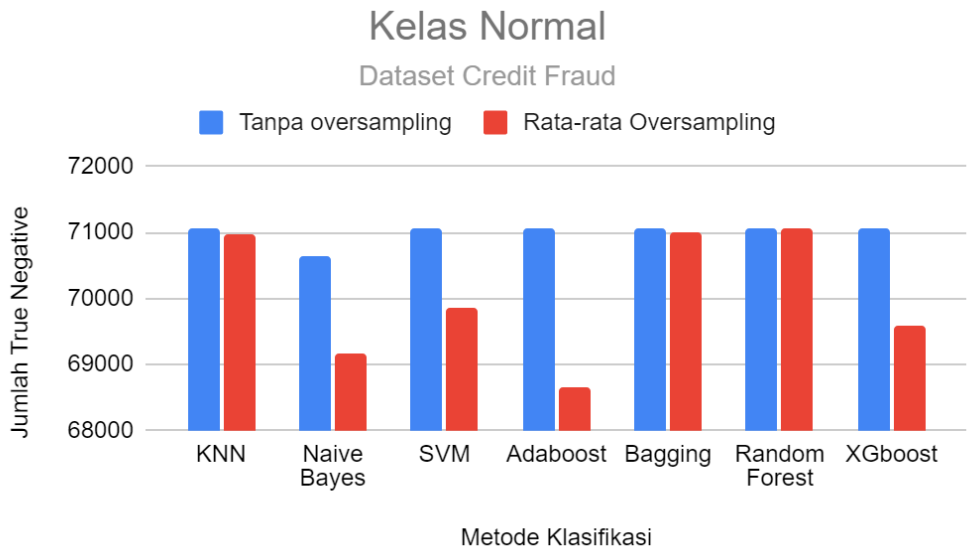
Berdasarkan Tabel 4.13, model yang tidak dilakukan *oversampling*, hanya dapat memprediksi dengan benar paling banyak 99 dari 123 total data uji kelas *fruad* dan dapat memprediksi sejumlah 71077 dari 71079 dari total data uji kelas normal. Untuk dataset yang sudah dilakukan *oversampling*, model dapat memprediksi kelas *fraud* lebih tinggi dibandingkan tanpa menggunakan *oversampling*, dengan hasil prediksi tertinggi sebesar 112 dari 123 data uji kelas *fraud*, dan hasil prediksi kelas normal tertinggi sebesar 71071 dari 71077 data uji kelas normal.

Melalui hasil prediksi *True Positive* dan *True Negatif* ini, terlihat pengaruh dari penggunaan metode *oversampling* yang dapat menghasilkan prediksi kelas minoritas dengan lebih baik namun diiringi dengan turunnya hasil prediksi dari kelas mayoritas, begitu pula sebaliknya. Dapat dilihat seperti hasil model SVM yang tidak dilakukan *oversampling*, model tersebut dapat memprediksi kelas mayoritas sebanyak 71077 dari 71077 data, atau dapat terprediksi sempurna untuk kelas tersebut. Akan tetapi kelas minoritas yang berhasil diprediksi hanyalah 80 dari 123 data. Hal ini dikarenakan, sebelum dilakukan *oversampling*, model cenderung memprediksi data sebagai kelas mayoritas dan kurang memperhatikan kelas minoritas. Apabila model ini digunakan, maka akan banyak transaksi *fraud* yang terdeteksi sebagai transaksi normal, hal ini tentu akan membahayakan dan merugikan nasabah. Setelah dilakukan *oversampling*, model tersebut dapat memprediksi kelas minoritas jauh lebih baik, yakni menjadi sekitar 109 dari 123 data, namun hasil prediksi data mayoritas yang menurun, menjadi sekitar 70970.

Perbandingan hasil prediksi *True Positive* dan *True Negatif* tiap kelas sebelum *oversampling* dan setelah *oversampling* dapat dilihat pada Gambar 4.27 untuk kelas *fraud* (minoritas), dan Gambar 4.28 untuk kelas normal (mayoritas). Pada Gambar 4.27 terlihat bahwa hasil prediksi sebelum dilakukan *oversampling* cukup rendah dibandingkan dengan sesudah dilakukan *oversampling* menggunakan rata-rata dari keempat metode *oversampling* yang digunakan. Sedangkan pada Gambar 4.28, terlihat bahwa penggunaan *oversampling* cenderung menurunkan tingkat prediksi kelas mayoritas daripada sebelum menggunakan metode *oversampling*.



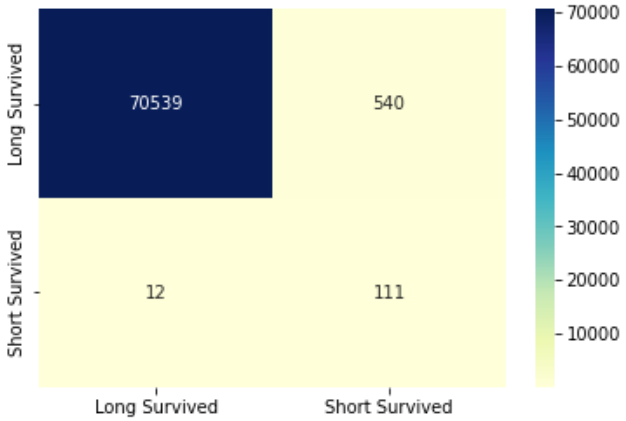
Gambar 4.27 Perbandingan Hasil *Oversampling* pada Kelas *Fraud*



Gambar 4.28 Perbandingan Hasil *Oversampling* pada Kelas Normal

Apabila mengacu pada hasil skenario uji coba pada bab 4.2.3, maka metode yang memberikan hasil terbaik adalah XGBoost dengan *oversampling* ROS. Melalui tabel 4.13, terlihat metode tersebut berhasil memprediksi data kelas *fraud* sebanyak 111 dan kelas normal sebanyak 70557 yang ditampilkan melalui *confusion matrix* pada Gambar 4.29. Baris pertama-kolom pertama merupakan data kelas *fraud* yang berhasil diprediksi dengan benar, baris pertama kolom kedua merupakan data kelas normal yang salah terprediksi menjadi kelas *fraud*.

Lalu baris kedua kolom pertama merupakan data kelas *fraud* yang salah terprediksi menjadi kelas *normal,* hasil prediksi ini merupakan hal yang cukup berbahaya karena apabila sebuah transaksi *fraud* tetapi terprediksi menjadi normal, maka nasabah tidak akan mendapatkan informasi apabila kartu kreditnya sedang terjadi transaksi penipuan yang dapat merugikan nasabah dan bank terkait. Terakhir, baris kedua kolom kedua merupakan data kelas *fraud* yang terprediksi dengan benar. Untuk hasil dari *confusion matrix* setiap model pada dataset ini, dapat dilihat pada Lampiran L.3.



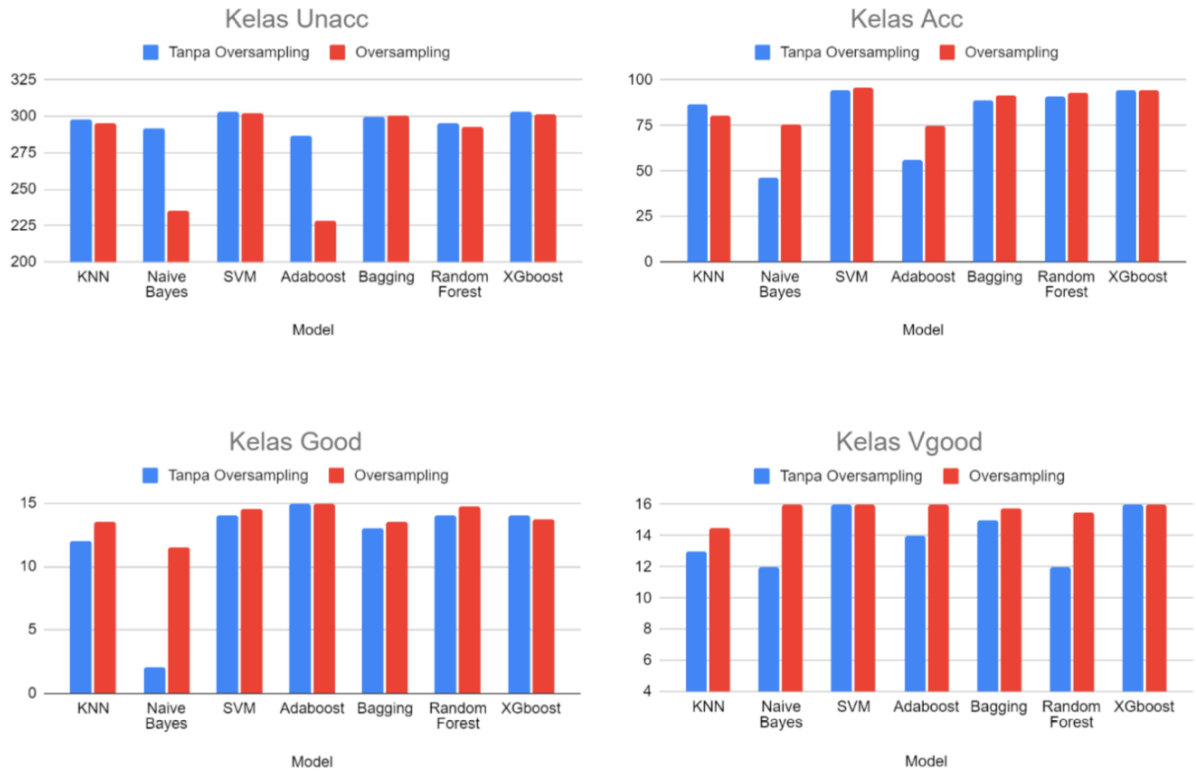
Gambar 4.29 *Confusion Matrix* XGboost – BorderlineSMOTE Dataset *Credit Card Fraud*

d. Dataset *Car Evaluation*

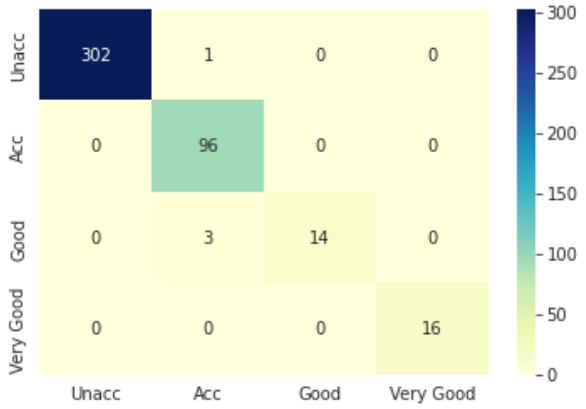
Hal yang diprioritaskan dari hasil evaluasi model menggunakan *confusion matrix* pada dataset *Car Evaluation* ini adalah banyaknya data yang dapat terprediksi dengan tepat untuk setiap kelas karena dataset ini merupakan *multiclass*. Data yang dievaluasi di sini merupakan data *testing* yang terdiri dari 303 data kelas *Unacc*, 96 data kelas *Acc*, 17 data kelas *Good*, dan 16 data kelas *Very Good*, di mana kelas *Unacc* merupakan kelas mayoritas dan kelas *acc*, *Good*, dan *VeryGood* merupakan kelas minoritas.

Pada Tabel 4.14 dilakukan rekapitulasi hasil prediksi *True Positive* atau banyakanya data tiap kelas yang diprediksi dengan benar dari metode *oversampling* yang digunakan dan tanpa *oversampling*. Pada tabel tersebut, model yang tidak dilakukan *oversampling* dapat memprediksi data dengan dengan hasil yang cukup baik, yakni 303 dari 303 kelas *Unacc*, 94 dari 97 kelas *Acc*, 14 dari 17 kelas *Good*, dan 16 dari 16 kelas *VerryGood* yang diperoleh menggunakan model SVM. Hasil klasifikasi tersebut dioptimalkan kembali dengan penggunaan metode *oversampling*, hingga diperoleh hasil klasifikasi kelas minoritas yang hampir sempurna yakni 302 dari 303 kelas *Unacc*, 96 dari 97 kelas *Acc*, 15 dari 17 kelas *Good*, dan 16 dari 16 kelas *VerryGood* yang diperoleh menggunakan metode SVM dan *oversampling* ROS.

Melalui hasil prediksi *True Positive*, terlihat pengaruh dari penggunaan metode *oversampling* yang dapat menghasilkan prediksi kelas minoritas lebih baik daripada sebelum dilakukan metode *oversampling*. Pada beberapa model, penggunaan *oversampling* dapat menurunkan hasil klasifikasi kelas mayoritas(*Unacc)*, seperti yang terlihat pada Gambar 4.30 yang menampilkan hasil prediksi kelas *Unacc* yang cenderung menurun setelah dilakukan metode *oversampling*.



Gambar 4.30 Perbandingan Hasil *Oversampling* pada Setiap Kelas Dataset *Car Evaluation*



Gambar 4.31 Hasil *Confusion Matrix* Metode SVM – ADASYN Dataset *Car Evaluation*

Apabila mengacu pada hasil skenario uji coba pada bab 4.2.4, di mana pengukuran performa dilakukan menggunakan *F1-Score*, maka metode klasifikasi terbaik yang dihasilkan adalah SVM dengan *oversampling* ADASYN. Dilihat melalui *confusion matrix* yang dihasilkan pada Gambar 4.31, di mana data yang berada pada diagonal utama (dari kiri atas ke kanan bawah) menunjukkan data yang terprediksi dengan benar, sedangkan data yang berada di luar diagonal utama merupakan data yang salah terklasifikasi menjadi kelas lainnya. Pada model ini, berhasil memprediksi data dengan benar pada kelas *Unacc* sebanyak 302 dengan 1 data salah terklasifikasi menjadi kelas *Acc*, lalu kelas *Acc* yang dapat diprediksi dengan sempurna sebanyak 96, kelas *Good* terprediksi sebanyak 14 dengan 3 data yang salah terklasifikasi menjadi kelas *Acc*, dan kelas *VerryGood* yang dapat terprediksi dengan sempurna sebanyak 16. Berdasarkan hasil *confusion matrix*, sebagian besar data yang salah terklasifikasi merupakan data kelas *Unacc* yang terprediksi menjadi kelas *Acc* dan sebaliknya, dengan hasil *confusion matrix* setiap model dapat dilihat pada Lampiran L.4.

Tabel 4.14 Hasil *True Positive* dan *True Negative* Dataset *Car Evaluation*

Model	Imbalanced				SMOTE				ROS				ADASYN				Borderline				Rata-rata Oversampling			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
KNN	298	87	12	13	298	77	14	14	288	81	15	13	296	82	13	15	298	82	13	16	295	81	14	15
Naive Bayes	292	46	2	12	235	79	14	16	229	75	14	16	240	74	16	16	237	73	16	16	235	75	12	16
SVM	**303**	**94**	**14**	**16**	303	96	14	16	**302**	**96**	**15**	**16**	302	96	14	16	303	95	15	16	303	96	14	16
Adaboost	287	56	15	14	226	75	15	16	230	72	15	16	231	76	15	16	226	75	15	16	228	75	15	16
Bagging	300	89	13	15	302	92	13	16	302	90	15	15	300	93	13	16	299	91	12	16	301	92	14	16
Random Forest	295	91	14	12	293	93	15	14	293	93	15	16	294	93	15	16	289	93	13	16	292	93	15	16
XGboost	303	94	14	16	301	94	13	16	302	95	15	16	302	93	13	16	**302**	**95**	**15**	**16**	302	94	14	16

Keterangan kolom :

1 = Kelas *Unacc*

2 = Kelas *Acc*

3 = Kelas *Good*

4 = Kelas *VeryGood*

(Halaman ini sengaja dikosongkan)

BAB 5 Kesimpulan dan Saran

Bab ini membahas tentang kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan pada bab sebelumnya. Kesimpulan nantinya sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut di masa yang akan datang.

5.1 Kesimpulan

Dalam pengerjaan Tugas Akhir ini setelah melalui tahap perancangan sistem, implementasi metode, dan uji coba, diperoleh kesimpulan sebagai berikut:

1. Cara melakukan *preprocessing* data tidak seimbang bergantung dengan karakteristik dan kebutuhan dari masing-masing dataset. Pada penelitian ini dilakukan tahap *preprocessing* dasar untuk seluruh dataset dan *preprocessing* tambahan untuk dataset yang memerlukan tindakan lebih lanjut dari *preprocessing* dasar sebelumnya. Tahap *Preprocessing* dasar terdiri dari *Exploratory Data Analysis* (EDA), *scaling* dataset, pemisahan dataset, dan *oversampling*. Sedangkan *preprocessing* tambahan hanya diterapkan pada dataset *Credit Card Fraud* yakni *feature engineering,* dan *data transformation* untuk dataset *Car evaluation*. Pada tahap seleksi fitur, dataset *Haberman's Survival,* COVID-19, dan *Credit Card Fraud* menghasilkan performa lebih tinggi ketika dilakukan seleksi fitur, sedangkan dataset *Car Evaluation* menghasilkan performa lebih baik ketika tidak dilakukan seleksi fitur.
2. Penerapan metode *oversampling* pada data latih menghasilkan performa lebih baik pada dataset yang memiliki rasio kelas data *imbalanced* tinggi, seperti pada dataset *Credit Card Fraud.* Sedangkan untuk dataset yang memiliki rasio kelas data *imbalanced* rendah, seperti dataset *Haberman's Survival,* metode *oversampling* tidak memberikan pengaruh yang signifikan dalam meningkatkan performa hasil klasifikasi.
3. Penerapan metode *ensemble learning* pada penelitian ini menghasilkan performa yang tidak terlalu signifikan dibandingkan dengan metode *single learning.* Metode *single learning* menghasilkan performa lebih baik pada dataset *Haberman's Survival* dan COVID-19 dengan menggunakan metode KNN. Sedangkan untuk metode *ensemble learning* menghasilkan performa terbaik pada dataset *Car Evaluation* dan *Credit Card Fraud* menggunakan metode XGBoost.
4. Evaluasi kinerja metode *oversampling* dan *ensemble learning* dilakukan dengan menggunakan *confusion matrix*, di mana untuk dataset *binary* digunakan pengukuran performa berupa *recall* yang menghasilkan performa tertinggi masing-masing dataset sebesar 0.6883 untuk dataset *Haberman's Survival* dengan menggunakan metode klasifikasi KNN dan metode *oversampling* Borderline-SMOTE. Dataset COVID-19 menggunakan metode klasifikasi KNN dengan metode *oversampling* SMOTE sebesar 0.8391. Dataset *Credit Card Fraud* menghasilkan *recall* sebesar 0.9476 dengan menggunakan metode klasifikasi XGBoost dan metode *oversampling* ROS. Sedangkan untuk dataset *multiclass* pada *Car Evaluation,* dilakukan pengukuran performa menggunakan *F1-Score* yang menghasilkan performa tertinggi sebesar 0.9989 menggunakan metode klasifikasi XGBoost dan metode *oversampling Borderline*-SMOTE.
5. Berdasarkan hasil prediksi *True Positive* dan *True Negative* pada masing-masing dataset, penggunaan metode *oversampling* dapat meningkatkan prediksi data minoritas, akan tetapi diiringi dengan penurunan prediksi dari kelas mayoritas

5.2 Saran

Penelitian mengenai analisis kinerja metode *oversampling* dan *ensemble learning* pada data tidak seimbang ini, dapat ditingkatkan dengan melakukan perbandingan menggunakan dataset dengan karakteristik yang mirip lainnya agar dapat dilakukan analisa lebih lanjut mengenai pengaruh penggunaan metode *oversampling* berdasarkan karakteristik dataset tersebut. Selain itu, dapat ditingkatkan pula dengan cara mencoba pengukuran performa lainnya dan melakukan *hyperparameter tunning* untuk mendapatkan parameter yang menghasilkan model dengan performa yang lebih optimal.

DAFTAR PUSTAKA

AlShourbaji, I., Helian, N., Sun, Y., & Alhameed, M. (2021). Anovel HEOMGA Approach for Class Imbalance Problem in the Application of Customer Churn Prediction. *SN Computer Science*, *2*(6), 1–12. https://doi.org/10.1007/s42979-021-00850-y

Awoyemi, John O; Adetunmbi, Adebayo O; Oluwadare, S. A. (2017). Credit card fraud detection using machine learning techniques: A comparative analysis. In *2017 international conference on computing networking and informatics (ICCNI)* (pp. 1–9).

Brandt, J., & Lanzén, E. (2020). *A Comparative Review of SMOTE and ADASYN in Imbalanced Data Classification*. 42.

Cahyana, Nurheri and Khomsah, Siti and Aribowo, A. S. (2019). Improving imbalanced dataset classification using oversampling and gradient boosting. In *2019 5th International Conference on Science in Information Technology (ICSITech)* (pp. 217–222).

Chen, R. C., Dewi, C., Huang, S. W., & Caraka, R. E. (2020). Selecting critical features for data classification based on machine learning methods. *Journal of Big Data*, *7*(1). https://doi.org/10.1186/s40537-020-00327-4

Farooq, F., Ahmed, W., Akbar, A., Aslam, F., & Alyousef, R. (2021). Predictive modeling for sustainable high-performance concrete from industrial wastes: A comparison and optimization of models using ensemble learners. *Journal of Cleaner Production*, *292*, 126032. https://doi.org/10.1016/j.jclepro.2021.126032

Ferreira, P., Le, D. C., & Zincir-Heywood, N. (2019). Exploring Feature Normalization and Temporal Information for Machine Learning Based Insider Threat Detection. *15th International Conference on Network and Service Management, CNSM 2019*, *Cnsm*. https://doi.org/10.23919/CNSM46954.2019.9012708

Gandhi, R. (2018). *K Nearest Neighbours — Introduction to Machine Learning Algorithms*. https://towardsdatascience.com/k-nearest-neighbours-introduction-to-machine-learning-algorithms-18e7ce3d802a

Ghojogh, B., & Crowley, M. (2019). *The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial*. 1–23. http://arxiv.org/abs/1905.12787

Han, W., Huang, Z., Li, S., & Jia, Y. (2019). Distribution-Sensitive Unbalanced Data Oversampling Method for Medical Diagnosis. *Journal of Medical Systems*, *43*(2). https://doi.org/10.1007/s10916-018-1154-8

Jackins, V., Vimal, S., Kaliappan, M., & Lee, M. Y. (2021). AI-based smart prediction of clinical disease using random forest classifier and Naive Bayes. *Journal of Supercomputing*, *77*(5), 5198–5219. https://doi.org/10.1007/s11227-020-03481-x

Jiang, Y., Tong, G., Yin, H., & Xiong, N. (2019). A Pedestrian Detection Method Based on Genetic Algorithm for Optimize XGBoost Training Parameters. *IEEE Access*, *7*, 118310–118321. https://doi.org/10.1109/ACCESS.2019.2936454

Kasanah, A. N., Muladi, M., & Pujianto, U. (2019). Penerapan Teknik SMOTE untuk Mengatasi Imbalance Class dalam Klasifikasi Objektivitas Berita Online Menggunakan Algoritma KNN. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *3*(2), 196–201. https://doi.org/10.29207/resti.v3i2.945

Lee, D., & Kim, K. (2021). An efficient method to determine sample size in oversampling based on classification complexity for imbalanced data. *Expert Systems with Applications*, *184*(May), 115442. https://doi.org/10.1016/j.eswa.2021.115442

Malhotra, R., & Jain, J. (2020). Handling imbalanced data using ensemble learning in software defect prediction. *Proceedings of the Confluence 2020 - 10th International Conference on Cloud Computing, Data Science and Engineering*, 300–304. https://doi.org/10.1109/Confluence47617.2020.9058124

Muqiit WS, A., & Nooraeni, R. (2020). Penerapan Metode Resampling Dalam Mengatasi Imbalanced Data Pada Determinan Kasus Diare Pada Balita Di Indonesia (Analisis Data Sdki 2017). *Jurnal MSA (Matematika Dan Statistika Serta Aplikasinya)*, *8*(1), 19. https://doi.org/10.24252/msa.v8i1.13452

Pratama, A., Wihandika, R. C., & Ratnawati, D. E. (2018). Implementasi algoritme support vector machine (SVM) untuk prediksi ketepatan waktu kelulusan mahasiswa. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *2*(April), 1704–1708.

Sastrawan, A. S. (2009). Analisis Pengaruh Metode Over Sampling Dalam Churn. *Computer*, *2009*(Snati).

Sevastianov, L. A., & Shchetinin, E. Y. (2020). On methods for improving the accuracy of multi-class classification on imbalanced data. *CEUR Workshop Proceedings*, *2639*, 70–82.

Siringoringo, R. (2018). Klasifikasi Data Tidak Seimbang Menggunakan Algoritma SMOTE dan k-Nearest Neighbor. *Jurnal ISD*, *3*(1), 44–49.

Syukron, A., & Subekti, A. (2018). Penerapan Metode Random Over-Under Sampling dan Random Forest Untuk Klasifikasi Penilaian Kredit. *Jurnal Informatika*, *5*(2), 175–185. https://doi.org/10.31311/ji.v5i2.4158

Tharwat, A. (2018). Classification assessment methods. *Applied Computing and Informatics*, *17*(1), 168–192. https://doi.org/10.1016/j.aci.2018.08.003

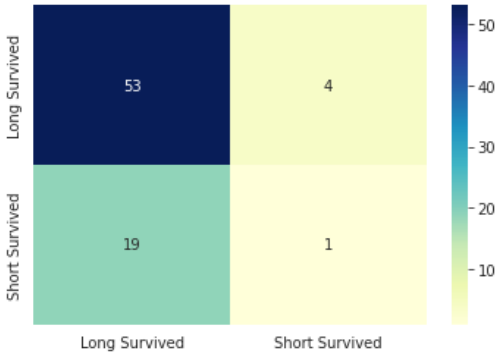
Wibowo, P., & Fatichah, C. (2021a). An in-depth performance analysis of the oversampling techniques for high-class imbalanced dataset. *Register: Jurnal Ilmiah Teknologi Sistem Informasi*, *7*(1), 63–71. https://doi.org/10.26594/register.v7i1.2206

Wibowo, P., & Fatichah, C. (2021b). Pruning-based oversampling technique with smoothed bootstrap resampling for imbalanced clinical dataset of Covid-19. *Journal of King Saud University - Computer and Information Sciences*. https://doi.org/10.1016/j.jksuci.2021.09.021

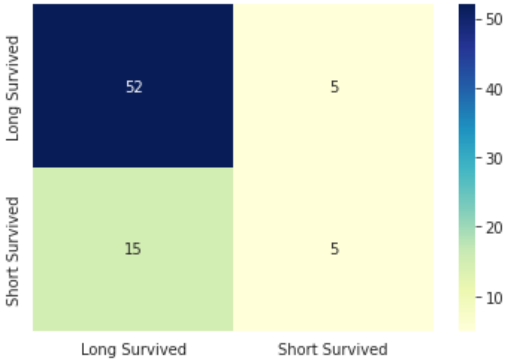
Yang, F. J. (2018). An implementation of naive bayes classifier. *Proceedings - 2018 International Conference on Computational Science and Computational Intelligence, CSCI 2018*, 301–306. https://doi.org/10.1109/CSCI46756.2018.00065

LAMPIRAN

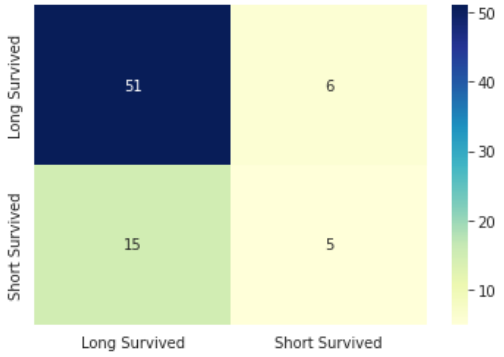
L.1 Hasil *Confusion Matrix* Dataset *Haberman's Survival* (Seleksi Fitur)



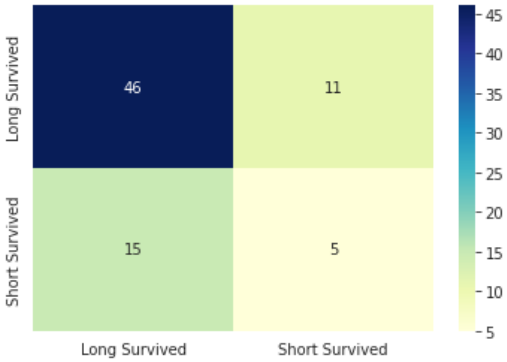
KNN - Imbalanced



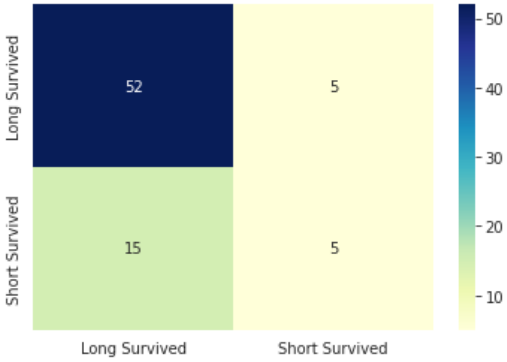
KNN - SMOTE



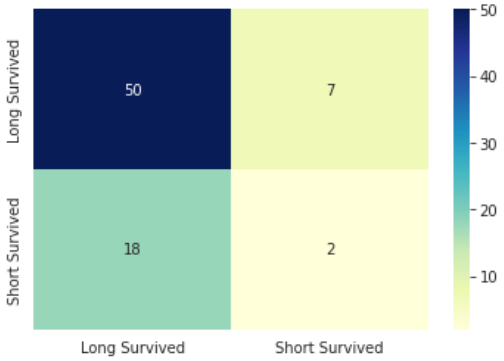
KNN - ROS



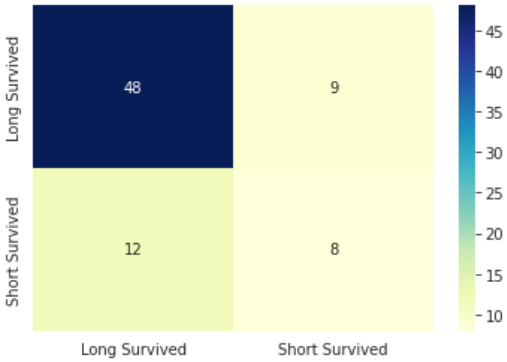
KNN - ADASYN



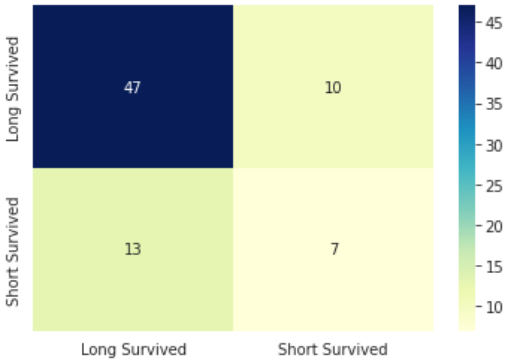
KNN - Borderline SMOTE



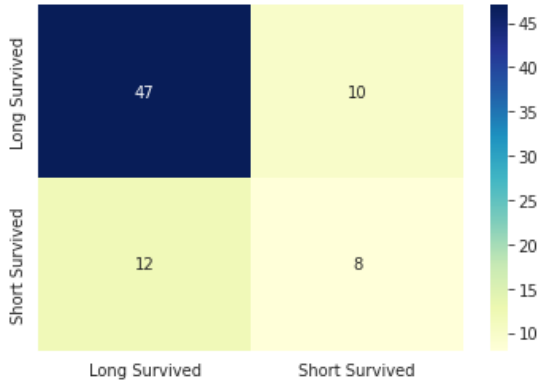
SVM – Imbalanced



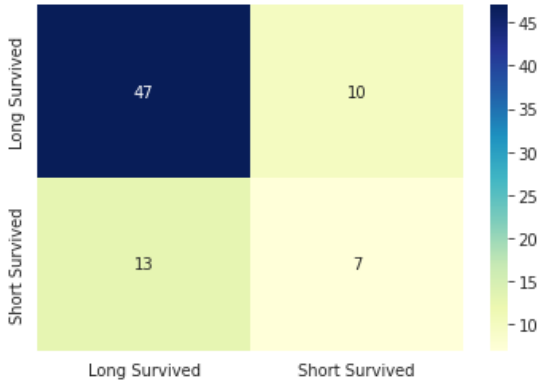
SVM – SMOTE



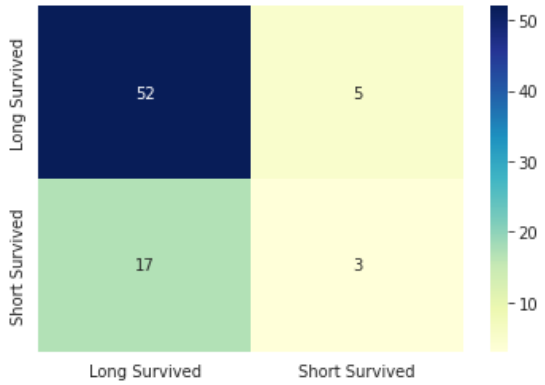
SVM – ROS



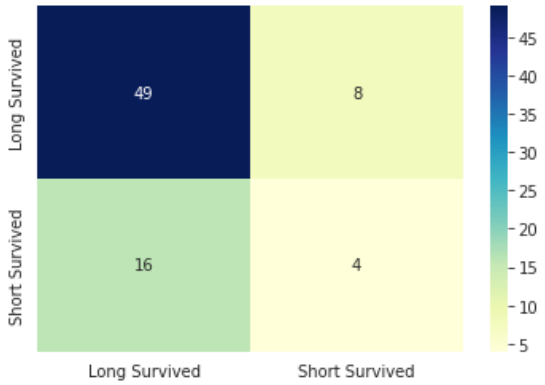
SVM – ADASYN



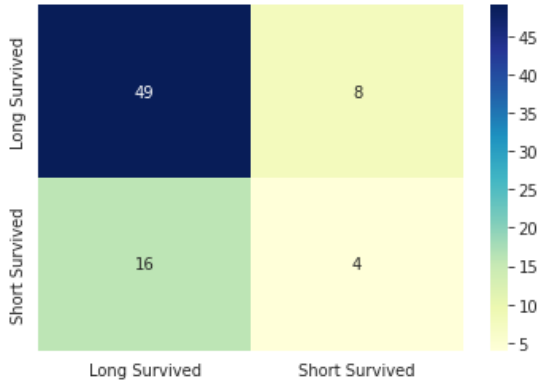
SVM – Borderline SMOTE



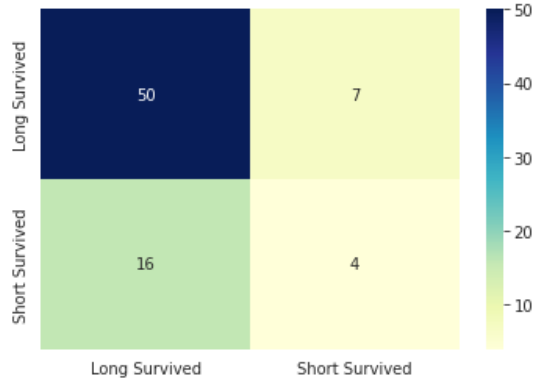
Naïve Bayes – Imbalanced



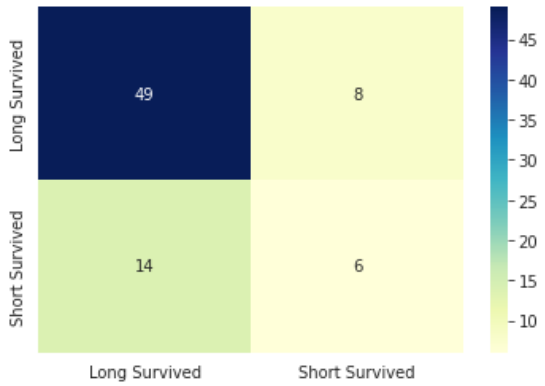
Naïve Bayes – SMOTE



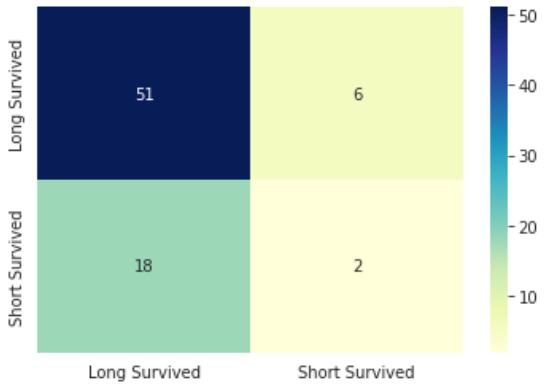
Naïve Bayes – ROS



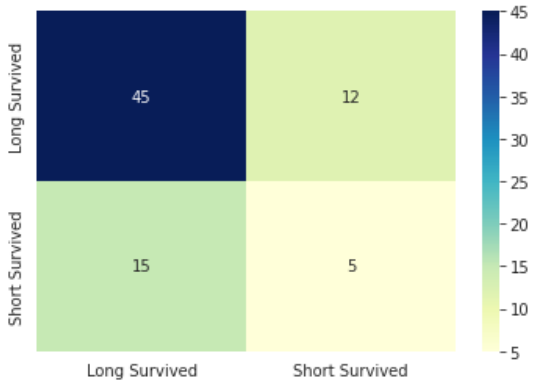
Naïve Bayes – ADASYN



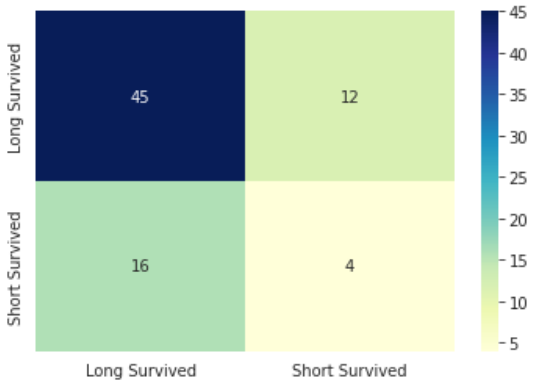
Naïve Bayes – Borderline SMOTE



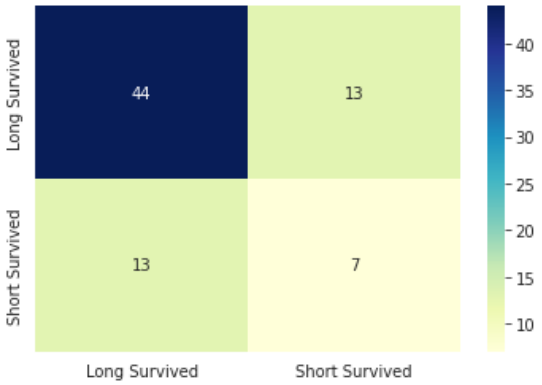
Random Forest – Imbalanced



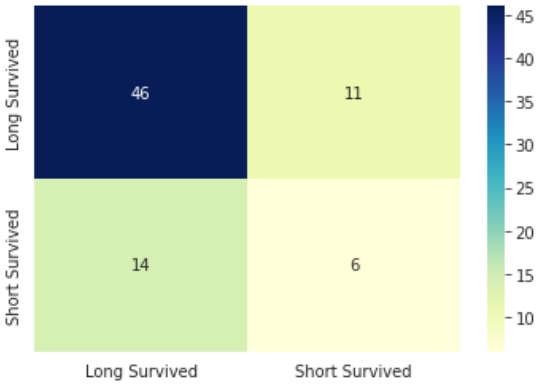
Random Forest – SMOTE



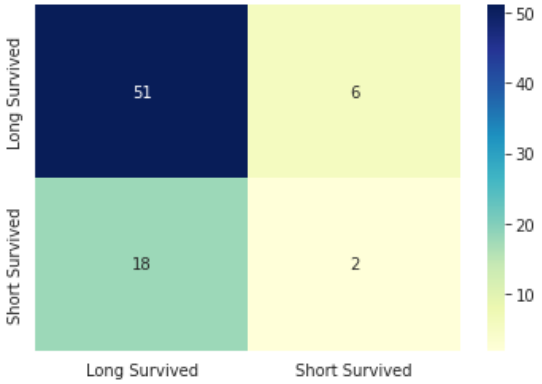
Random Forest – ROS



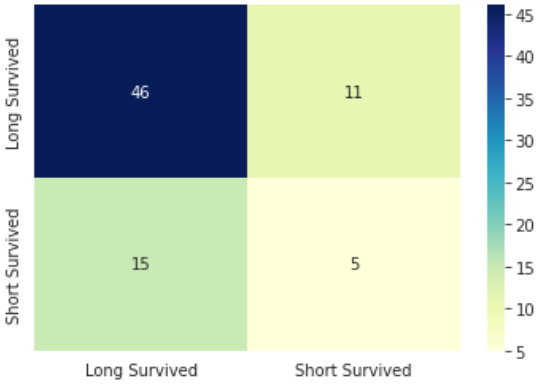
Random Forest – ADASYN



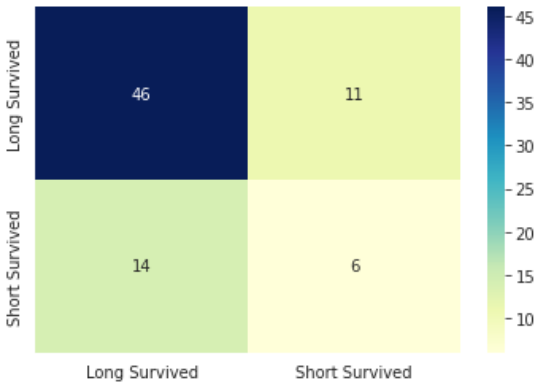
Random Forest – Borderline SMOTE



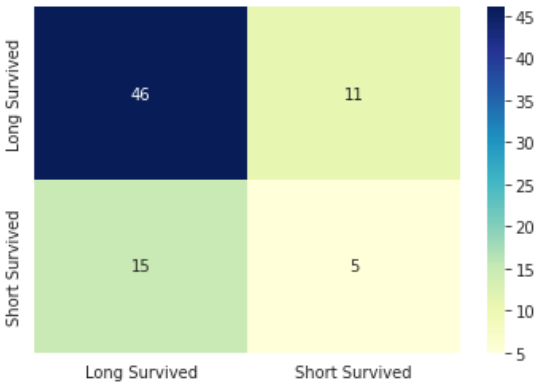
XGboost – Imbalanced



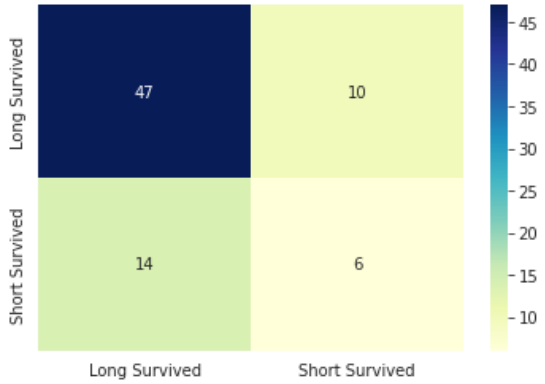
XGboost – SMOTE



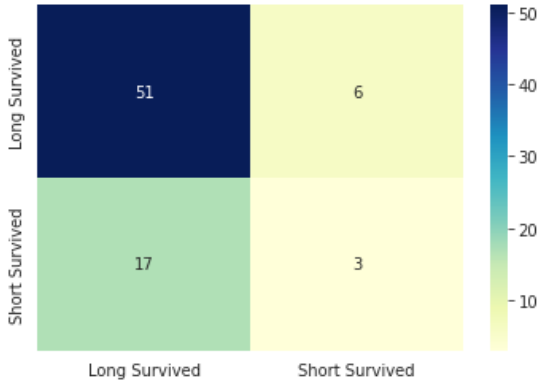
XGboost – ROS



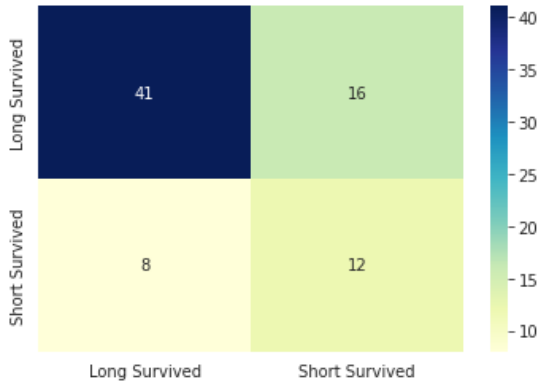
XGboost – ADASYN



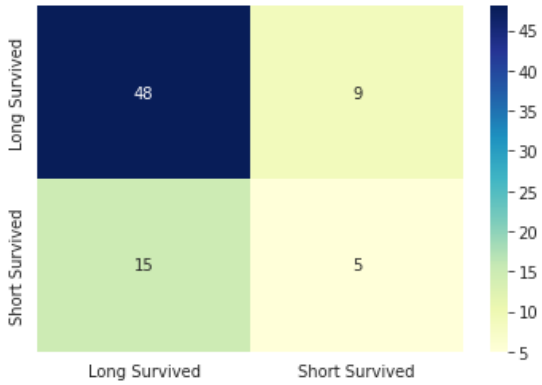
XGboost – Borderline SMOTE



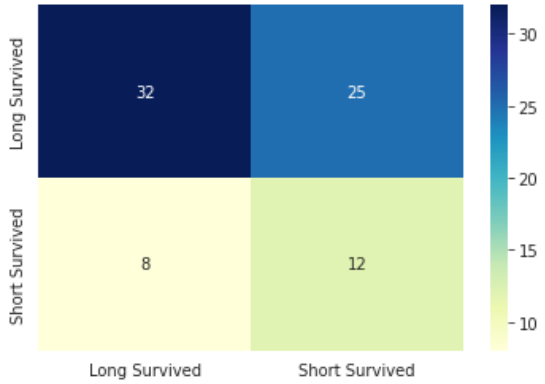
Adaboost – Imbalanced



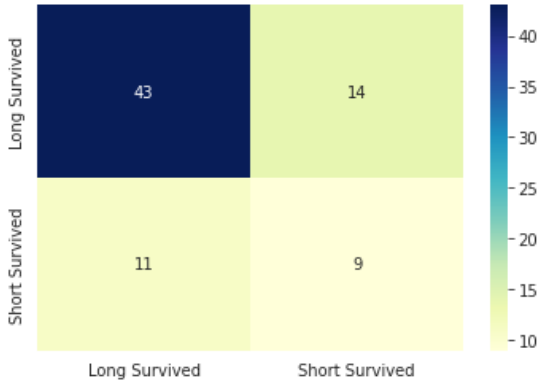
Adaboost – SMOTE



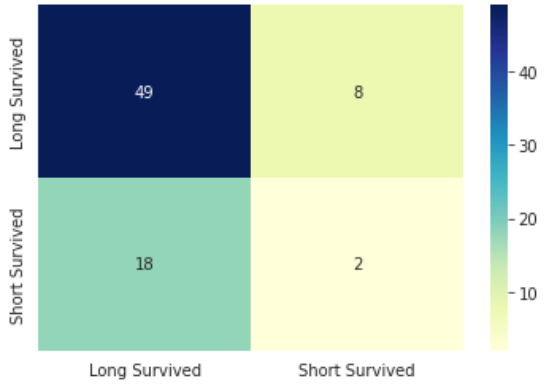
Adaboost – ROS



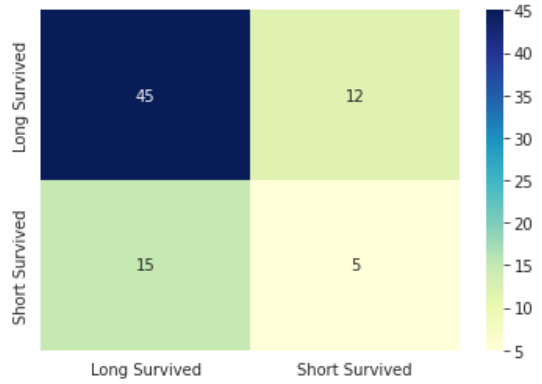
Adaboost – ADASYN



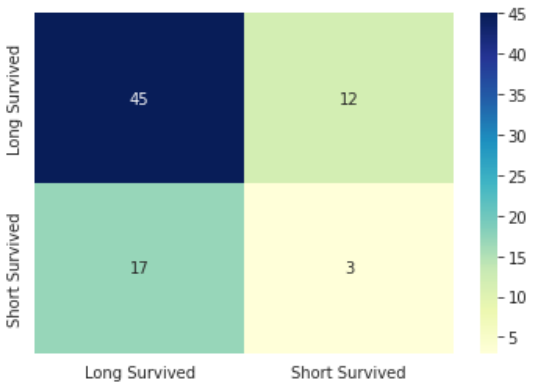
Adaboost – Borderline SMOTE



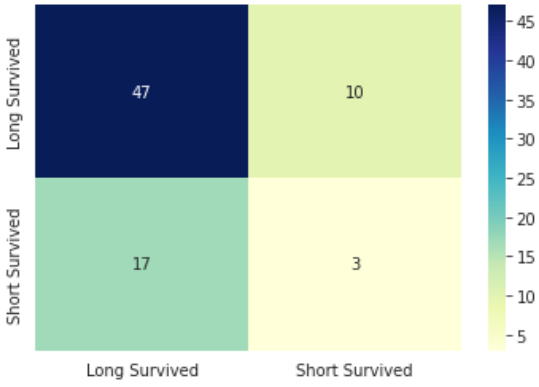
Bagging – Imbalanced



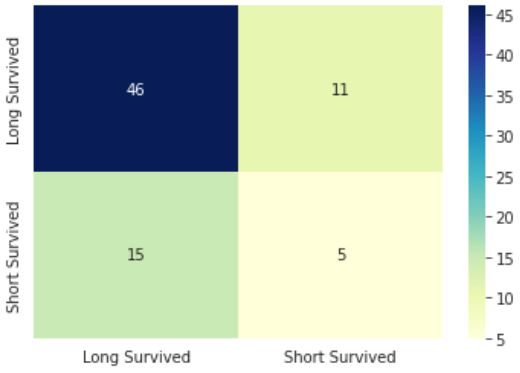
Bagging – SMOTE



Bagging – ROS

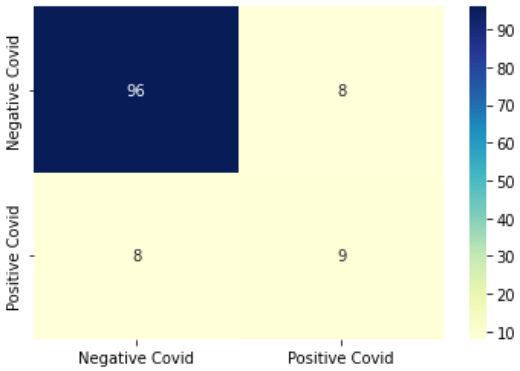


Bagging – ADASYN

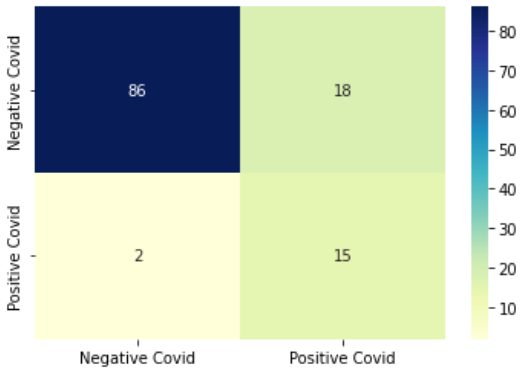


Bagging – Borderline SMOTE

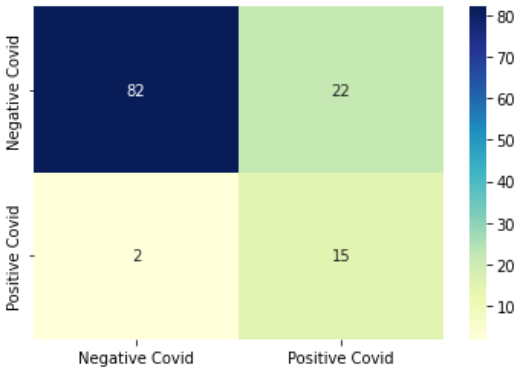
L.2 Hasil *Confusion Matrix* Dataset COVID-19 (Seleksi Fitur)



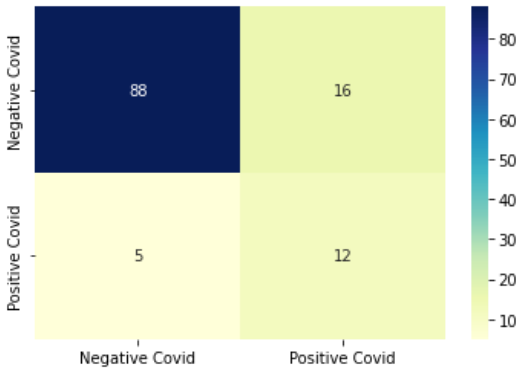
KNN - Imbalanced



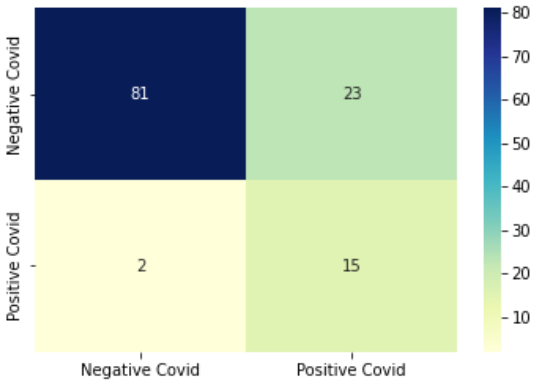
KNN - SMOTE



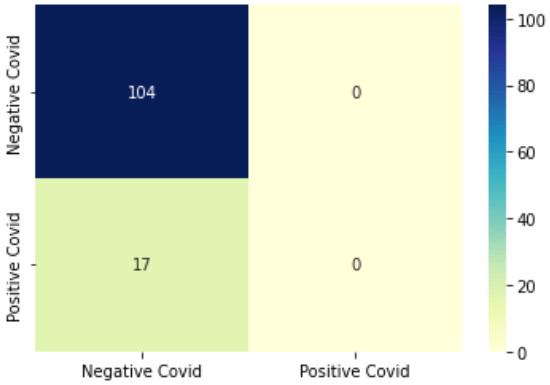
KNN - ROS



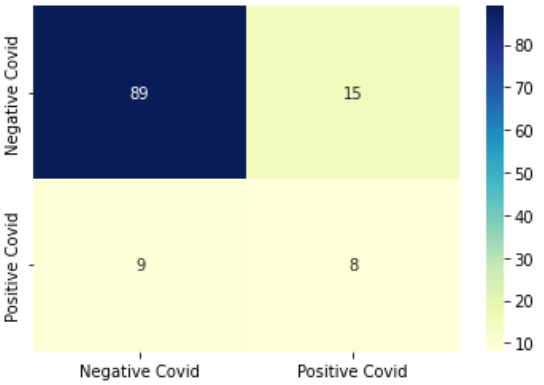
KNN - ADASYN



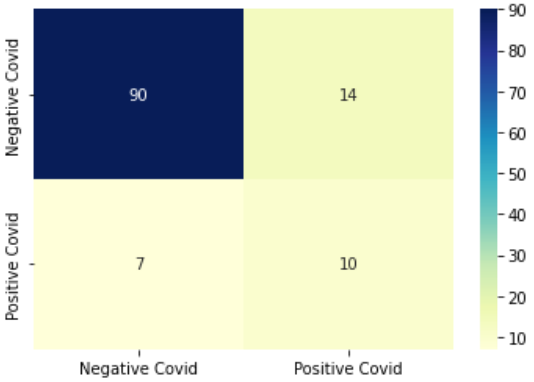
KNN - Borderline SMOTE



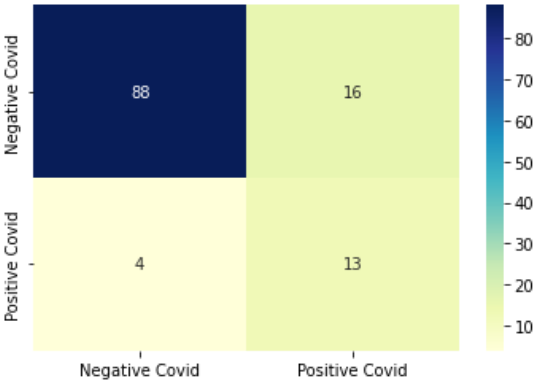
SVM – Imbalanced



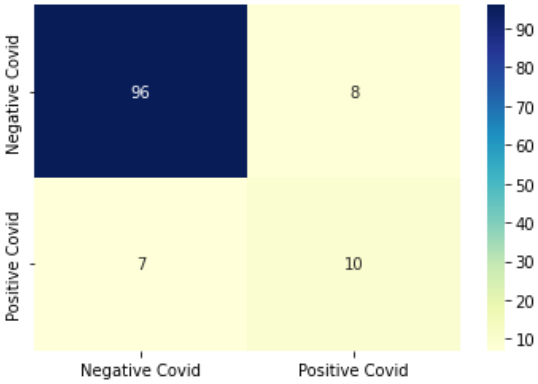
SVM – SMOTE



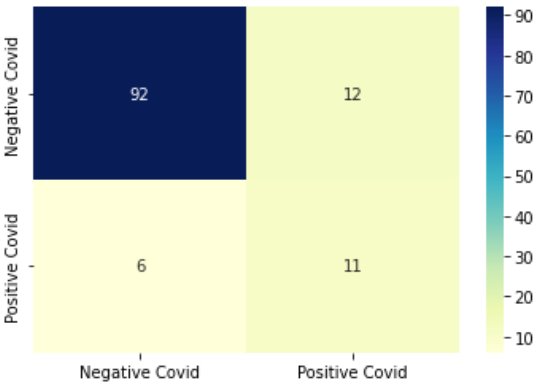
SVM – ROS



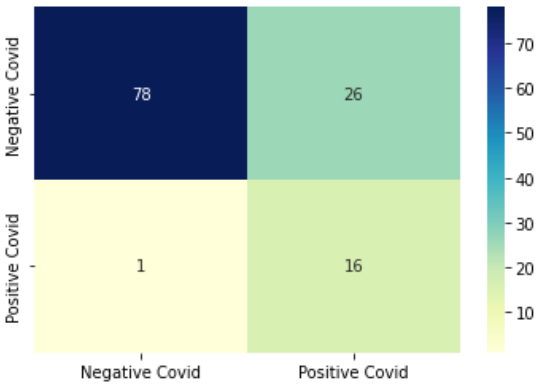
SVM – ADASYN



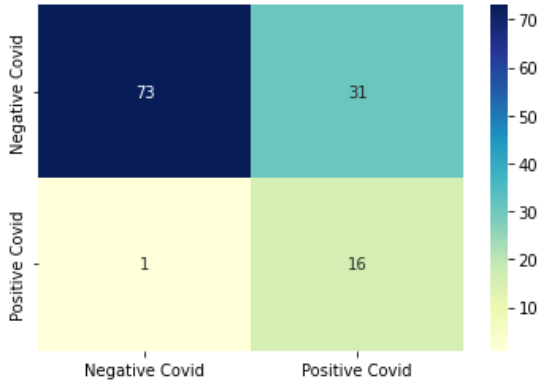
SVM – Borderline SMOTE



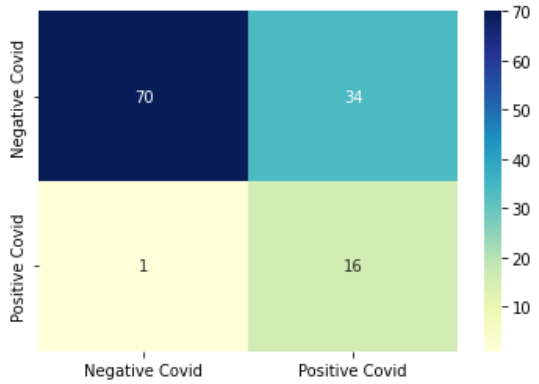
Naïve Bayes – Imbalanced



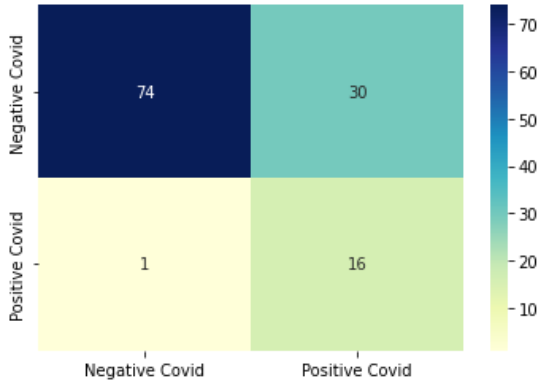
Naïve Bayes – SMOTE



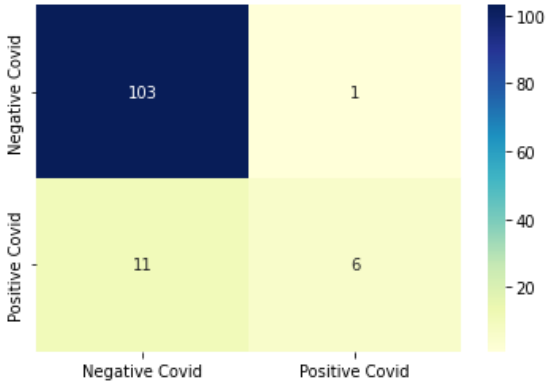
Naïve Bayes – ROS



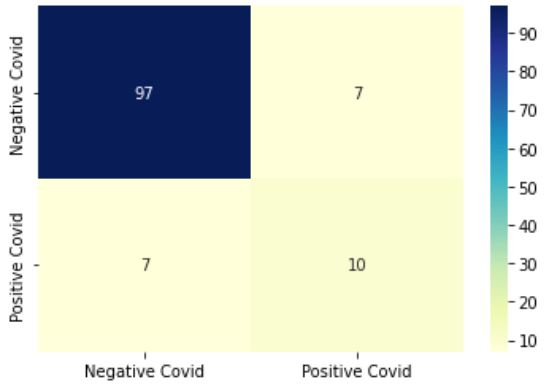
Naïve Bayes – ADASYN



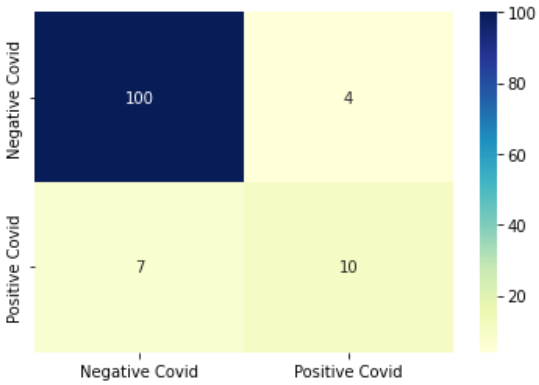
Naïve Bayes – Borderline SMOTE



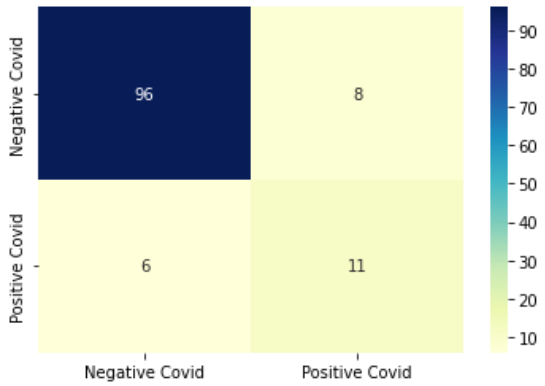
Random Forest – Imbalanced



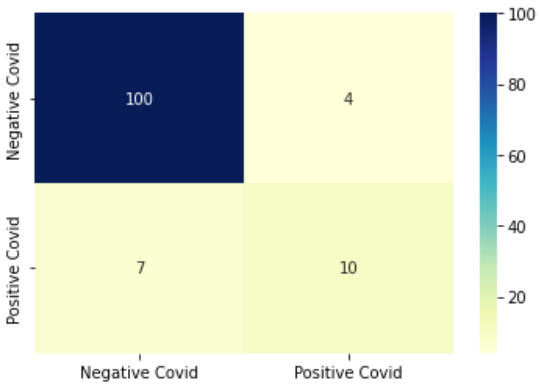
Random Forest – SMOTE



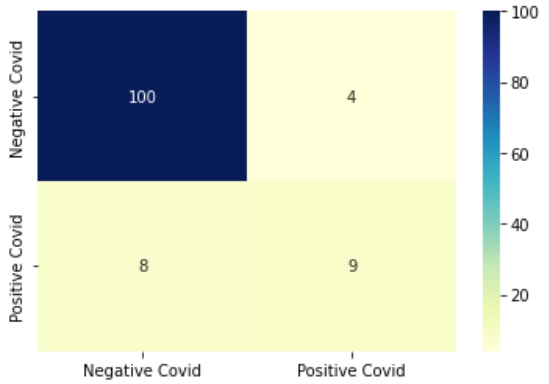
Random Forest – ROS



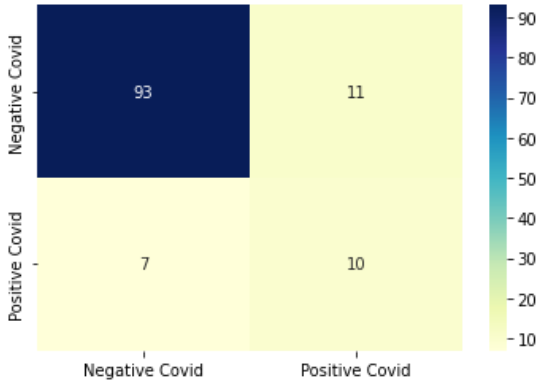
Random Forest – ADASYN



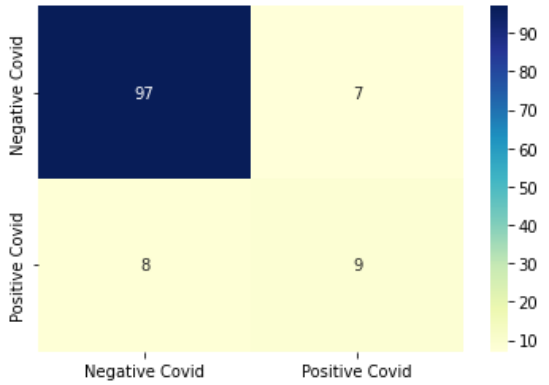
Random Forest – Borderline SMOTE



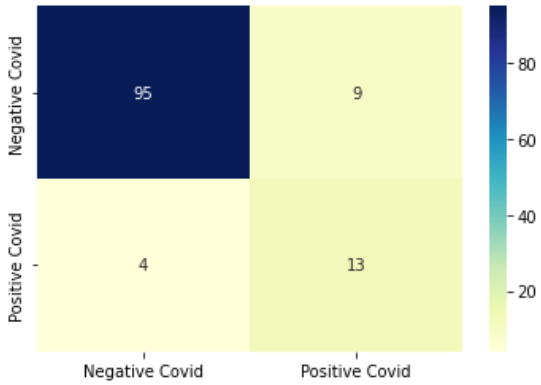
XGboost – Imbalanced



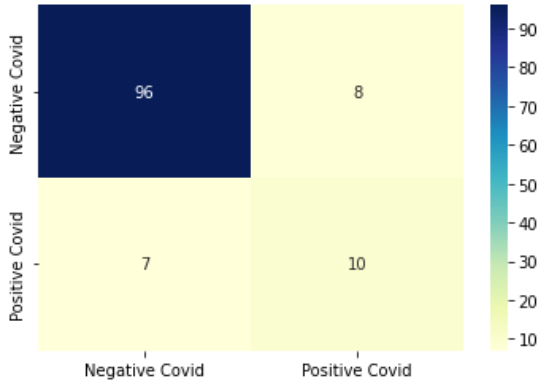
XGboost – SMOTE



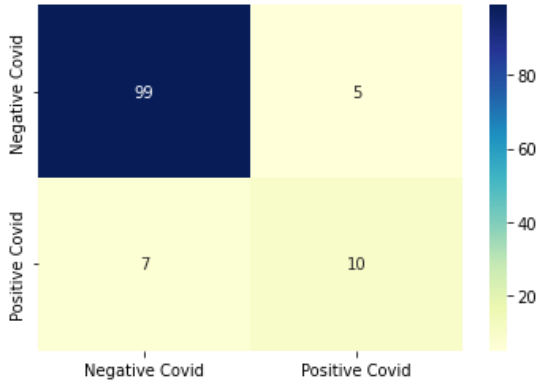
XGboost – ROS



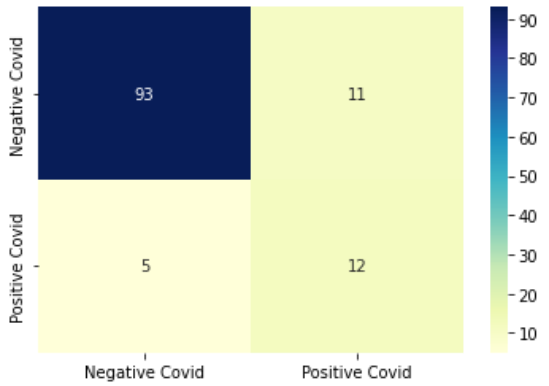
XGboost – ADASYN



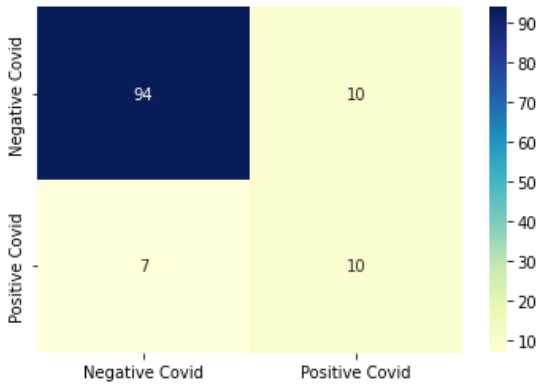
XGboost – Borderline SMOTE



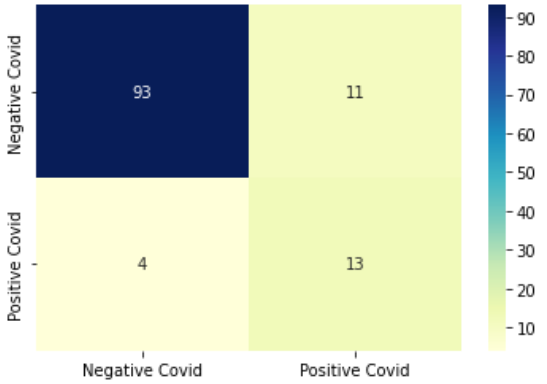
Adaboost – Imbalanced



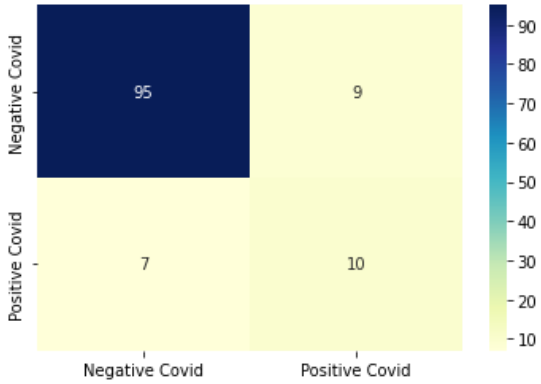
Adaboost – SMOTE



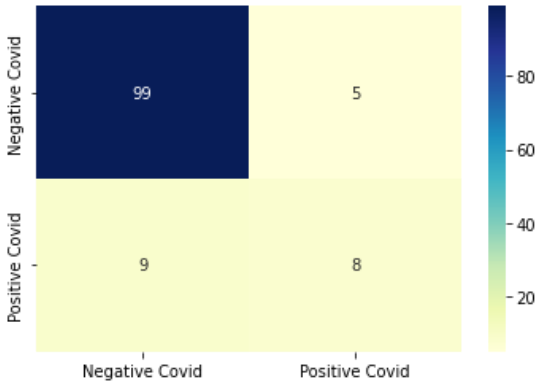
Adaboost – ROS



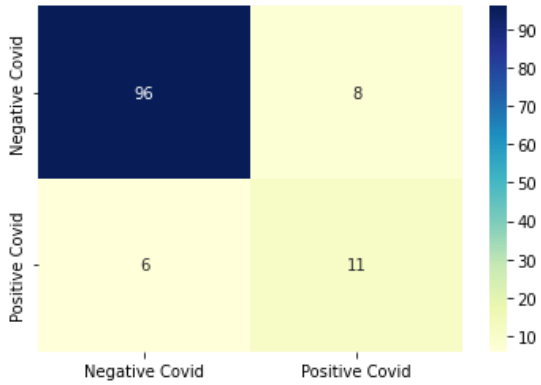
Adaboost – ADASYN



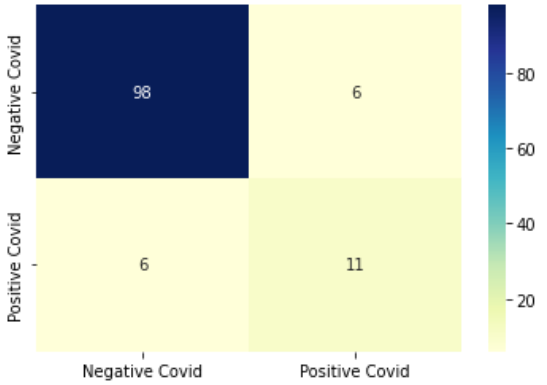
Adaboost – Borderline SMOTE



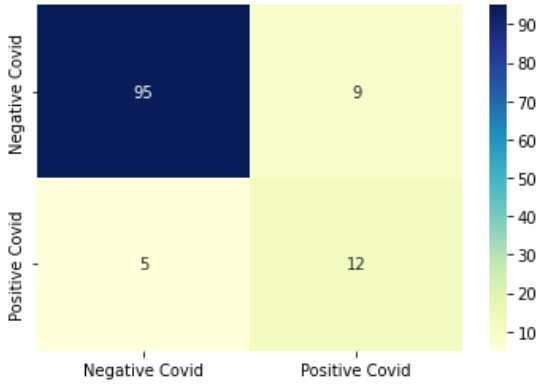
Bagging – Imbalanced



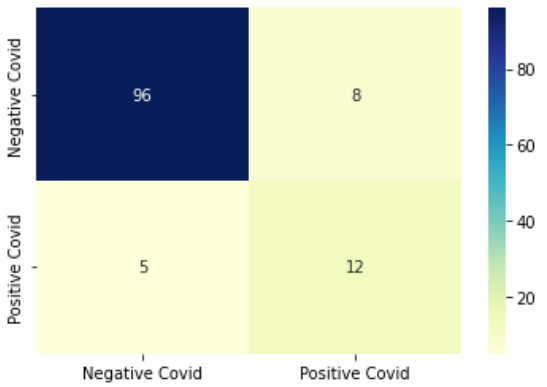
Bagging – SMOTE



Bagging – ROS



Bagging – ADASYN



Bagging – Borderline SMOTE

L.3 Hasil Confusion Matrix Dataset Credit Card Fraud (Seleksi Fitur)

```
Confusion Matrix
[[71068    11]
[   26    97]]
```
KNN - Imbalanced

```
Confusion Matrix
[[70924   155]
[   17   106]]
```
KNN - SMOTE

```
Confusion Matrix
[[71035    44]
[   19   104]]
```
KNN - ROS

```
Confusion Matrix
[[70923   156]
[   17   106]]
```
KNN - ADASYN

```
Confusion Matrix
[[71050    29]
[   20   103]]
```
KNN - Borderline SMOTE

```
Confusion Matrix
[[71077     2]
[   43    80]]
```
SVM – Imbalanced

```
Confusion Matrix
[[69813  1266]
[   16   107]]
```
SVM – SMOTE

```
Confusion Matrix
[[70336   743]
[   14   109]]
```
SVM – ROS

```
Confusion Matrix
[[68886  2193]
[   31    92]]
```
SVM – ADASYN

```
Confusion Matrix
[[70913   166]
[   20   103]]
```
SVM – Borderline SMOTE

```
Confusion Matrix
[[70656   423]
[   24    99]]
```
Naïve Bayes – Imbalanced

```
Confusion Matrix
[[70069  1010]
[   19   104]]
```
Naïve Bayes – SMOTE

```
Confusion Matrix
[[69867  1212]
[   19   104]]
```
Naïve Bayes – ROS

```
Confusion Matrix
[[68149  2930]
[   17   106]]
```
Naïve Bayes – ADASYN

```
Confusion Matrix
[[69294  1785]
[   18   105]]
```
Naïve Bayes – Borderline SMOTE

```
Confusion Matrix
[[71075     4]
[   27    96]]
```
Random Forest – Imbalanced

```
Confusion Matrix
[[71053    26]
[   22   101]]
```
Random Forest – SMOTE

```
Confusion Matrix
[[71071     8]
[   27    96]]
```
Random Forest – ROS

```
Confusion Matrix
[[71053    26]
[   24    99]]
```
Random Forest – ADASYN

```
Confusion Matrix
[[71069    10]
[   26    97]]
```
Random Forest – Borderline SMOTE

```
Confusion Matrix
[[71069    10]
[   26    97]]
```
XGboost – Imbalanced

```
Confusion Matrix
[[70055  1024]
[   12   111]]
```
XGboost – SMOTE

```
Confusion Matrix
[[70557   522]
[   12   111]]
```
XGboost – ROS

```
Confusion Matrix
[[67738  3341]
[   11   112]]
```
XGboost – ADASYN

```
Confusion Matrix
[[70950   129]
[   22   101]]
```
XGboost – Borderline SMOTE

```
Confusion Matrix
[[71059    20]
[   42    81]]
```
Adaboost – Imbalanced

```
Confusion Matrix
[[69214  1865]
[   13   110]]
```
Adaboost – SMOTE

```
Confusion Matrix
[[69580  1499]
[   13   110]]
```
Adaboost – ROS

```
Confusion Matrix
[[65991  5088]
[   11   112]]
```
Adaboost – ADASYN

```
Confusion Matrix
[[70784   295]
[   20   103]]
```
Adaboost – Borderline SMOTE

```
Confusion Matrix
[[71075     4]
[   30    93]]
```
Bagging – Imbalanced

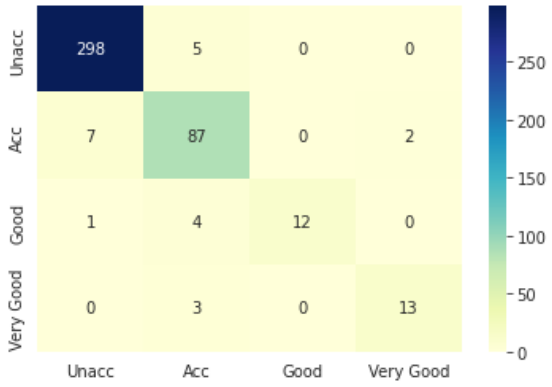
```
Confusion Matrix
[[70990    89]
[   27    96]]
```
Bagging – SMOTE

```
Confusion Matrix
[[71059    20]
[   33    90]]
```
Bagging – ROS

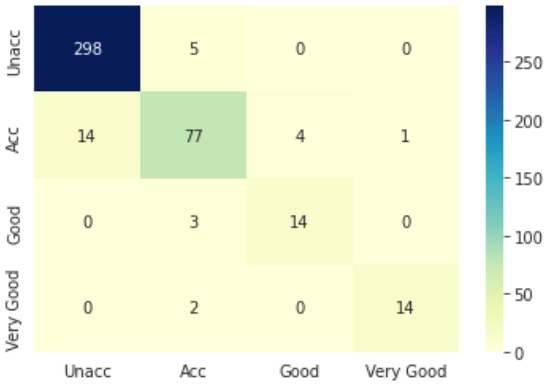
```
Confusion Matrix
[[70985    94]
[   28    95]]
```
Bagging – ADASYN

```
Confusion Matrix
[[71059    20]
[   36    87]]
```
Bagging – Borderline SMOTE

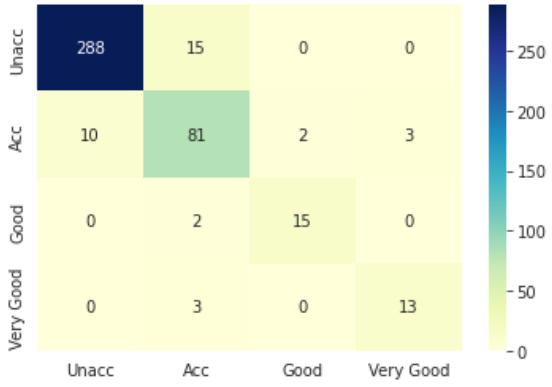
L.2 Hasil Confusion Matrix Dataset Car Evaluation (Tanpa Seleksi Fitur)



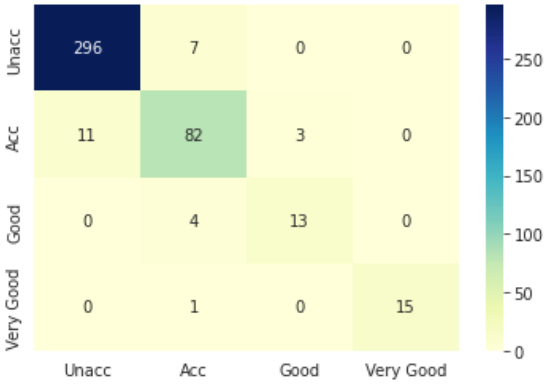
KNN - Imbalanced



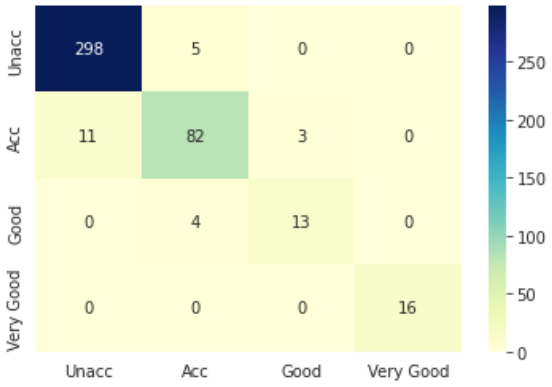
KNN - SMOTE



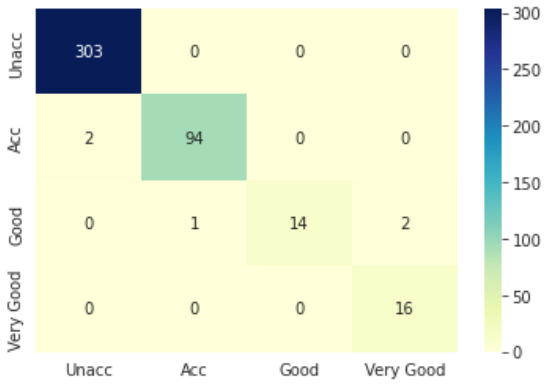
KNN - ROS



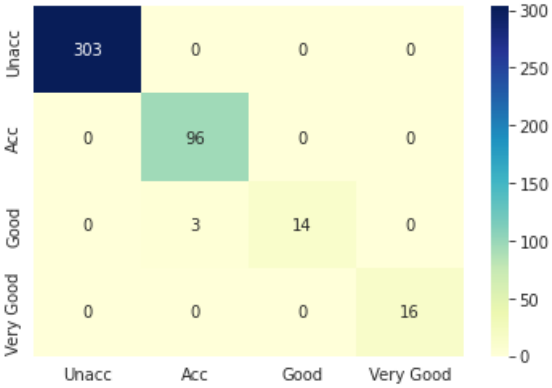
KNN - ADASYN



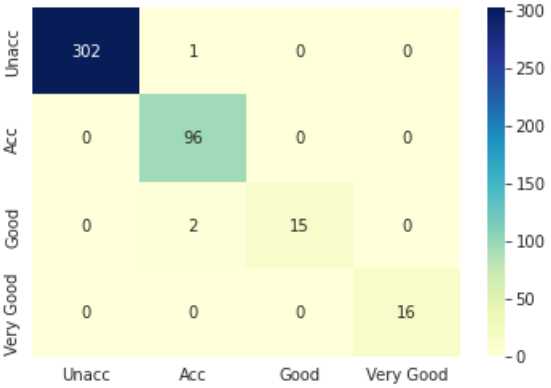
KNN - Borderline SMOTE



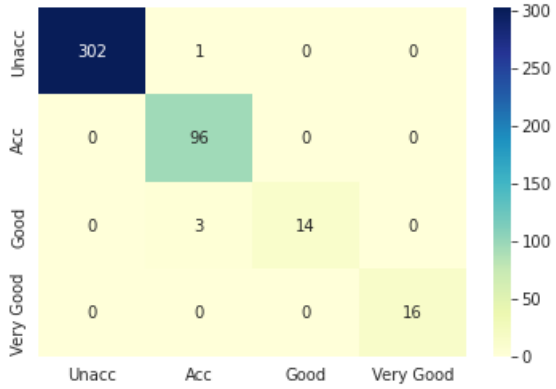
SVM – Imbalanced



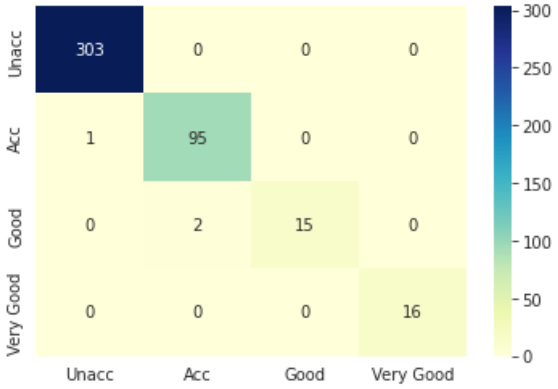
SVM – SMOTE



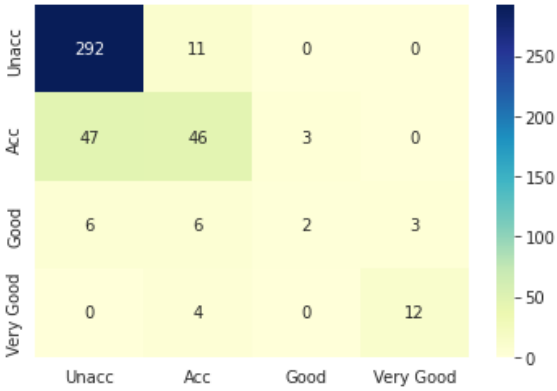
SVM – ROS



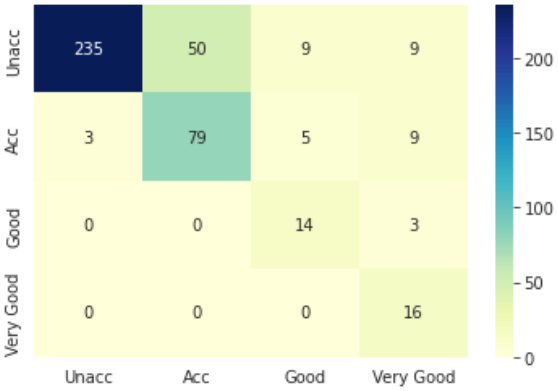
SVM – ADASYN



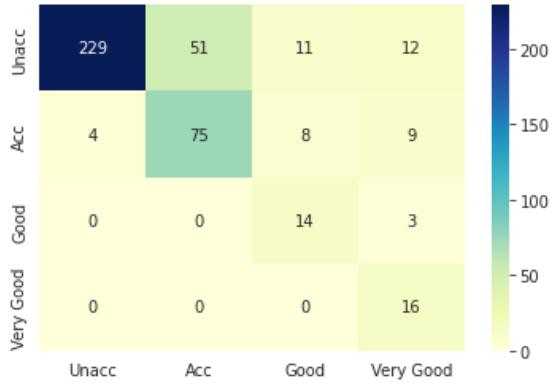
SVM – Borderline SMOTE



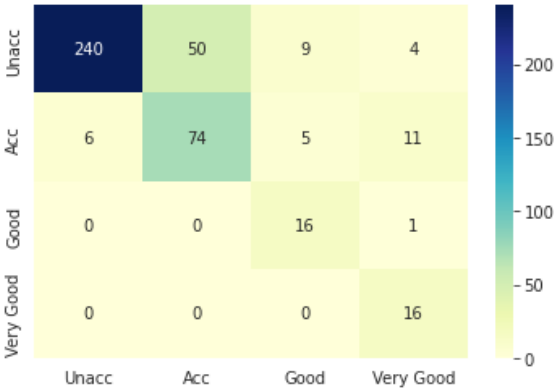
Naïve Bayes – Imbalanced



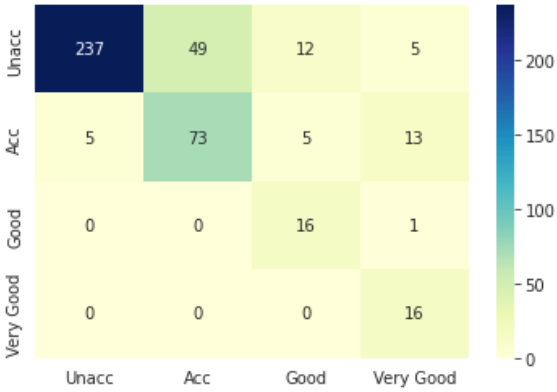
Naïve Bayes – SMOTE



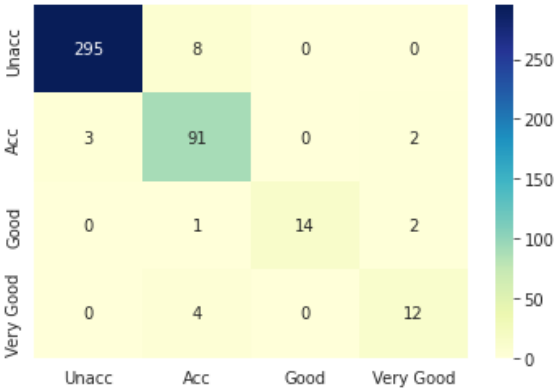
Naïve Bayes – ROS



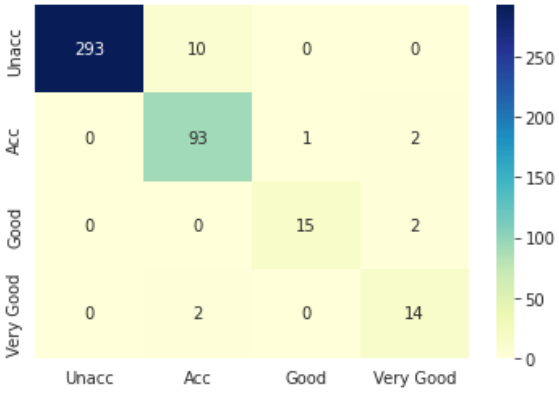
Naïve Bayes – ADASYN



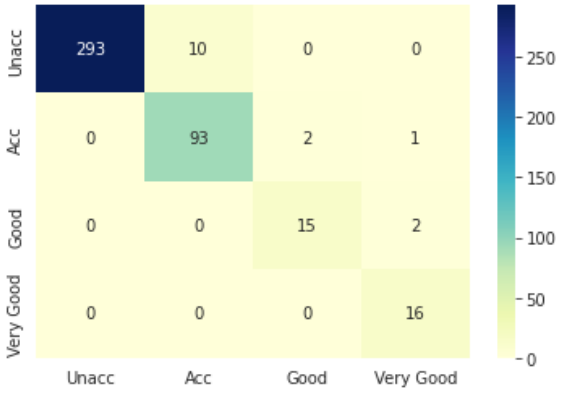
Naïve Bayes – Borderline SMOTE



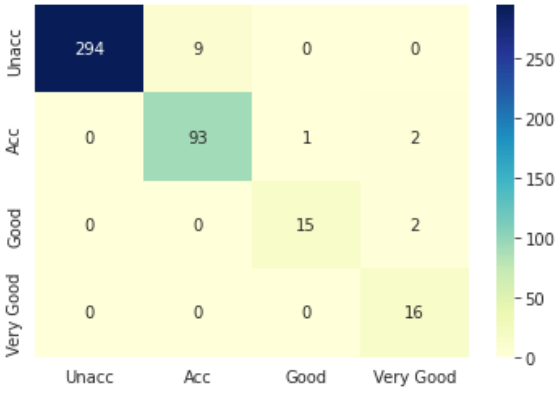
Random Forest – Imbalanced



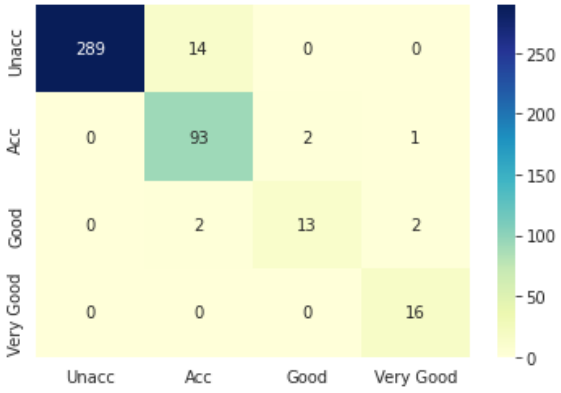
Random Forest – SMOTE



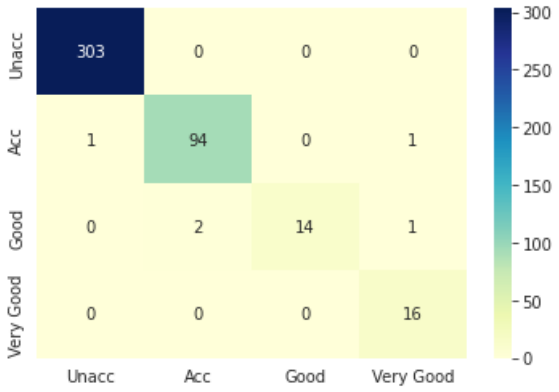
Random Forest – ROS



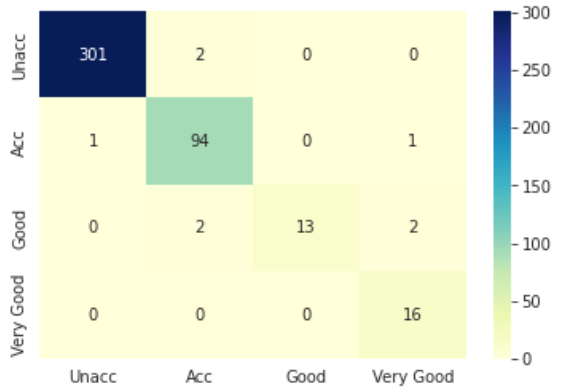
Random Forest – ADASYN



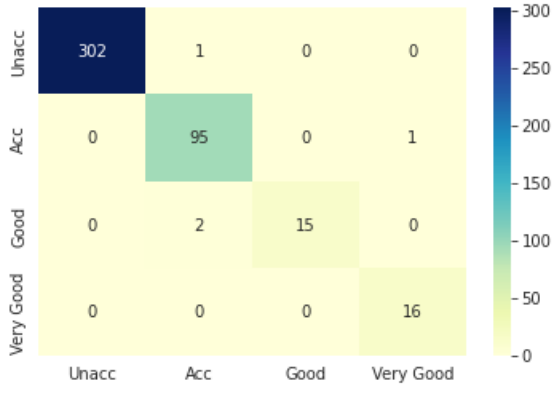
Random Forest – Borderline SMOTE



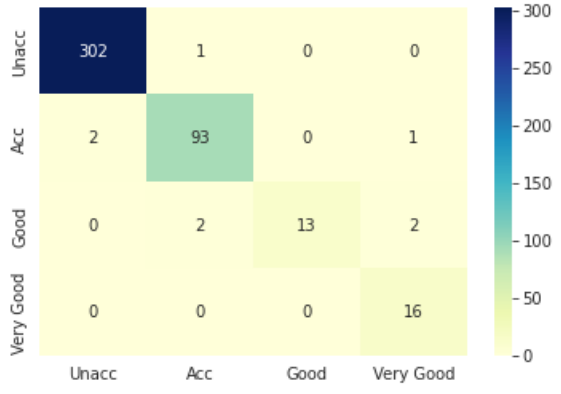
XGboost – Imbalanced



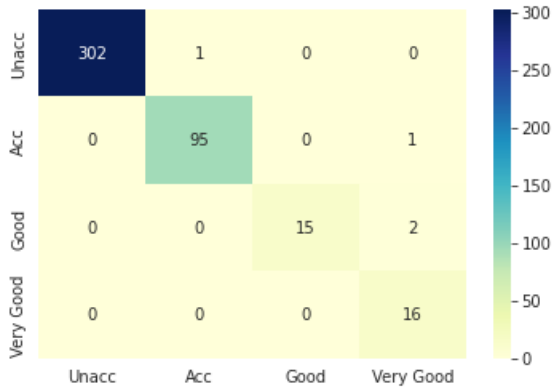
XGboost – SMOTE



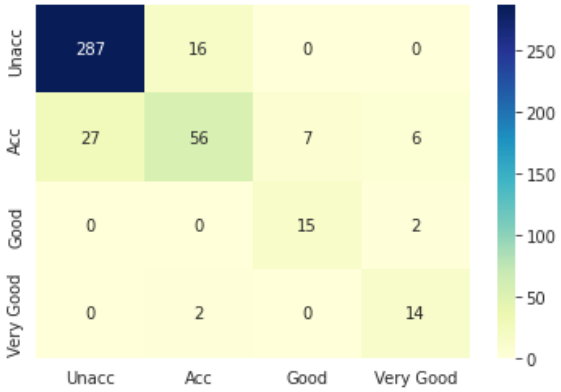
XGboost – ROS



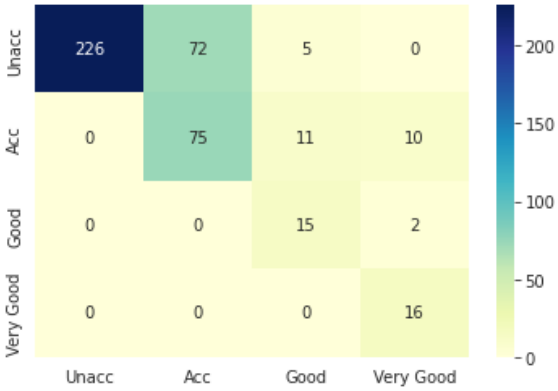
XGboost – ADASYN



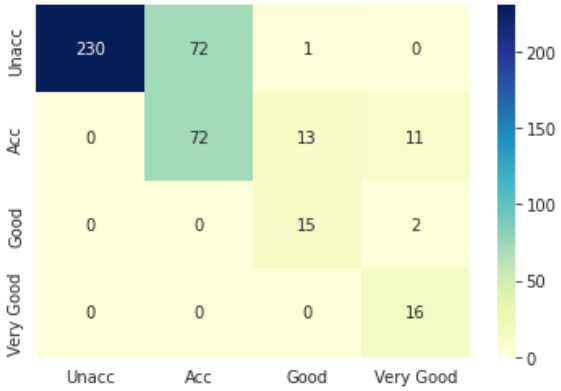
XGboost – Borderline SMOTE



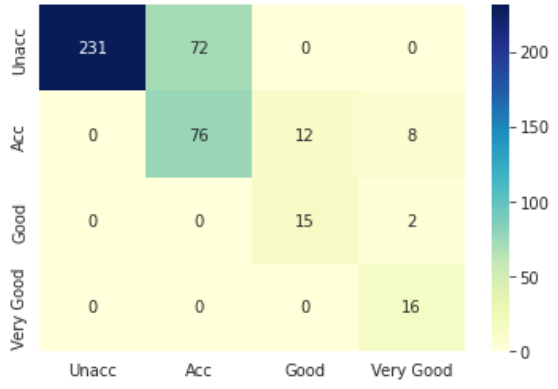
Adaboost – Imbalanced



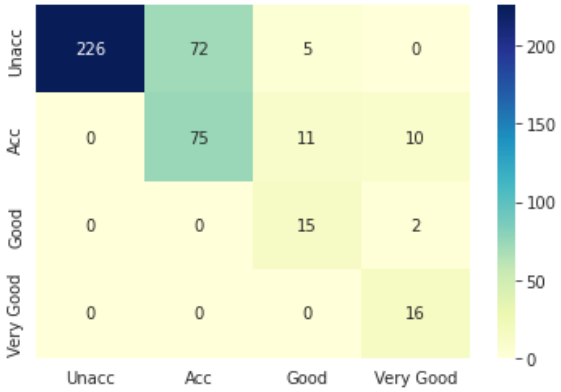
Adaboost – SMOTE



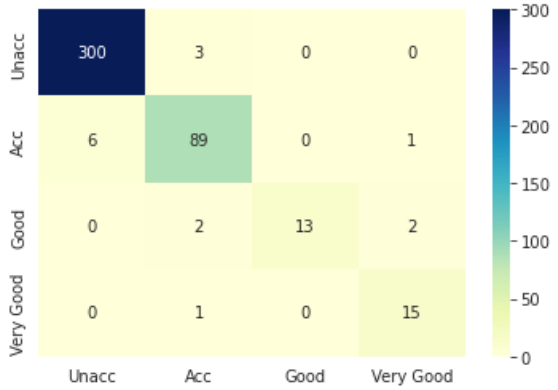
Adaboost – ROS



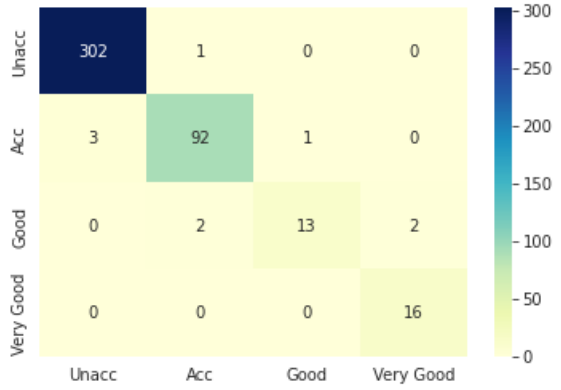
Adaboost – ADASYN



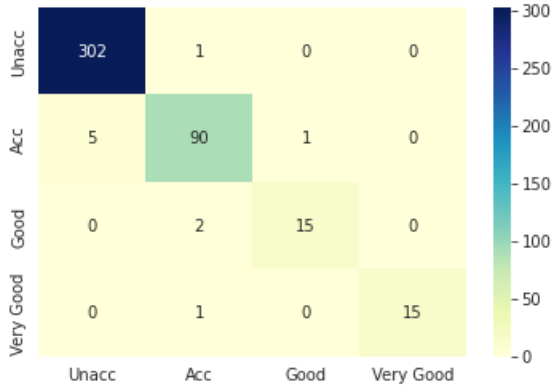
Adaboost – Borderline SMOTE



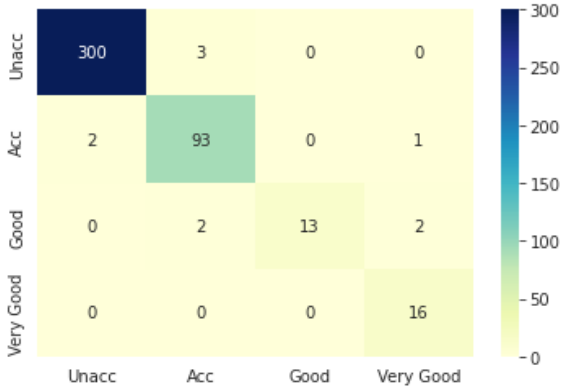
Bagging – Imbalanced



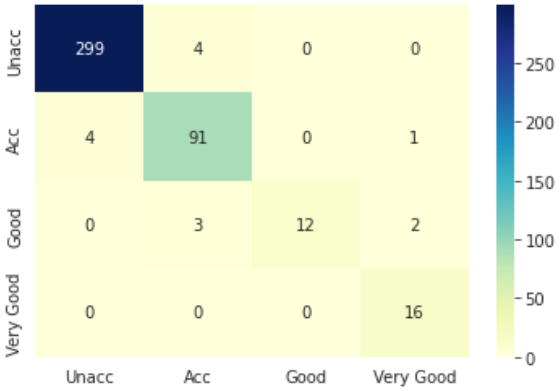
Bagging – SMOTE



Bagging – ROS



Bagging – ADASYN



Bagging – Borderline SMOTE

BIODATA PENULIS

Yulia Niza, lahir di Sukoharjo pada tanggal 31 Juli 2000 merupakan anak keempat dari 4 bersaudara. Penulis telah menempuh pendidikan formal yaitu di SDN Ngasinan 1 (2006-2012), SMPN 1 Bulu (2012-2015) dan SMAN 1 Sukoharjo (2015-2018). Setelah lulus dari SMAN, Penulis mengikuti SNMPTN dan diterima pada program studi Sarjana di Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas (FTEIC), Institut Teknologi Sepuluh Nopember pada tahun 2018 dan terdaftar dengan NRP 05111840000053.

Penulis aktif di organisasi Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) sebagai staf departemen Kesejahteraan Mahasiswa (KESMA) pada tahun 2019 dan diamanahi menjadi Sekretaris Departemen KESMA pada tahun 2020. Selain aktif di organisasi, penulis juga aktif pada kepanitiaan di ITS, yakni menjadi staf Pasar Malam Minggu ITS (PAMMITS) 2019, staf Schematics REEVA tahun 2019, *Instructor Committee* Hubungan Masyarakat Ramadhan Di Kampus (RDK 41) tahun 2020, dan kepanitiaan lainnya. Selain kepanitiaan dan organisasi, penulis juga aktif sebagai Asisten Laboratorium Komputasi Cerdas dan Visi (KCV) pada departemen Teknik Informatika, serta aktif sebagai Asisten Dosen pada beberapa mata kuliah pada Teknik Informatika. Komunikasi dengan penulis dapat melalui *email*: nizayulia@gmail.com.