# ANALISIS METODE OVERSAMPLING SMOTE DAN ADASYN PADA RANDOM FOREST DAN XGBOOST DALAM MENANGANI SISTEM PENDETEKSI PENIPUAN KARTU KREDIT

**TUGAS AKHIR**

Diajukan sebagai syarat menyelesaikan jenjang strata Satu (S-1)
di Program Studi Teknik Informatika, Jurusan Teknologi,
Produksi dan Industri, Institut Teknologi Sumatera


**Oleh:**

**Raihan Alghiffari**

**120140005**


ITERA


**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**INSTITUT TEKNOLOGI SUMATERA**

**LAMPUNG SELATAN**


**2025**

# LEMBAR PENGESAHAN

Tugas Akhir Sarjana dengan judul ”ANALISIS METODE OVERSAMPLING SMOTE DAN ADASYN PADA RANDOM FOREST DAN XGBOOST DALAM MENANGANI SISTEM PENDETEKSI PENIPUAN KARTU KREDIT” adalah benar dibuat oleh saya sendiri dan belum pernah dibuat dan diserahkan sebelumnya, baik sebagian ataupun seluruhnya, baik oleh saya ataupun orang lain, baik di Institut Teknologi Sumatera maupun di institusi pendidikan lainnya.

Lampung Selatan, 24-02-2025
Penulis,

Foto 3x4

Raihan Alghiffari
NIM 120140005

Diperiksa dan disetujui oleh,

| Pembimbing | Tanda Tangan |
|---|---|
| 1. Nama dan Gelar Pembimbing I<br>NIP. 123456789 | ..... |
| 2. Nama dan Gelar Pembimbing II<br>NIP. 123456789 | ..... |

| Penguji | Tanda Tangan |
|---|---|
| 1. Nama dan Gelar Penguji I<br>NIP. 123456789 | ..... |
| 2. Nama dan Gelar Penguji II<br>NIP. 123456789 | ..... |

Disahkan oleh,
Koordinator Program Studi Teknik Informatika
Fakultas Teknologi Industri
Institut Teknologi Sumatera

Andika Setiawan, S.Kom., M.Cs.
NIP 19911127 2022 03 1 007

# HALAMAN PERNYATAAN ORISINALITAS

**Tugas Akhir dengan judul "ANALISIS METODE OVERSAMPLING SMOTE DAN ADASYN PADA RANDOM FOREST DAN XGBOOST DALAM MENANGANI SISTEM PENDETEKSI PENIPUAN KARTU KREDIT" adalah karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan benar.**

**Nama : Raihan Alghiffari**

**NIM : 120140005**

**Tanda Tangan : .....**

**Tanggal : .....**

**HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI**
**TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS**

Sebagai civitas akademik Institut Teknologi Sumatera, saya yang bertanda tangan di bawah ini:

Nama : Raihan Alghiffari
NIM : 120140005
Program Studi : Teknik Informatika
Fakultas : Teknologi Industri
Jenis Karya : Tugas Akhir

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Institut Teknologi Sumatera **Hak Bebas Royalti Noneksklusif** ***(Non-exclusive Royalty Free Right)*** atas karya ilmiah saya yang berjudul:

**ANALISIS METODE OVERSAMPLING SMOTE DAN ADASYN PADA RANDOM FOREST DAN XGBOOST DALAM MENANGANI SISTEM PENDETEKSI PENIPUAN KARTU KREDIT**

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Institut Teknologi Sumatera berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Lampung Selatan
Pada tanggal : 24-02-2025

Yang menyatakan

Raihan Alghiffari

# KATA PENGANTAR

*Pada halaman ini mahasiswa berkesempatan untuk menyatakan terima kasih secara tertulis kepada pembimbing dan pihak lain yang telah memberi bimbingan, nasihat, saran dan kritik, kepada mereka yang telah membantu melakukan penelitian, kepada perorangan atau lembaga yang telah memberi bantuan keuangan, materi dan/atau sarana. Cara menulis kata pengantar beraneka ragam, tetapi hendaknya menggunakan kalimat yang baku. Ucapan terima kasih agar dibuat tidak berlebihan dan dibatasi pada pihak yang terkait secara ilmiah (berhubungan dengan subjek/materi penelitian).*

**Contoh**:

Puji syukur kehadirat Allah SWT atas limpahan rahmat, karunia, serta petunjuk-Nya sehingga penyusunan tugas akhir ini telah terselesaikan dengan baik. Dalam penyusunan tugas akhir ini penulis telah banyak mendapatkan arahan, bantuan, serta dukungan dari berbagai pihak. Oleh karena itu pada kesempatan ini penulis mengucapan terima kasih kepada:

1. [isi dengan nama Rektor ITERA]
2. [isi dengan nama Dekan FTI]
3. [isi dengan nama Kaprodi IF]
4. [isi dengan nama Koordinator TA]
5. [isi dengan nama Dosen Pembimbing]
6. [isi dengan nama Orang Tua, Keluarga, dan kerabat lainnya]
7. [isi dengan nama lain-lain]

Akhir kata penulis berharap semoga tugas akhir ini dapat memberikan manfaat bagi kita semua, amin.

# RINGKASAN

ANALISIS METODE OVERSAMPLING SMOTE DAN ADASYN PADA RANDOM FOREST DAN XGBOOST DALAM MENANGANI SISTEM PENDETEKSI PENIPUAN KARTU KREDIT

Raihan Alghiffari

Halaman Ringkasan berisi uraian singkat tentang latar belakang masalah, rumusan masalah, tujuan, metodologi penelitian, hasil dan analisis data, serta kesimpulan dan saran. Isi ringkasan tidak lebih dari 1500 kata (sekitar 3 halaman).

# ABSTRAK

ANALISIS METODE OVERSAMPLING SMOTE DAN ADASYN PADA RANDOM FOREST DAN XGBOOST DALAM MENANGANI SISTEM PENDETEKSI PENIPUAN KARTU KREDIT

Raihan Alghiffari

Halaman ABSTRAK berisi uraian tentang latar belakang, tujuan, metodologi penelitian, hasil / kesimpulan. Ditulis dalam BAHASA INDONESIA tidak lebih dari 250 kata, dengan jarak antar baris satu spasi.

Pada akhir abstrak ditulis kata “Kata Kunci” yang dicetak tebal, diikuti tanda titik dua dan kata kunci yang tidak lebih dari 5 kata. Kata kunci terdiri dari kata-kata yang khusus menunjukkan dan berkaitan dengan bahan yang diteliti, metode/instrumen yang digunakan, topik penelitian. Kata kunci diketik pada jarak dua spasi dari baris akhir isi abstrak.

**Kata Kunci: Kata Kunci 1, Kata Kunci 2**

# ABSTRACT

Analysis of SMOTE and ADASYN Oversampling Methods in Random Forest and Advanced Gradient Boosting Techniques in Handling Credit Card Fraud Detection Systems

Raihan Alghiffari

Halaman ABSTRACT berisi uraian tentang latar belakang, tujuan, metodologi penelitian, hasil / kesimpulan. Ditulis dalam BAHASA INGGRIS tidak lebih dari 250 kata, dengan jarak antar baris satu spasi. Secara khusus, kata dan kalimat pada halaman ini tidak perlu ditulis dengan huruf miring meskipun menggunakan Bahasa Inggris, kecuali terdapat huruf asing lain yang ditulis dengan huruf miring (misalnya huruf Latin atau Greek, dll).

Pada akhir abstract ditulis kata “Keywords” yang dicetak tebal, diikuti tanda titik dua dan kata kunci yang tidak lebih dari 5 kata. Keywords terdiri dari kata-kata yang khusus menunjukkan dan berkaitan dengan bahan yang diteliti, metode/instrumen yang digunakan, topik penelitian. Keywords diketik pada jarak dua spasi dari baris akhir isi abstrak.

**Keywords: Kata Kunci 1, Kata Kunci 2**

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# DAFTAR PERSAMAAN

# BAB I

# PENDAHULUAN

## 1.1 Latar Belakang

Kartu kredit merupakan alat pembayaran pengganti uang tunai dalam melakukan transaksi. Kemudahan untuk tidak perlu membawa uang tunai meningkatkan penggunaan kartu kredit di seluruh dunia [1]. Namun, di balik balik segala kemudahan dan keuntungan yang manjadikan ketergantungan terhadap penggunaan kartu kredit, penggunaan kartu kredit tidak lepas dari risiko kejahatan, seperti penipuan transaksi [2]. Berdasarkan data dari Experian 60% dari pengguna kartu kredit pernah mengalami penipuan kartu kredit yang mana memberikan kerugian yang besar bagi pengguna kartu kredit dan juga agensi kartu kredit [3] dan bahkan menurut analisis *Merchant Cost Consulting* kerugian dari transaksi penipuan kartu kredit akan terus bertambah dan bisa mencapai 43 miliar dolar pada tahun 2026[4]. Tentu masalah ini akan menjadi masalah yang besar dimasa mendatang jika tidak ditangani dengan baik.

Penipuan kartu kredit bisa berupa penipuan kartu kredit internal dan external, penipuan kartu kredit internal bisa terjadi karena persetujuan antara pemegang kartu dan bank dengan menggunakan identitas palsu untuk melakukan penipuan sedangkan penipuan kartu kredit external melibatkan penggunaan kartu kredit curian untuk mendapatkan uang tunai[5]. Banyak sekali penelitian yang berfokus pada penipuan kartu kredit external dikarenakan menyumbang sebagian besar penipuan kartu kredit[6]. Mendeteksi penipuan kartu kredit secara manual memakan waktu yang banyak dan tidak efisien, disebabkan *big data* yang sudah tidak memungkinankan metode manual bisa dilakukan. yang mana, lembaga keuangan sekarang lebih fokus kepada *computational methodologies* sebagai metode yang lebih efisien dalam mengatasi hal tersebut[7].
Data mining merupakan salah satu *computational methodologies* terbaik untuk mendeteksi penipuan kartu kredit dengan cara membagi menjadi dua kelas yaitu transaksi normal dan transaksi penipuan[8]. Sistem pendeteksi penipuan kartu kredit

berkerja berdasarkan pada analisis perilaku pengguna kartu kredit dalam melakukan transaksi. Banyak teknik atau algoritma yang digunakan untuk mendeteksi penipuan kartu kredit seperti menggunakan neural network[9], genetic algorithm[10], frequent itemset mining[11], support vector machine[12], decision tree[13] dan lain-lain.

Hasil dari teknik atau algoritma yang telah digunakan memberikan hasil yang berbeda-beda. untuk itu, ada beberapa cara metode dan tahap dalam memilih algoritma yang digunakan dalam memilih dalam pengembangan model *machine learning* seperti identifikasi masalah dan melakukan komparasi algoritma yang berbeda-beda untuk kasus yang sama[14].

Masalah utama dalam membuat model pendeteksi penipuan kartu kredit ialah *unbalance data* yang sangat berat antara transaksi penipuan dan transaksi normal yang membuat performa model menjadi sangat buruk dan perlu adanya preprocessing data terlebih dahulu seperti melakukan oversampling agar data yang diinputkan ke machine learningk menjadi *balance* untuk menghasilkan performa terbaik[14].

Hasil riset-riset sebelumnya menunjukan bahwa algoritma *esemble learning* merupakan algoritma terbaik dalam membuat model pendeteksi penipuan kartu kredit[15]. *Esemble learning* merupakan algoritma yang meningkatkan performa prediksi dan dapat mengatasi *complex relationships* di sebuah machine learning dengan memanfaatkan *multiple decision trees* salah satu contoh algoritma esemble learning ialah random forest dan xgboost[16].

Beberapa riset sebelumnya berfokus pada metode komparasi model machine learning guna menentukan model machine learning terbaik dalam membuat model pendeteksi penipuan kartu kredit[14]. Namun, ada hal krusial yang jarang dijadikan fokus dalam penelitian yaitu process metode oversampling untuk mengatasi umbalance data, penggunaan metode oversampling yang berbeda-beda akan menghasilkan hasil performa yang berbeda-beda[17]. Metode oversampling yang digunakan dalam penelitian ini ialah *synthetic minority over-sampling technique*(SMO TE)[18] dan *adaptive synthetic sampling*(ADASYN)[19] dikarenakan metode tersebut memberikan hasil yang selalu berbeda dan tidak saling mengungguli satu sama lain dan menghasilkan hasil yang baik dikasus yang berbeda-beda sesuai dengan pola dataset dan tujuan pembuatan model[20].

Fokus utama dari riset penulis ialah melakukan *comparative analysis* metode oversampling *synthetic minority over-sampling*(SMOTE) dan *adaptive synthetic sampling*(ADASYN) dengan menggunakan algoritma esemble learning seperti random forest dan xgboost dalam membuat model pendeteksi penipuan kartu kredit dengan perbandigan tes *precision*, *recall*, *f1-score*, dan *Matthews's correlation coefficient* (MCC) *metrics*. Dataset yang digunakan ialah dataset *credit card fraud detection* yang dibuat dan dikumpulkan oleh grup machine learning dari *Université Libre de Bruxelles*(ULB)[21]. Dataset ini berisi berisi 284807 transaksi kartu kredit yang dilakukan oleh pemegang kartu di eropa selama dua hari dengan bobot transaksi normal sebanyak 99,83% dan fraud sebanyak 0,17% yang artinya dataset yang digunakan *highly unbalanced*[22]. Riset kali ini akan memperluas penanganan *unbalance data* penipuan kartu kredit untuk menentukan metode oversampling terbaik antara SMOTE dan ADASYN.

## 1.2 Rumusan masalah

Berdasarkan latar belakang yang ada, rumusan masalah dalam penelitian ini sebagai berikut:

1. Bagaimana cara membangun model pendeteksi penipuan kartu kredit dengan menggunakan algoritma Random Forest dan XGBoost dengan metode oversampling SMOTE dan ADASYN?
2. Bagaimana kinerja metode oversampling SMOTE dan ADASYN dalam meningkatkan akurasi model Random Forest dan XGBoost dalam mendeteksi penipuan kartu kredit?

## 1.3 Tujuan

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Merancang dan membangun model pendeteksi penipuan kartu kredit dengan menggunakan algoritma Random Forest dan XGBoost dengan metode oversampling SMOTE dan ADASYN

2. Mengetahui kinerja model dengan oversampling SMOTE dan ADASYN dalam meningkatkan akurasi model Random Forest dan XGBoost dalam mendeteksi penipuan kartu kredit

## 1.4 Batasan Masalah

Batasan-batasan permasalahan dalam penelitian ini mencakup hal-hal berikut:

1. Penelitian ini hanya menggunakan dataset *Credit Card Fraud Detection* dari kaggle dengan pengumpulannya berdasarkan hasil dari *research collaboration* Université Libre de Bruxelles.
2. Penelitian ini hanya akan menganalisis dua metode oversampling, yaitu SMOTE dan ADASYN, tanpa membandingkannya dengan metode oversampling atau undersampling lainnya.
3. Analisis ini akan fokus pada dua algoritma machine learning, yaitu Random Forest dan Advanced Gradient Boosting Techniques seperti XGBoost tanpa mempertimbangkan algoritma lain seperti Support Vector Machine, Neural Networks, atau k-Nearest Neighbors.

## 1.5 Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah sebagai berikut:

1. Mengetahui cara melakukan oversampling dengan metode SMOTE dan ADASYN
2. Mengetahui cara membuat model Random Forest dan Advanced Gradient Boosting Techniques seperti XGBoost.
3. Mengetahui metode oversampling terbaik diantara SMOTE dan ADASYN didalam sebuah algoritma Random Forest dan XGBoost untuk membuat model pendeteksi penipuan kartu kredit.

## 1.6 Sistematika Penulisan

Adapun sistematika penulisan dalam penelitian adalah sebagai berikut:

### 1.6.1 Bab I

Bab ini merangkum beberapa hal yang berkaitan dengan penelitian yakni latar belakang, pengenalan masalah, perumusan pertanyaan penelitian, pendekatan metodologi, sasaran tujuan penelitian, batasan–batasan cakupan penelitian, dampak manfaat hasil penelitian, dan susunan sistematis penulisan.

### 1.6.2 Bab II

Bab ini memfokuskan pada pengulasan tinjauan pustaka yang terkait dengan topik yang meliputi kartu kredit, SMOTE, ADASYN, Random Forest, XGBoost dan serta metode pengujian.

### 1.6.3 Bab III

Bab ini menguraikan tentang metodologi yang akan digunakan dalam penelitian. Ini meliputi pendekatan pembuatan model seperti pembuatan program, pengumpulan dataset, persiapan data, implementasi dan evaluasi terhadap program yang telah dirancang.

### 1.6.4 Bab IV

Bab ini menjelaskan hasil dari penelitian yang telah dilakukan dan berisikan pembahasan dan analisis secara detail tentang kinerja metode oversampling SMOTE dan ADASYN dalam meningkatkan akurasi model pendeteksi penipuan kartu kredit.

### 1.6.5 Bab V

Bab ini berisikan pembahasan tentang kesimpulan berdasarkan hasil dan pembahasan dari penelitian yang telah dilakukan dan juga terdapat saran yang dipaparrkan untuk pengembangan penelitian di masa yang akan datang.

# BAB II

# TINJAUAN PUSTAKA

## 2.1 Tinjauan Pustaka

Pada penelitian ini, peneliti melakukan eksplorasi dan mendapatkan tinjauan pustaka berdasarkan penelitian sebelumnya yang berkaitan dengan pembangunan model pendeteksi penipuan kartu kredit. Peneliti menjadikan penelitian terdahulu sebagai referensi dan landasan perbandingan serta kajian terhadap penelitian yang akan dilakukan. Berikut penelitian-penelitian terdahulu yang berhubungan dengan penelitian ini:

1. Tahun 2021, Asha RB, Suresh Kumar KR, melakukan penelitian terkait dengan *Credit card fraud detection using artificial neural network*. Tujuan Penelitian ini ialah untuk membandingkan hasil dari traditional machine learning seperti SVM dan ANN. Hasil dari penelitian ini ialah menunjukan bahwa ANN lebih baik dibandingkan traditional machine learning berdasarkan accuracy [23].
2. Tahun 2021, Tayebi, Mohammed dan El Kafhali, melakukan penelitian terkait dengan *Hyperparameter optimization using genetic algorithms to detect frauds transactions*. Tujuan penelitian ini bertujuan untuk menangani penipuan transaksi kartu kredit dengan menggunakan *machine learning* untuk mendeteksi penipuan dengan menerapkan *genetic algorithm*(GA) untuk mengoptimalkan hyperparameter dan melakukan komparasi dengan menggunakan *grid search method*. Evaluasi yang digunakan ialah *accuracy*, precision, *recall*, dan *F1 score*[24].
3. Tahun 2022, Putu Tirta Sari Ningsih, Muhammad Gusvarizon, Rudi Hermawan, melakukan penelitian terkait dengan Analisis sistem pendeteksi penipuan transaksi kartu kredit dengan algoritma machine learning. Tujuan penelitian ini ialah melakukan komparasi berbagai algoritma *machine learning* seperti *decision tree*, *random forest*, *logistic regression*, dan *support vector machine*

dalam membuat sistem pendeteksi penipuan transaksi kartu kredit. Evaluasi yang digunakan ialah *accuracy*, *precision*, *recall*, *F-1 score*, *receiver operating characteristics*(ROC), *area Under the Curve*(AUC), dan *matthew coefficient score*(MCC). Hasil penelitian ini membuktikan ensemble learning algoritm seperti random forest merupakan algoritma terbaik dalam membuat sistem pendeteksi penipuan transaksi kartu kredit karena memiliki nilai performa keseluruhan paling baik dengan MCC 87%[15].

4. Tahun 2022, Kumar Sheo, Gunjan Vinit Kumar, Ansari Mohd Dilshad, Pathak, Rashmi, melakukan penelitian terkait dengan *Credit card fraud detection using support vector machine*. Tujuan riset ini ialah menemukan sistem paling efektif dalam mengidentifikasi penipuan kartu kredit dengan menggunakan *support vector machine* dalam menglasifikasi penipuan dan normal transaksi kartu kredit dalam melakukan deteksi.[12].
5. Tahun 2020, Ruttala Sailusha, V. Gnaneswar, R. Ramesh, dan G. Ramakot eswara Raom, melakukan penelitian terkait dengan *Credit card fraud detection using machine learning*. Penelitian ini membandingkan dua algoritma *ensemble learning* yaitu *random forest* dan *adaboost*. Evaluasi yang digunakan ialah *accuracy*, *precision*, *recall*, *F-1 score*. Hasil dari penelitian ini memberikan kesimpulan kalau walaupun banyaknya teknik deteksi penipuan dia tidak bisa bilang kalau algoritma bisa mendeteksi penipuan secara menyeluruh dan hasil komparasinya menunjukan kalau secara akurasi *random forest* dan *adaboost* menunjukan kesamaan namun secara *precision*, *recall*, *F-1 score random forest* lebih unggul. [25].
6. Tahun 2024, Abdul Rehman Khalid, Nsikak Owoh, Omair Uthmani, Moses Ashawa, Jude Osamor, dan John Adejoh, melakukan penelitian terkait *Enhancing credit card fraud detection: an ensemble machine learning approach*. Tujuan penelitian ini melakukan komparasi analisis terkait *ensemble machine learning* seperti *support vector machine random forest* dan *boosting classifiers*. Hasil dari penelitian ini mengaris bawahi bahwa ensemble methods merupakan tool yang terbaik untuk menangani kasus penipuan transaksi. [26]
7. Tahun 2023, Purwar, Archana and Manju, Ms, melakukan penelitian terkait

*Credit card fraud detection using XGBoost for imbalance dataset*. Tujuan dari penelitian ini ialah membuat model pendeteksi transaksi penipuan kartu kredit dengan menggunakan *learning algorithm* xgboost. Hasil penelitian ini menunjukan bahwa xgboost menghasilkan performa model yang baik. [27]

Tabel 2.1: Literasi Penelitian Terdahulu

| **No.** | **Judul** | **Masalah** | **Metode** | **Hasil** |
|---|---|---|---|---|
| 1. | *Credit card fraud detection using artificial neural network* | Penipuan dalam transaksi kartu kredit merupakan hal yang umum saat ini karena sebagian besar dari kita lebih sering menggunakan metode pembayaran kartu kredit. Hal ini disebabkan kemajuan teknologi dan peningkatan transaksi online mengakibatkan penipuan yang menyebabkan kerugian finansial yang sangat besar. | support vector machine (SVM), k-nearest neighbor (KNN), Artificial neural network (ANN) | • SVM:<br>– Accuracy: 0.9349<br>– Precision: 0.9743<br>– Recall: 0.8976<br>• KNN:<br>– Accuracy: 0.9982<br>– Precision: 0.7142<br>– Recall: 0.0393<br>• ANN:<br>– Accuracy: 0.9992<br>– Precision: 0.8115<br>– Recall: 0.7619 |

| | | | | |
|---|---|---|---|---|
| 2. | *Hyperparameter optimization using genetic algorithms to detect frauds transactions* | sekarang servis online berkembang menjadi sangat besar dan maka dengan itu juga penipuan dalam servis online juga menjadi lebih besar, termasuk penipuan traksasi kartu kredit. | SMOTE, grid search (GS) methods, random forest (RF), AdaBoost (AB), logistic regression (LR), decision tree (DT), dan support vector machine (SVM) classifier. | • SVM:<br>– Precision: 95%<br>– Recall: 95%<br>– F1-score: 95%<br>– Accuracy: 94%<br>– Time: 19.039<br>• LR:<br>– Precision: 97%<br>– Recall: 91%<br>– F1-score: 94%<br>– Accuracy: 94%<br>– Time: 104.016<br>• AB:<br>– Precision: 98%<br>– Recall: 93%<br>– F1-score: 95%<br>– Accuracy: 95%<br>– Time: 235.037<br>• RF:<br>– Precision: 96%<br>– Recall: 92%<br>– F1-score: 94%<br>– Accuracy: 94%<br>– Time: 130.300<br>• DT:<br>– Precision: 94%<br>– Recall: 91%<br>– F1-score: 93%<br>– Accuracy: 92%<br>– Time: 12.709 |

| | | | | |
|---|---|---|---|---|
| 3. | Analisis sistem pendeteksi penipuan transaksi kartu kredit dengan algoritma machine learning | Banyaknya volume transaksi dan cepatnya proses transaksi yang berlangsung, membuat tidak mungkin untuk diawasi secara manual oleh manusia karena itu penpuan transaksi menjadi masalah dan harus diselesaikan dengan machine | analisis komparasi model + imbalance, model + SMOTE, model + param + imbalance, model + param + SMOTE, dengan model decision tree (DT), random forest (RF), logistic regression (LR), dan support vector machine (SVM) | Algoritma **Random Forest (RF)** memiliki performa paling baik di antara semua model, disusul oleh model dengan algoritma **Decision Tree (DT)**, hal ini pula yang membuktikan bahwa **ensemble learning adalah teknik terbaik untuk fraud detection**. |

| | | | | |
|---|---|---|---|---|
| 4. | *Credit card fraud detection using support vector machine* | Penipuan kartu kredit tidak diragukan lagi merupakan ekspresi penipuan kriminal. Identifikasi penipuan tampaknya menjadi masalah rumit yang memerlukan banyak keterampilan hingga memasukkan algoritma terkait pembelajaran mesin ke dalamnya. | SVM | Hasil simulasi menunjukan bahwa **SVM lebih baik dibandingkan *state of art approaches***. |

| | | | | |
|---|---|---|---|---|
| 5. | *Credit card fraud detection using machine learning* | Deteksi penipuan kartu kredit saat ini paling banyak dilakukan karena permasalahannya yang sering terjadi di dunia saat ini. Hal ini disebabkan peningkatan transaksi online dan platform e-commerce | Adaboost dan Random Forest | • Random Forest:<br>– Precision: 0.97%<br>– Recall: 89%<br>– F1-score: 0.93<br>– AUC: 94%<br>• Adaboost:<br>– Precision: 89%<br>– Recall: 81,5%<br>– F1-score: 84,5%<br>– AUC: 96% |

| | | | | |
|---|---|---|---|---|
| 6. | *Enhancing credit card fraud detection: an ensemble machine learning approach* | Di era kemajuan digital, meningkatnya penipuan kartu kredit mengharuskan adanya pengembangan sistem deteksi penipuan yang kuat dan efisien. | Membanding kan ensemble learning dan traditional machine learning: Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Random Forest (RF), Bagging, dan Boosting classifiers. | Metode **Ensemble Learning mengungguli model individual**, menunjukkan efektivitas penggabungan beberapa model. |

| | | | | |
|---|---|---|---|---|
| 7. | *Credit card fraud detection using XGBoost for imbalanced data set* | Peningkatan tingkat penipuan di seluruh dunia, berbagai algoritma pembelajaran mesin digunakan oleh analis dan peneliti untuk mendeteksi dan menganalisis penipuan dalam transaksi online. Namun, kumpulan data pelatihan mungkin memiliki beberapa instance dari satu atau lebih instance dari kelas lain dalam kasus klasifikasi biner yang membuat hasilnya menjadi bias. | XGBOOST | • F1 score: 82.78%<br>• Recall: 78.9%<br>• Accuracy: 99.3% |

## 2.2 Dasar Teori

### 2.2.1 Machine Learning

*Machine learning* merupakan sebuah algoritma yang dapat membuat komputer belajar tanpa diprogram secara langsung[28]. Kemampuan untuk belajar dan menentukan pilihan berdasarkan hasil pembelajaran, membuat *Machine Learning* sering disebut sebagai AI atau *Artificial Intelligence*. Namun, kenyataannya itu merupakan sub divisi dari AI dan pada tahun 1970an baru menjadi bagian dari evolusi AI dikarenakan sebelum tahun tersebut peneliti lebih berfokus kepada pendekatan *logical* dan *knowledge-based* dibandingkan algoritma dan juga lambatnya komputer dimasa lalu berdampak dalam performa *machine learning* yang membuat banyak peneliti tidak berfokus meneliti hal tersebut[29]. *Machine Learning* menjadi salah satu algoritma yang sangat penting dalam membantu perkembangan industri dikarenakan kemampuan automasinya yang bisa terus berkembang[30]. *Machine learning* terus membesar dan bisa digunakan di dalam sebuah *e-commerce*, *social media*, medis, bank dan lain-lain[30].

Algoritma *machine learning* dibuat dengan model matematika yang dijalankan di sebuah komputer yang diberikan inputan data atau biasa disebut *training data* untuk membuat sebuah keputusan tanpa secara khusus diprogram langsung oleh *programmer* untuk membuat keputusan[28].

### 2.2.2 Supervised Learning

*Supervised Learning* ialah salah satu tipe *machine learning* di mana algoritma dilatih menggunakan data yang sudah memiliki label atau kelas yang diketahui. *Supervised learning* berlandaskan sebuah *input* data dan memberikan *output* berupa label label seperti pada gambar 2.1[31].


Gambar 2.1: Cara Kerja Supervised Learning

*Supervised learning* dibagi menjadi dua ranah dalam *machine learning* yaitu klasifikasi dan regresi[31]. Algoritma Klasifikasi digunakan untuk membuat prediksi berdasarkan kategori yang ditentukan yang artinya memiliki jumlah kemungkinan *output* yang kecil, contohnya algoritma untuk memprediksi penipuan kartu kredit karena mengklasifikasi 2 kemungkinan yaitu transaksi penipuan dan transaksi normal [31]. Algoritma Regresi merupakan sebuah algoritma yang mendeteksi nilai yang berkelanjutan, contohnya seperti sistem prediksi kenaikan harga saham karena memungkinkan untuk memiliki banyak kemungkinan *output* untuk mengetahui hasil prediksinya[31].

### 2.2.3 *Gradient Descent*

*Gradient descent* merupakan sebuah *optimization algorithm* yang umum digunakan untuk melatih model *machine learning* dan *neural networks*[32]. *Gradient descent* melatih *machine learning* models dengan minimalisasi errors antara hasil prediksi dan nilai asli dengan mengganti parameter w dan b[32].

*Cost function* merupakan cara untuk mengetahui performa dari sebuah model, dengan cara mengurang *output* hasil prediksi dengan hasil sebenarnya untuk mendapatkan nilai akurasi, semakin kecil keluaran nilai dari *cost function* maka akurasi semakin baik[28]. Contohnya dalam model *linear regression* yang mempunyai rumus untuk modelnya dalam bentuk seperti ini $y = wx + b$ yang mana $y = f_{w,b}(x)$ maka formula dari cost function menjadi seperti ini.

$$J(w,b) = \frac{1}{2m}\sum_{i=0}^{m-1}(f_{w,b}(x^{(i)}) - y^{(i)})^2 \tag{2.1}$$

dengan diketahui bahwa:

$J(w, b)$ = *cost function* dengan parameter w dan b

$w$ = slope / turunan

$b$ = *y-axis intercept*

$m$ = jumlah data training

$y$ = nilai sebenarnya

$f_{w,b}$ = nilai prediksi

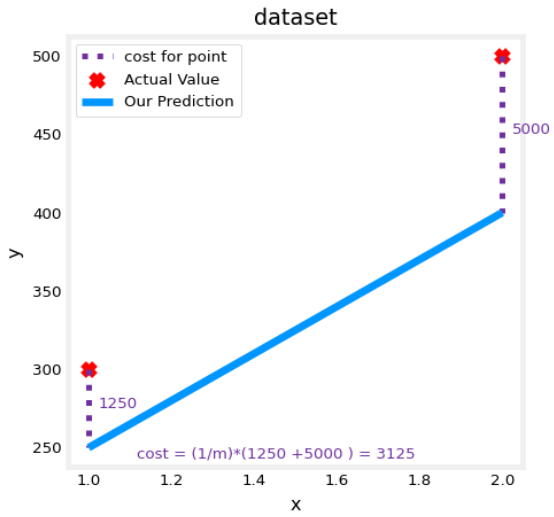
$i$ = nilai iterasi

#### 2.2.3.1 Perhitungan *Cost Function*

Perhitungan *cost function* untuk membuat sebuah model prediksi *linear regression* dengan dataset sederhana dengan jumlah 2 data dengan fitur x dan y. Data pertama x = 1 dan y = 300 dan data kedua x = 2 dan y = 500.

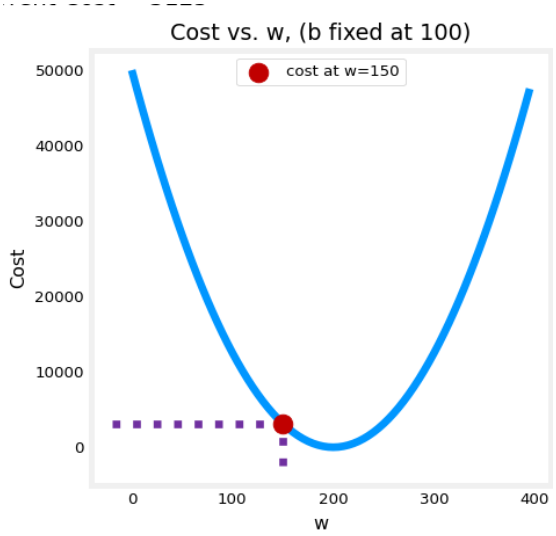
Perhitungan cost function digunakan untuk mengetahui gambaran performa dari sebuah model dengan inputan w dan b tertentu. Contoh inputkan w = 150, b = 100 dan nilai x yang di inputkan ke dalam sebuah model *linear regression* $y = wx + b$ dan setelah itu hitung cost functionnya.

Hasil dari *cost function* w = 150 dan b = 100 adalah 3.125 yang tergambarkan dengan grafik *scatter plot* untuk y dan x seperti pada gambar 2.2.




Gambar 2.2: Scatter Plot Linear Regression

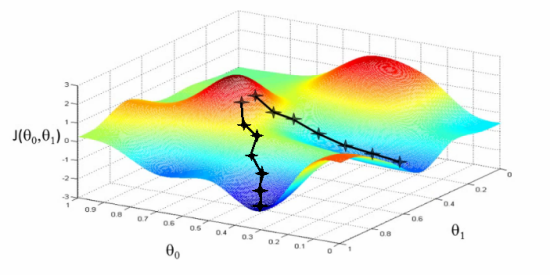
Grafik *scatter plot* tersebut menunjukan bahwa hasil *cost function* w = 150 dan b = 100 belum menghasilkan *output* yang baik dikarenakan *straight line* belum *fit* kedalam data dan bisa dilihat pada gambar 2.3 dan nilai w yang belum mencapai nilai *local minimum*.




Gambar 2.3: Scatter Plot Cost Function

### 2.2.3.2 Cara Kerja *Gradient Descent*

*Gradient descent* merupakan algoritma yang digunakan untuk terus mengubah w dan b untuk mengurangi $j(w, b)$ sampai akhirnya mencapai *local minimum*[32]. Analoginya ialah seperti pendaki yang ingin turun bukit namun banyak kabut disekitarnya sehingga dia tidak bisa melihat contohnya seperti pada gambar 2.4.




Gambar 2.4: Gradient Descent

Seperti yang terlihat pada gambar 2.4. pendaki akan mencoba untuk mencari cara untuk turun ke dasar dengan cara mengecek sekelilingnya dan setelah itu

mencoba untuk melangkah satu kali dan setelah itu mengecek apakah langkah tersebut membuatnya turun atau tidak, jika turun maka pendaki akan mencoba untuk melangkah ke arah yang sama dan jika tidak maka pendaki mencoba langkah yang berbeda dan ada hal perlu diperhatikan pada gambar 2.4 yang menunjukan bahwa local minimum bisa terdapat lebih dari 1[33].

Gradient descent memiliki formula seperti cost function untuk menjalankan sebuah algoritma nya untuk mengganti nilai w dan b agar bisa mencapai *local minimum*.

Formula gradient descent[33]:

$$\text{repeat until convergence: } \begin{cases} w = w - \alpha \dfrac{\partial J(w,b)}{\partial w} \\ b = b - \alpha \dfrac{\partial J(w,b)}{\partial b} \end{cases} \tag{2.2}$$

yang diketahui bahwa:

$$\alpha = \textit{Learning rate}$$
$$\partial = \textit{Partial derivative}$$

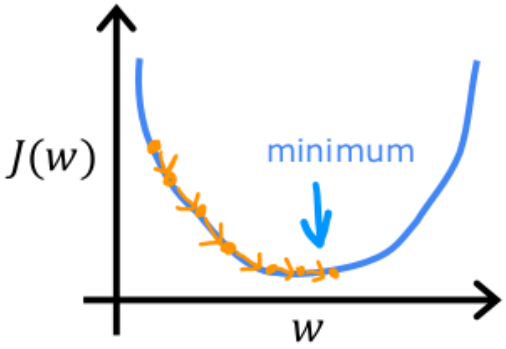
Yang mana parameter w dan b akan *update simultaneously*.

Hasil dari *partial derivative* dari $\frac{\partial J(w,b)}{\partial w}$ dan juga $\frac{\partial J(w,b)}{\partial b}$ seperti berikut.

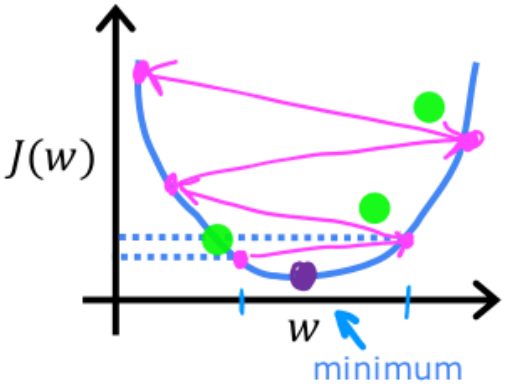
$$\begin{aligned} \frac{\partial J(w,b)}{\partial w} &= \frac{1}{m} \sum_{i=0}^{m-1} \left( f_{w,b}\left(x^{(i)}\right) - y^{(i)} \right) x^{(i)} \\ \frac{\partial J(w,b)}{\partial b} &= \frac{1}{m} \sum_{i=0}^{m-1} \left( f_{w,b}\left(x^{(i)}\right) - y^{(i)} \right) \end{aligned} \tag{2.3}$$

#### 2.2.3.3 Learning Rate

Learning Rate($\alpha$) merupakan salah satu parameter terpenting dalam *machine learning* terkhusus dalam konteks *gradient descent*[34]. *Learning rate* menentukan ukuran langkah yang dapat diambil pada setiap iterasinya[35]. Menentukan *learning rate* merupakan hal yang krusial dikarenakan jika *learning rate* terlalu kecil maka gradient descent akan berjalan sangat lambat seperti pada gambar 2.5 dan apabila nilai learning rate terlalu besar maka bisa terjadi *overshoot* atau tidak pernah mencapai *local minimum* seperti pada gambar 2.6[34].




Gambar 2.5: Learning Rate Terlalu Kecil




Gambar 2.6: Learning Rate Terlalu Besar

Ada berbagai cara untuk menentukan nilai learning rate terbaik salah satu cara paling umum dilakukan ialah dengan menggunakan *grid search*, dengan *grid search* kita bisa melakukan berbagai iterasi dengan nilai learning rate berbeda-beda dan lakukan analsis komparasi untuk menentukan nilai *learning rate* terbaik[36]. Salah satu cara untuk memilih range nilai learning rate yang akan dimasukan ke grid search ialah dengan skala logaritmic contohnya dari $10^{-6}$ sampai $10^{-1}$[36].

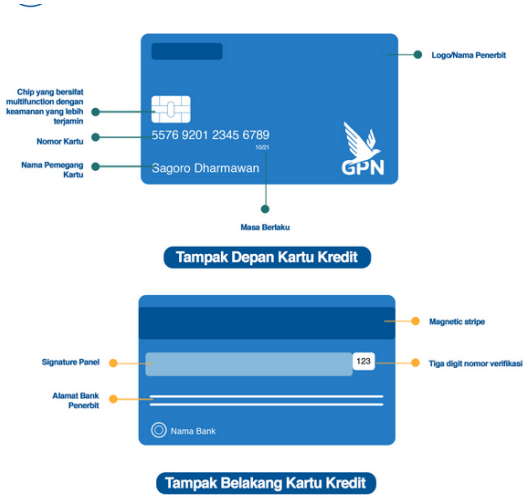
### 2.2.4 Kartu Kredit

Kartu kredit merupakan alat pembayaran nontunai yang memberikan fasilitas kredit kepada pemegang nya[37]. Pemegang kartu kredit wajib membayar tagihan akibat penggunaan kartu tersebut pada waktu yang ditentukan dalam kontrak[37]. Kewajiban pembayaran ini berlaku untuk seluruh transaksi yang dilakukan, termasuk pembelian

barang dan jasa serta penarikan uang tunai[38].

Ada dua jenis transaksi kartu kredit: transaksi *card-present*(CP) dan transaksi *card-not-present*(CNP)[39]. Transaksi *card-present* terjadi ketika kartu kredit secara fisik digunakan di lokasi transaksi, misalnya saat berbelanja di toko dan menggesek kartu di mesin EDC. Sebaliknya, transaksi *card-not-present* tidak memerlukan kehadiran fisik kartu, seperti saat berbelanja online atau melakukan pembayaran melalui telepon dan dalam transaksi CNP, data kartu kredit dimasukkan secara manual atau otomatis ke dalam sistem pembayaran[39]. Berikut merupakan contoh bagian-bagian yang ada pada sebuah kartu kredit.

1. *Chip* kartu kredit terletak di bagian depan kartu. *Chip* ini ditambahkan pada berbagai aplikasi yang dapat mengenkripsi data agar dapat disimpan dengan lebih aman.
2. Nomor kartu adalah 16 digit nomor kartu kredit. Nomor ini tidak akan pernah sama dengan nomor kartu kredit lainnya.
3. *Cardholder Name* adalah nama pemilik atau pemegang kartu kredit.
4. Nama atau logo penerbit adalah nama atau logo perusahaan penerbit kartu.
5. Masa berlaku adalah tanggal dalam format dua digit setelah bulan dan tahun masa berlaku kartu kredit.
6. Logo jaringan kartu (GPN pada diagram) adalah logo jaringan kartu kredit.
7. Terdapat strip magnet di bagian belakang yang dapat dibaca dengan menggesek kartu saat bertransaksi.
8. Tanda tangan pemegang kartu dimasukkan pada kolom tanda tangan.
9. Nomor verifikasi atau CVV/CVC adalah 3 digit angka di sebelah kolom tanda tangan.
10. Alamat bank penerbit adalah alamat perusahaan/bank yang menerbitkan kartu.
11. Nama atau logo penerbit kartu.

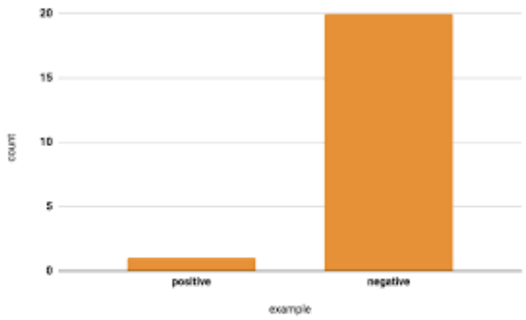



Gambar 2.7: Kartu Kredit

### 2.2.5 *Imbalanced data*

*Imbalanced data* merupakan sebuah kondisi di sebuah data di mana jumlah data antara kelas satu dan lainnya tidak seimbang yang artinya salah satu kelas memiliki jumlah yang lebih besar dibandingkan lainnya[40]. *Imbalanced dataset* kemunculannya merupakan hal alami yang akan selalu muncul pada kejadian yang langka, namun kejadian langka tersebut bisa menyebabkan hal buruk apabila terjadi, salah satu contohnya ialah pada data transaksi penipuan, data penyakit kanker[41]. Kejadian langka tersebut dapat mengakibatkan kerugian besar apabila terjadi dan dibutuhkan sistem yang bisa memprediksi kejadian langka tersebut[41].

*Imbalanced dataset* merupakan masalah yang besar di sebuah *machine learning* dikarenakan dapat mengakitbatkan bias yang cenderung lebih ke kelas mayoritas. hal ini membuat model menjadi sulit untuk mendeteksi kelas dengan jumlah lebih sedikit atau minoritas[40]. Berikut ini merupakan visualisasi *imbalanced data* yang dapat dilihat pada gambar 2.8.




Gambar 2.8: Visualisasi Imbalanced Data

### 2.2.6 *Hyperparameter*

*Hyperparameter* adalah parameter yang ditentukan sebelum proses pelatihan model pembelajaran mesin dimulai[42]. Berbeda dengan parameter model, yang dipelajari secara otomatis selama pelatihan, *hyperparameter* memengaruhi cara model mempelajari dan mengambil keputusan[43]. Contoh *hyperparameter* ialah:

1. *Learning rate*.
2. jumlah iterasi.
3. jumlah *tree* dalam *ensemble methods*.
4. kedalaman *tree* dalam *decision tree*.
5. jumlah *neuron* dalam *neural network*.

*Hyperparameter* berguna dalam membangun strutur dan kompleksitas dari model *machine learning*. Pemilihan *hyperparameter* yang tepat dapat sangat mempengaruhi kinerja model[43]. Cara menentukan *hyperparameter* bisa dilakukan dengan berbagai cara seperti berikut ini.[42]:

1. *Grid Search*
2. *Random Search*
3. *Bayesian Optimization*
4. *Manual Tuning*
5. *Automated Hyperparameter Tuning(seperti Optuna)*

### 2.2.7 Feature Engineering

*Feature engineering* adalah proses penting dalam *machine learning* yang melibatkan pemilihan, modifikasi, atau penciptaan fitur (variabel input) yang membantu sebuah model untuk belajar[28]. Tujuan dari *feature engineering* adalah untuk meningkatkan performa model dengan menyediakan representasi data yang lebih baik[28].

#### 2.2.7.1 Feature Creation

Feature creation merupakan salah satu tipe *feature engineering* yang mana fitur-fitur atau variabel-variabel baru diciptakan berdasarkan data yang sudah ada[28]. Tujuannya adalah untuk menyediakan informasi tambahan yang mungkin tidak langsung tersedia dalam data mentah, tetapi bisa membantu model pembelajaran mesin dalam membuat prediksi yang lebih akurat[44]. Salah satu contoh kegunaan dari feature creation adalah dalam mengenali sebuah ciri-ciri tertentu yang digunakan untuk membuat sistem prediksi, seperti ingin prediksi harga sebuah rumah berdasarkan data panjang dan lebar, dengan data tersebut kita bisa membuat fitur baru seperti luas dengan mengalikan panjang dan lebar. Contoh visualisasi dalam kasus ini ialah saat kita memiliki data yang memiliki fitur panjang dan lebar dan mencoba untuk dimasukan kedalam sebuah model machine learning dan ternyata hasil dari model machine learning belum bisa dikatakan baik seperti pada gambar 2.9. Dalam hal ini kita mungkin harus mencoba membuat feature creation baru agar dapat menghasilkan hasil model machine learning yang lebih baik dengan membuat feature creation luas dan ternyata hasil dari machine learning menjadi lebih baik seperti pada gambar 2.10.

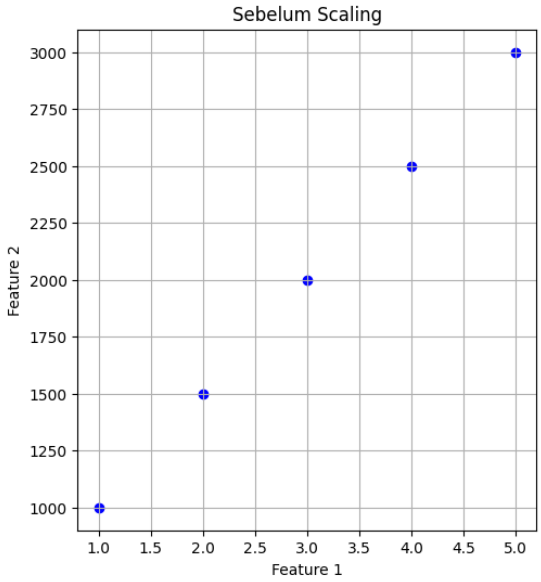

Gambar 2.9: Visualisasi Sebelum Feature Creation


Gambar 2.10: Visualisasi Sesudah Feature Creation

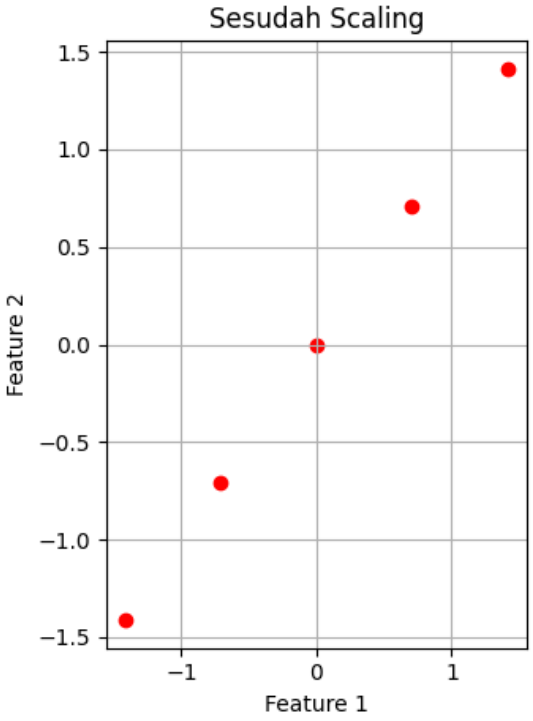
Hasil dari visualiasi diatas menunjukan kalau feature creation dapat meningkatkan kemampuan model dalam belajar namun setelah mengetahui hal tersebut diperlukan pertimbangan agar tidak terjadi overfitting saat melakukan feature creation.

#### 2.2.7.2 Feature Scaling

*Feature scaling* adalah sebuah proses untuk mengubah rentang nilai fitur dalam sebuah *dataset* menjadi seimbang atau dengan skala yang sama[28]. Hasil proses *feature scaling* sangat penting dalam *machine learning* dikarenakan untuk menghindari sensitivitas dalam algoritma yang membuat ada sebuah fitur mendominasi fitur lainnnya[44]. Feature Scaling juga mempengaruhi bagaimana algoritma *gradient descent* bekerja, dengan memiliki skala yang tidak seimbang maka akan membuat *gradient descent* menjadi lebih lama dalam menemukan *local minimum* atau bahkan tidak menemukannya sama sekali[33]. Cara melakukan *feature scaling* dalam python ialah dengan menggunakan *library scikit-learn* terkait *preprocessing*. Hasil dari *feature scaling* ialah membuat rentang nilai antara fitur satu dan lainnya menjadi lebih setara seperti pada gambar 2.11 menunjukan bahwa fitur 2 dan fitur 3 mempunyai rentang nilai yang sangat berbeda, dan setelah melakukan *feature scaling* seperti pada gambar 2.12 menunjukan bahwa rentang nilainya menjadi lebih seimbang.




Gambar 2.11: Visualisasi Sebelum Feature Scaling




Gambar 2.12: Visualisasi Sesudah Feature Scaling

Dari contoh diatas dapat dilihat bahwa rentang nilai dari fitur satu dan fitur dua menjadi lebih seimbang yang artinya membuat machine learning untuk belajar dengan hasil yang lebih baik

### 2.2.8 *Oversampling*

*Oversampling* adalah teknik yang digunakan dalam *machine learning* dan analisis data untuk menangani masalah *unbalanced* data dalam dataset[17]. *Unbalanced data* muncul ketika terdapat satu kelas atau fitur yang memiliki jumlah data sampel yang jauh lebih banyak dibandingkan kelas atau fitur lainnya[21]. Contohnya, dalam *dataset* untuk mendeteksi penipuan kartu kredit, total jumlah transaksi yang normal mungkin jauh lebih banyak daripada jumlah transaksi penipuan.
Tujuan dari *oversampling* ialah untuk membuat data menjadi *balance* atau seimbang agar saat diinputkan ke *machine learning*, model tidak menjadi bias terhadap kelas mayoritas. Salah satu contoh dari metode *oversampling ialah* SMOTE[18] dan ADASYN[19].

### 2.2.9 SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE atau bisa disebut *Synthetic Minority Over-sampling technique* adalah sebuah metode yang digunakan untuk mengatasi masalah ketidaksemibangan (imbalance) kelas dalam dataset[18]. SMOTE digunakan untuk tugas klasifikasi. SMOTE bekerja dengan membuat sampel sintetis dari kelas minoritas untuk meningkatkan jumlah sampel kelas minoritas sehingga dataset menjadi lebih seimbang dengan kelas mayoritas[18]. SMOTE tidak menduplikasi sample yang sudah ada tapi SMOTE menghasilkan sampel baru dengan interpolasi sample minoritas[18].

#### 2.2.9.1 Cara Kerja SMOTE

Berikut cara kerja dibalik SMOTE:

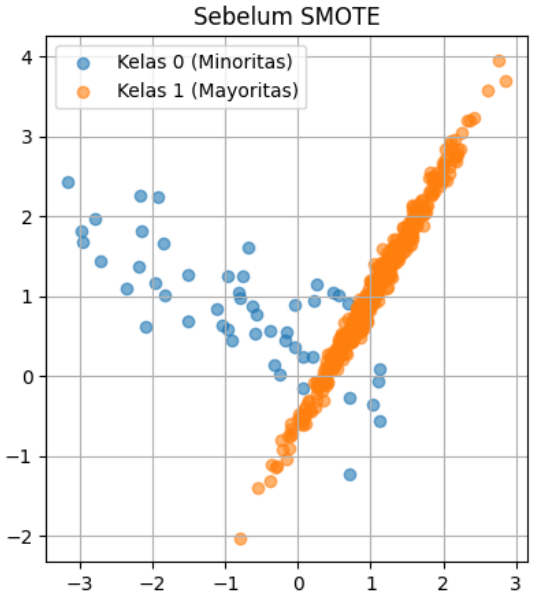
1. Pemilihan sampel pada fitur minoritas dengan contohnya sampel x pada suatu fitur minoritas
2. Cari *k-nearest neighbors* (tetangga terdekat) dari sampel x didalam fitur minoritas.
3. Pilih secara acak salah satu dari *k-nearest neighbors* yang kita sebut sebagai $x_{neighbor}$
4. Buat sampel baru dengan linear interpolasi antara sampel x dan $x_{neighbor}$ yang

kita sebut $x_{synthetic}$ dengan perhitungan sebagai berikut:

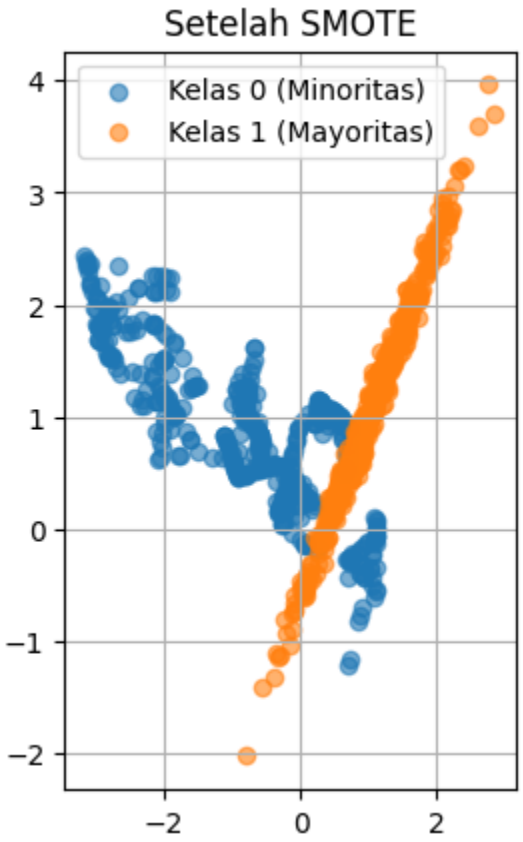
$$x_{synthetic} = x + \lambda.(x_{neighbor} - x) \tag{2.4}$$

yang mana $\lambda$ merupakan bilangan acak antara 0 dan 1.

Hasil dari metode oversampling SMOTE akan menghasilkan pola data yang sebelumnya merupakan seperti ini pada gambar 2.13 menjadi seperti ini pada gambar 2.14. Pada gambar 2.14 dapat dilihat kalau metode oversampling SMOTE membuat data sample berdasarkan pola dari data sebelumnya dengan membuat garis interpolasi antar data poin terdekatnya yang mana menghasilkan data sintetis yang bnyk namun sama dengan pola data sebelumnya.




Gambar 2.13: Visualisasi Data Sebelum SMOTE




Gambar 2.14: Visualisasi Data Sesudah SMOTE

### 2.2.10 ADASYN (Adaptive Synthetic Sampling)

ADASYN (*Adaptive Synthetic Sampling*) adalah teknik yang digunakan untuk menangani ketidakseimbangan(imbalance) kelas dalam tugas klasifikasi, namun dengan pendekatan yang lebih adaptif dibandingkan metode seperti SMOTE[19]. ADASYN bekerja dengan membuat sampel sintetis baru untuk kelas minoritas. ADASYN berfokus pada sampel yang sulit untuk diklasifikasikan yang terletak di dekat perbatasan antara kelas mayoritas dan minoritas[19].

#### 2.2.10.1 Cara kerja ADASYN

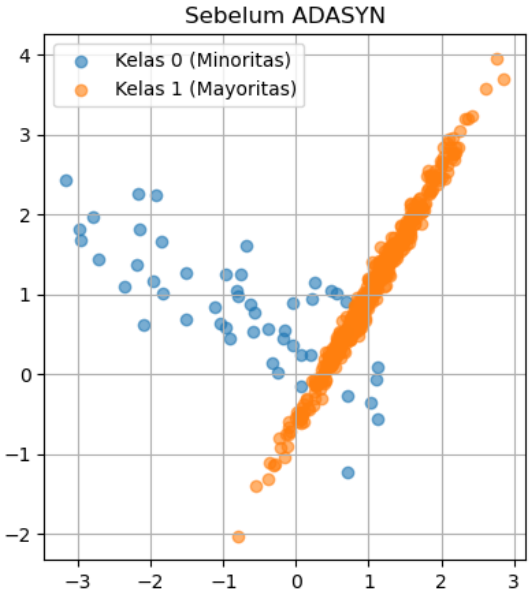
Berikut cara kerja dibalik ADASYN:

1. Yang pertama ialah tentukan jumlah sampel sintetis yang perlu dibuat untuk setiap sampel minoritas $x_i$ berdasarkan tingkat kesulitan klasifikasi lokal. Tingkat kesulitan lokal $d_i$ dihitung berdasarkan jumlah tetangga terdekat dari kelas

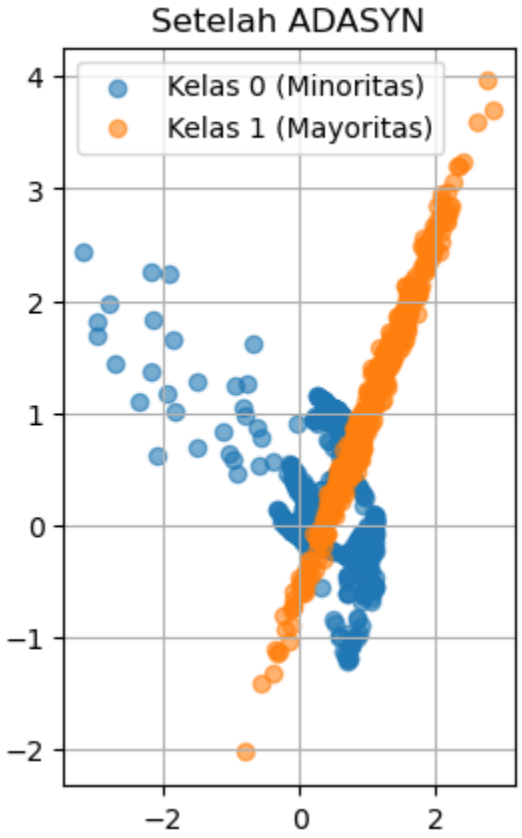
mayoritas di sekitar $x_i$.

2. Yang kedua hitung bobot adaptif atau $r_i$ untuk setiap sampel minoritas $x_i$ yang mana $n_{min}$ adalah jumlah sampel kelas minoritas. Menghitung bobot adaptif ini digunakan untuk menentukan seberapa banyak sampel sintetis yang perlu dibuat untuk setiap sampel minoritas.
3. Yang terakhir ialah membuat sampel sintetis berdasarkan jumlah total sampel sintetis yang diinginkan. sampel sintetis dibuat dengan menggunakan rumus interpolasi linear SMOTE seperti pada rumus 2.4.

Hasil dari metode oversampling ADASYN akan menghasilkan pola data yang berbeda dengan sebelumnya seperti pada gambar 2.15 menjadi seperti ini pada gambar 2.16. Pada gambar 2.16 dapat dilihat kalau metode oversampling ADASYN membuat data sintetis berdasarkan bobot yang sudah ditentukan sebelumnya dan setalh itu membuat garis interpolasi antar data poin terdekatnya yang mana menghasilkan data sintetis yang banyak namun memiliki data yang cenderung berkumpul pada pertemuan antara data minortas dan mayoritas.



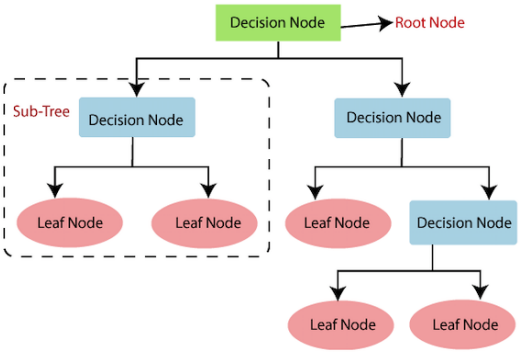

Gambar 2.15: Visualisasi Data Sebelum ADASYN




Gambar 2.16: Visualisasi Data Sesudah ADASYN

### 2.2.11 *Decision Tree*

*Decision Tree* merupakan sebuah algoritma machine learning yang digunakan untuk tugas klasifikasi dan regresi dengan memprediksi berdasarkan serangkaian kondisi atau pertanyaan tertentu[45]. *Decision Tree* sering di visualisasikan dengan gambar pohon terbaik dengan bagian paling atas disebut *root node* dan setiap *node* dibawah *root node* yang ditunjuk oleh panah dan menunjuk panah bisa disebut dengan *decision node atau branches* dan untuk *node* yang hanya ditunjuk dan tidak menunjuk *node* disebut *leaf node*[46]. Visualisasinya dapat dilihat pada gambar 2.17.




Gambar 2.17: Decision Tree

Cara *decision tree* membuat keputusan ialah dengan dengan melakukan perhitungan pada setiap node nya untuk melanjutkan keputusan dengan cara menghitung *gini impurity*, impurity berarti dalam *decision tree* merupakan campuran dari keputusan iya atau tidak dan setelah menghitung gini impurity pada setiap keputusan iya dan tidak, kita hitung *total gini impurity* untuk menjadi ukuran kualitas dari prediksi[47]. Berikut merupakan rumus *gini impurity*:

$$Gini(D) = 1 - \sum_{i=1}^{k} [p(i)]^2 \tag{2.5}$$

Yang dapat kita ketahui bahwa:

$D$ = merupakan node.

$k$ = merupakan jumlah kelas

$p(i)$ = merupakan peluang sampel yang termasuk kelas i pada node.

Berikut merupakan rumus dari *total gini impurity*:

$$Total\ Gini\ Impurity = \sum_{i=1}^{n} \left[ \frac{|D_i|}{|D|} \cdot Gini(D_i) \right] \tag{2.6}$$

Yang mana dapat kita ketahui bahwa:

$n$ = merupakan jumlah *node* pada sebuah *tree*

$D_i$ = merupakan *dataset* pada *node* i

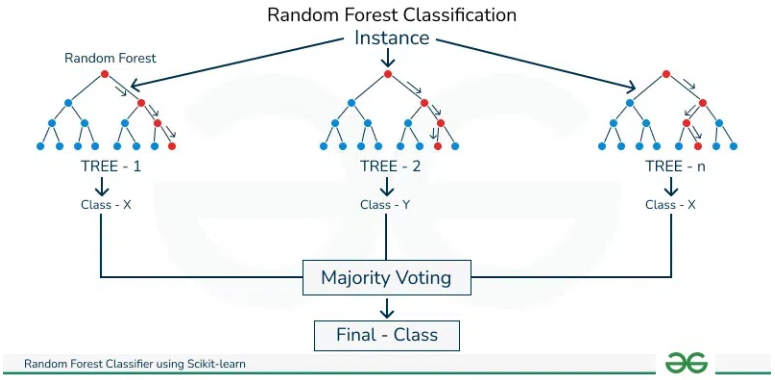
$D$ = merupakan jumlah total sampel dalam *dataset*

### 2.2.12 Random Forest

*Random Forest* merupakan algoritma yang dibuat oleh Leo Breiman yang mana pembuatan random forest ini merupakan lanjutan dari ide paper dia sebelumnya terkait bagging[48], [49]. *Random forest* merupakan salah satu algoritma ensemble learning bertipe *bagging* yang digunakan untuk klasifikasi dan regresi. Algoritma ini bekerja dengan menggabungkan beberapa *decision tree* untuk menghasilkan model yang lebih baik dalam menghindari *overfitting*[48]. Setiap tree dalam *random forest* dilatih dengan menggunakan subset data yang berbeda yang kita sebut sebagai *boostrapped dataset* yang mana cara kerjanya cukup sederhana hanya memilih secara acak data yang ada pada original dataset dengan aturan:

1. Saat pemilihan acak boleh memilih data yang sama yang artinya apabila saat memilih data secara acak itu terpilih data yang sama maka itu diperbolehkan.
2. Ukuran dari *boostrapped dataset* itu harus sama dengan original dataset.

Setelah membuat *boostrapped dataset* akan dilakukan pembuatan decision tree dengan menggunakan boostrapped dataset, dalam pembuatan decision tree kita perlu menentukan berapa *max feature* yang kita pilih secara acak pada setiap *split tree*.
setelah pembuatan tree selesai dilakukan maka kita mengulang proses tersebut hingga ratusan kali atau sesuai yang kita yang tentukan dan keputusan akhir dibuat dengan menggabungkan hasil dari semua *tree* sepeti pada gambar 2.18.




Gambar 2.18: Random Forest

*Random forest* memikiki banyak *hyperparameter* yang digunakan untuk mengatur bagaimana random forest bekerja. *Hyperparameter* yang digunakan dalam *random forest* ialah sebagai berikut.

1. n_estimator digunakan untuk menentukan seberapa banyak *tree* yang ingin digunakan pada model *random forest*.
2. max_depth digunakan untuk menentukan maksimum kedalaman *tree*.
3. min_sample_split digunakan untuk menentukan jumlah minimum yang dibutuhkan untuk *split* pada *internal node*.
4. min_sample_leaf digunakan untuk menentukan jumlah *minimum* yang dibutuhkan untuk membuat *leaf node*.
5. max_feature digunakan untuk menentukan seberapa banyak *feature* yang dibutuhkan pada setiap *split*.
6. bootstrap digunakan untuk menentukan apakah menggunakan *boostrap sampling* atau tidak untuk membuat *training set* pada setiap *tree*.

### 2.2.13 XGBoost

Extreme Gradient Boosting atau XGBoost adalah salah satu ensemble learning bertipe boosting dan merupakan sebuah algoritma *machine learning* yang dioptimalkan untuk implementasi metode *gradient boosting*[50]. Algoritma ini dikenal karena kecepatannya, efisiensinya, dan kemampuannya menghasilkan model yang sangat akurat, sehingga sering digunakan dalam kompetisi pembelajaran mesin seperti yang diadakan di platform Kaggle. *XGBoost* digunakan untuk menyelesaikan masalah klasifikasi dan regresi[51].
*XGBoost* memikiki banyak *hyperparameter* yang digunakan untuk mengatur bagaimana xgboost bekerja. *Hyperparameter* yang digunakan dalam *xgboost* ialah sebagai berikut[52]:

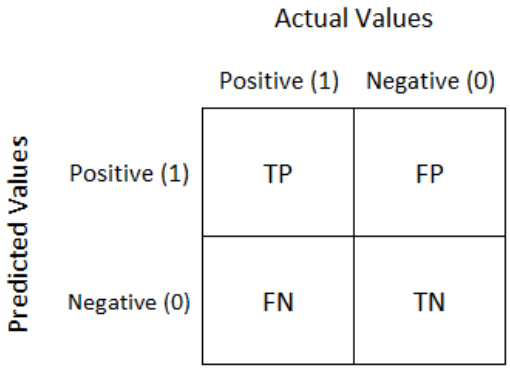
1. max_depth digunakan untuk menentukan maksimum kedalaman *tree*.
2. learning_rate digunakan untuk menentukan seberapa banyak step yang dilakukan pada setiap iterasi.
3. n_estimator digunakan untuk menentukan seberapa banyak tree yang ingin

digunakan pada model xgboost.

4. min_child_weight digunakan untuk menentukan jumlah bobot minimum yang digunakan dalam *child node*.
5. subsample digunakan untuk mengontrol seberapa banyak bagian *training instances* yang digunakan pada setiap tree didalam xgboost.
6. colsample_bytree digunakan untuk mengontrol seberapa banyak *feature* yang dipilih secara *random sample* pada setiap *tree* saat *training*.
7. gamma digunakan untuk menentukan minimum dari *loss reduction* yang dibutuhkan untuk split *tree*.
8. alpha digunakan untuk mengontrol *L1 regularization*.

### 2.2.14 Evaluasi Model

Pengujian model pada penelitian ini dengan membandingkan hasil test *precision*, *recall*, *f1-score*, dan *Matthews's correlation coefficient* (MCC) *metrics* untuk mendapatkan hasil terbaik. Untuk mendapatkan pengukuran tersebut dibutuhkan *confusion matrix* seperti berikut.




Gambar 2.19: Confusion Matrix

yang mana *confusion matrix* dibagi menjadi 4 yaitu:

1. True Positive (TP), jumlah data prediksi positif yang terdeteksi benar.
2. True Negative (TN), jumlah data prediksi negatif yang terdeteksi benar.
3. False Positive (FP), jumlah data prediksi negatif yang terdeteksi data positif.
4. False Negative (FN), jumlah data prediksi positif yang terdeteksi data negatif.

Barulah berdasarkan *confusion matrix* kita dapat menghitung *precision*, *recall*, *f-1 score*, dan *Matthews's correlation coefficient* (MCC) *metrics*.

#### 2.2.14.1 Precision

*Precision* adalah perbandingan antara jumlah prediksi positif yang benar(TP) dengan total jumlah prediksi positif(TP + FP). *Precision* menunjukan seberapa seberapa akurat model dalam memprediksi kelas positif. *Precision* digunakan untuk menjawab pertanyaan seperti "dari semua *instances* yang diprediksi positif, berapa banyak yang benar-benar positif?". *Precision* dihitung dengan formula seperti berikut:

$$Precison = \frac{TP}{TP + FP} \tag{2.7}$$

Kegunaan dari *precision* itu sangat penting dalam kasus dimana false positive tinggi. Contohnya dalam pendeteksi penipuan kartu kredit, kita ingin mengurangi kemungkinan mendeteksi transaksi yang tidak penipuan namun dikategorikan sebagai transaksi penipuan.

#### 2.2.14.2 Recall

*Recall* adalah perbandingan antara jumlah prediksi positif yang benar(TP) dengan total jumlah data aktual yang positif(TP+FN). *Recall* menunjukkan seberapa baik model dalam menangkap semua data positif yang ada. *Recall* digunakan untuk menjawab pertanyaan seperti "Dari semua kejadian yang benar-benar positif, berapa banyak yang diprediksi positif dengan tepat?". *Recall* dapat dihitung dengan formula seperti berikut:

$$Precison = \frac{TP}{TP + FN} \tag{2.8}$$

Kegunaan dari *recall* ialah pada kasus dimana memerlukan deteksi penyakit kita ingin recall yang tinggi untuk menangkap semua kasus penyakit agar tidak ada pasien yang tidak terdiagnosis. Recall berfokus pada memastikan bahwa semua kasus positif terdekteksi meskipun ada beberapa kesalahan *false positive* sedangkan *precision* berguna ketika penting untuk memastikan bahwa prediksi benar-benar akurat, sehingga mengurangi jumlah *false positive*.

#### 2.2.14.3 F1-Score

*F1-Score* adalah rata-rata harmonik dari precision dan recall. *F1-Score* memberikan keseimbangan antara precision dan recall yang berguna saat ada ketidakseimbangan kelas.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.9}$$

Kegunaan dari *f1-score* ialah pada situasi dimana penting untuk menjaga keseimbangan antara *precision* dan recall, untuk menghindari bias terhadap kelas mayoritas.

#### 2.2.14.4 Matthews correlation coefficient

*Matthew's Correlation Coefficient* (MCC) adalah ukuran evaluasi yang digunakan untuk mengukur kinerja sebuah model dalam melakukan klasifikasi biner. MCC berkerja dengan mempertimbangkan semuan nilai yang ada dalam *confusion matrix*(TP, TN, FP, FN) dalam menggambarkan hasil kualitas prediksi model, dikarenakan mcc mempertimbangkan semua elemen yang ada pada confusion metrix maka mcc dapat dijadikan sebuah metrik evaluasi yang baik untuk kelas yang tidak seimbang(*unbalance*)[53]. MCC dapat dihitung dengan formula seperti berikut:

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.10}$$

Hasil dari nilai *matthew's Correlation Coefficient* diinterpretasikan sebagai

1. +1: Model membuat prediksi sempurna (semua prediksi benar).
2. 0: Model tidak bisa melakukan prediksi dan sama seperti tebakan acak.
3. 1: Model membuat prediksi yang salah secara total.

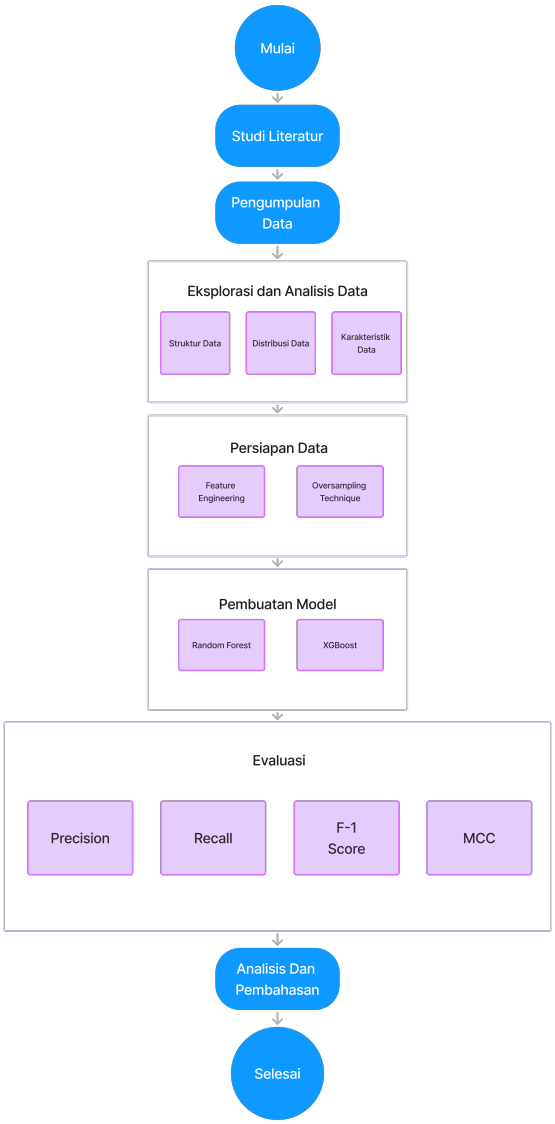
Pada penelitian ini juga peneliti akan menggunakan mcc sebagai metrik utama untuk menentukan gambaran performa model dan peneliti tidak akan menggunakan roc auc disebabkan semakin tinggi hasil dari mcc(contoh: MCC = 0.9) maka semakin tinggi pula nilai roc auc namun tidak sebaliknya[54].

# BAB III

# ANALISIS DAN PERANCANGAN

## 3.1 Alur Penelitian

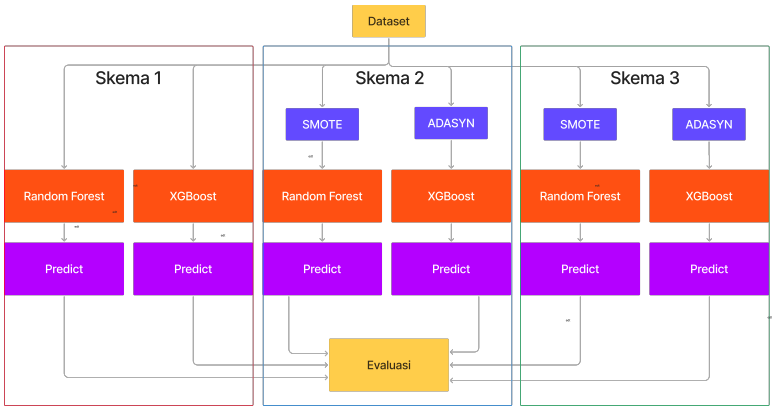
berikut merupakan alur penelitian yang dilakukan oleh peneliti.




Gambar 3.1: Alur Penelitian

## 3.2 Penjabaran Langkah Penelitian

Alur penelitian yang akan dilakukan memiliki beberapa tahapan yang dilakukan secara bertahap dan terdapat sub tahapan pada persiapan data dan juga pembuatan model yang dapat kita lihat pada gambar 3.2.



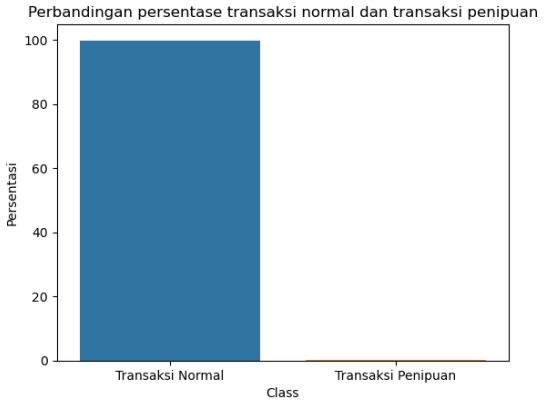

Gambar 3.2: Skema Pembuatan Model

### 3.2.1 Studi Literatur

Proses ini merupakan proses awal yang perlu dilakukan dalam melakukan penilitan guna memahami dan mengetahui teori-teori yang dibutuhkan dalam penelitian. Studi literatur yang dilakukan mencakup algoritma apa yang digunakan dan metode apa yang digunakan dalam membangun model pendeteksi kartu kredit. Hasil dari proses studi literatur yang peneliti lakukan yang berlandasan kepada 6 penelitian yang tinjau pustakanya menunjukan belum adanya analisis yang berfokus kepada teknik oversampling dalam mengatasi transaksi penipuan kartu kredit. Riset-riset sebelumnya banyak berfokus kepada algoritma *machine learning* apa yang digunakan, padahal proses *oversampling* sangat menentukan apakah sebuah algoritma itu bisa berjalan dengan baik atau tidak[15], terkhusus pada kasus deteksi penipuan kartu kredit. *Imbalanced data* pada dataset kartu kredit merupakan hal krusial dalam pembuatan model pendekteksi penipuan kartu kredit dikarenakan dapat membuat model *machine learnng* menjadi bias terhadap kelas minoritas dan salah cara efektif untuk mengatasi masalah hal tersebut ialah dengan menggunakan teknik oversampling[17].

### 3.2.2 Pengumpulan Data

Pada proses ini peneliti menggunakan dataset yang dikumpulkan oleh *machine learning group* dari Université Libre de Bruxelles[22], [55] yang berisi transaksi yang terjadi di eropa. Dataset ini memiliki 492 frauds(penipuan) dari 284.807 transaksi. Dataset ini dipilih oleh peneliti dikarenakan *dataset* tersebut sangat tidak seimbang( *highly unbalanced*) dengan persentase penipuan sebesar 0.172% yang dapat dilihat pada gambar 3.3 yang membuat dataset tersebut sangat cocok dalam penelitian ini. Dataset tersebut disajikan dalam bentuk file CSV. Dataset tersebut memiliki 30 fitur dan satu label dengan 28 fiturnya sudah dalam bentuk *PCA transformation* guna melindungi privasi data(confidentiality issues) dan 2 fitur lainya tidak dirubah kedalam bentuk PCA adalah *time*, dan *amount* dan untuk label berupa *class*. Fitur 'Time' berisi jumlah detik yang telah berlalu antara setiap transaksi dan transaksi pertama dalam dataset. Fitur 'Amount' merupakan jumlah transaksi. Label 'Class' ini berisi 2 bentuk 1 dan 0 yang mana 1 merupakan *fraud* dan 0 *non-fraud*.



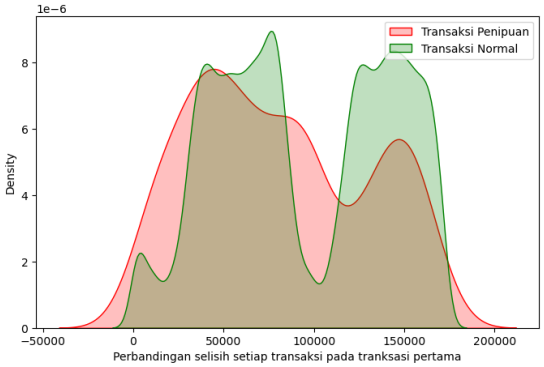

Gambar 3.3: Perbandingan Transaksi Normal dan Penipuan

### 3.2.3 Eksplorasi dan Analisis Data

Hasil dari explorasi dan analisis data pada tipe data dataset *credit card fraud detection* memiliki tipe data *float64* pada hampir semua fiturnya kecuali pada fitur class yang memiliki tipe data *int64* dikarenakan fitur class masih dalam bentuk int64, sebaiknya diubah tipe datanya kedalam bentuk tipe data *category* dikarenakan membuat penggunaan memori menjadi efisien[56]. Memori efisien berpotensi meningkatkan

performa dalam *training model*.

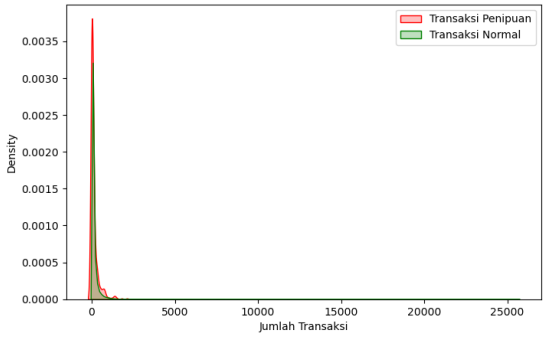
Peneliti selanjutnya mencoba untuk melakukan analisis dataset pada fitur time dan amount dikarenakan 2 fitur tersebut tidak diubah kedalam bentuk PCA dan peneliti ingin melihat apakah fitur tersebut tetap dipertahankan atau tidak. Pada fitur *time* peneliti akan melakukan observasi distribusi pada transaksi normal dan penipuan berdasarkan waktu dengan menggunakan diagram KDE untuk melihat distribusinya, yang dapat dilihat pada gambar 3.4 yang mana pada figure tersebut bisa kita lihat kalau kedua transaksi normal dan penipuan memiliki distribusi ganda yang memiliki dua puncak yang berarti merupakan *bimodal distribution*[57], hal ini menunjukkan bahwa transaksi dalam dataset terjadi pada dua periode waktu yang berbeda dan setelah itu dapat kita lihat kalau transaksi normal dan penipuan juga merupakan *distribution overlap*[57] dikarenakan hal itu kita akan tetap mempertahankan menggunakan *feature* tersebut.



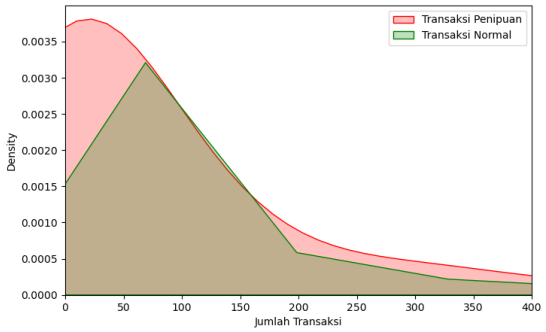

Gambar 3.4: Perbandingan Distribusi Transaksi Normal dan Penipuan Berdasarkan Waktu

Pada fitur *amount* peneliti akan melakukan observasi distribusi pada transaksi normal dan penipuan berdasarkan *amount* yang dilakukan menggunakan diagram KDE sama seperti saat melakukan observasi pada fitur *time*. Berdasarkan hasil observasi distribusi pada gambar 3.5 menunjukan bahwa distribusi dari kedua fitur tersebut merupakan *skewed distribution* yang terpusat pada jumlah transaksi yang rendah(diantara 0 sampai 100) yang dapat kita lihat pada gambar 3.6. Kedua fitur tersebut merupakan *overlap distribution* namun sedikit perbedaan yang cukup jelas bahwa distribusi transaksi penipuan lebih rendah daripada transaksi normal dan perbedaan selanjutnya pada kedua fitur tersebut ialah transaksi penipuan jarang

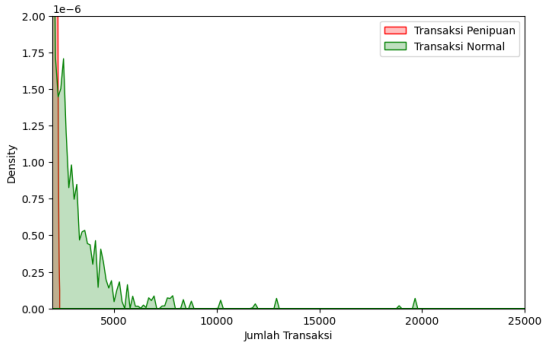
terjadi pada jumlah transaksi besar(diatas 2500 sampai 25000) yang dapat dilihat pada gambar 3.7 dikarenakan banyaknya hubungan yang membedakan transaksi normal dan penipuan, fitur *amount* akan digunakan oleh peneliti.




Gambar 3.5: Perbandingan Jumlah Transaksi Transaksi Normal dan Penipuan




Gambar 3.6: Jumlah Transaksi 0 Sampai 100




Gambar 3.7: Jumlah Transaksi 2500 Sampai 25000

### 3.2.4 Persiapan Data

Proses memuat persiapan data sebelum diinputkan ke dalam *machine learning*. Setelah peneliti melakukan analisis dan eksplorasi data, ada beberapa hal yang perlu

dilakukan dalam persiapan data, yang pertama ialah mengecek apakah ada *missing value* didalam dataset, dengan cara mengecek persentase dari setiap feature yang ada pada dataset dengan menggunakan formula berikut:

$$\text{Null\_Percentage}_i = \left( \frac{\sum_{j=1}^{n} \mathbf{1}\{x_{ij} = \text{null}\}}{n} \right) \times 100 \tag{2.1}$$

Setelah itu peneliti akan melakukan*split* data train dan test dengan metode pengambilan datanya dengan menggunakan *stratified sampling* dengan format 80:20, 80% data training dan 20% data test dengan menggunakan modul sklearn model selection dengan melakukan import train_test_split. Setelah itu Penelti akan melakukan feature scaling pada feature amount dikarenakan feature amount ukuran nya masih sangat berbeda dibandingkan dengan feature lainya yang sudah dilakukan PCA transformation. Untuk itu, penelti akan melakukan feature scaling dengan menggunakan *Robust Scaling* dari sklearn. Hasil dari feature scaling pada data training dapat dilihat perubahaannya dari table 3.2 menjadi 3.3.

Tabel 3.1: 5 Data Train Amount Sebelum Scaling

| No. | Amount |
|---|---|
| 1 | 7.32 |
| 2 | 2.99 |
| 3 | 175.10 |
| 4 | 6.10 |
| 5 | 86.10 |

Tabel 3.2: 5 Data Train Amount Sesudah Feature Scaling

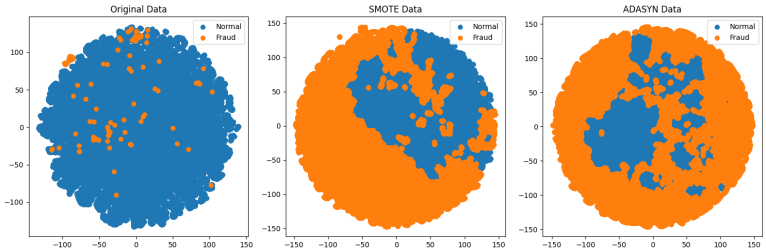
| No. | Amount |
|---|---|
| 1 | 0.000258 |
| 2 | 0.000161 |

*Continued*

Tabel 3.2: 5 Data Train Amount Sesudah Feature Scaling

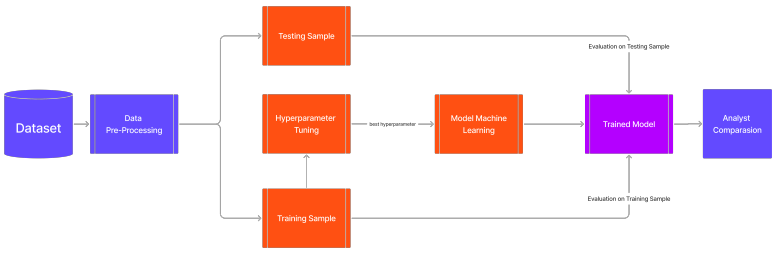
| No. | Amount |
|---|---|
| 3 | 0.006816 |
| 4 | 0.000237 |
| 5 | 0.003351 |

Selanjutnya peneliti akan melakukan metode *oversampling* yang akan diimplementasikan pada skema 2 dan 3 yang dapat divisualisasikan kedalam plot diagram. namun sebelum itu, agar dapat divisualisasikan kedalam plot diagram yang berupa 2 dimensi data harus diubah dimensinya dikarenakan data yang kita miliki merupakan 30 dimensi, perubahan dimensi tanpa menghilangkan relationship data ialah dengan menggunakan T-SNE dikarenakan T-SNE , hasil dari T-SNE dapat dilihat pada gambar 3.8.




Gambar 3.8: Visualisasi Data TSNE sebelum dan Sesudah SMOTE dan ADASYN

### 3.2.5 Pembuatan Model




Gambar 3.9: Alur Pembuatan Model

Proses ini akan berfokus pada tuning *hyperparameter* model machine learning dengan *cross validation* dengan menggunakan scoring mcc dan melakukan *training model* dengan nilai *hyperparameter* terbaik. Tuning Hyperparameter dilakukan dengan menggunakan *hyperparameter optimization framework* bernama optuna. Pada algoritma *machine learning random forest hyperparameter* yang akan dituning berupa[58]:

1. n_estimator dengan range nilai dari 200 ke 500.
2. max_depth dengan range nilai 5 ke 30.
3. min_sample_split dengan range nilai 5 ke 30.
4. min_sample_leaf dengan range nilai 5 ke 30.
5. max_feature dengan pilihan 'auto', 'sqrt', dan 'log2'.
6. bootstrap dengan pilihan *true* or *false*.

Pada algoritma *machine learning xgboost hyperparamter* yang akan dituning dengan menggunakan optuna adalah sebagai berikut[59]:

1. max_depth dengan range 3 ke 10.
2. learning_rate dengan range 0.01 ke 0.1.
3. n_estimator dengan range 200 ke 500.
4. min_child_weight dengan range 3 ke 10.
5. subsample dengan range 0.6 ke 1.0.
6. colsample_bytree dengan range 0.6 ke 1.0.
7. gamma dengan range 0 ke 1.0.
8. alpha dengan range 0 ke 5.

Setelah melakukan hyperparameter tuning penelti akan memilih *hyperparameter* terbaik untuk membuat model random forest dan juga xgboost.

### 3.2.6 Evaluasi

Pengujian dilakukan dengan melihat berdasarkan nilai dari *precision*, *recall*, *f-1 score*, dan *Matthews's correlation coefficient* (MCC) pada setiap model yang digunakan dalam penelitian ini seperti *random forest* dan *xgboost*. Evalusi akan dilakukan dengan

4 skenario yaitu dengan membandingkan hasil skenario tersebut dan setiap skenario akan membuat dua model *random forest* dan *xgboost* dengan metode tertentu.

1. Skenario pertama: Pada skenario pertama peneliti akan melakukan pengecekan performa dari model *random forest* dan *xgboost* tanpa melakukan metode oversampling, yang artinya model dibuat berdasarkan data yang tidak seimbang. Pengecekan perfoma akan dilihat berdasarkan *precision*, *recall*, *f1-score*, dan *Matthews's correlation coefficient* (MCC) .
2. Skenario kedua: pada skenario kedua peneliti akan melakukan pengecekan performa dari model *random forest* dan *xgboost* dengan menggunakan metode oversampling SMOTE. Pengecekan perfoma akan dilihat berdasarkan *precision*, *recall*, *f1-score*, dan *Matthews's correlation coefficient* (MCC).
3. Skenario ketiga: pada skenario ketiga peneliti akan melakukan pengecekan performa dari model *random forest* dan *xgboost* dengan menggunakan metode oversampling ADASYN. Pengecekan perfoma akan dilihat berdasarkan *precision*, *recall*, *f1-score*, dan *Matthews's correlation coefficient* (MCC).
4. Skenario keempat: pada skenario keempat peneliti akan melakukan pengecekan performa secara menyeluruh dengan membandingkan seluruh hasil dari setiap 6 percobaan yang dilakukan dan menentukan hasil percobaan terbaik dengan melihat metriks performa berdasarkan *precision*, *recall*, *f1-score*, dan *Matthews's correlation coefficient* (MCC).

### 3.2.7 Analisis dan Pembahasan

Pada tahap in, dilakukan analisis lebih lanjut beserta pembahasan secara detail terhadap pengujian yang telah dilakukan. Setelah melakukan analisis akan dilakukan proses pembahasan dalam bentuk penulisan laporan.

### 3.2.8 Perhitungan SMOTE

Perhitungan SMOTE dimulai dengan mengetahui data minoritas, sebagai contoh kita memiliki 3 data minortas yaitu:

1. Sample A: (2, 3)

2. Sample B: (4, 7)

3. Sample C: (3, 4)

Setelah mengetahui data minoritas kita akan mencari *iniitial sample* yang didapatkan dengan melakukan pemilihan acak. Setelah itu, kita akan mencari tetangga terdekat dari *initial sample* ke sampel lainnya dengan menggunakan *Euclidean distance*. Untuk mendapatkan nilai *Euclidean distance* itu dengan menggunakan teorema pythagoras dengan rumus:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.2}$$

Kita dapatkan secara acak *intial sample* yaitu sampel A dan selanjutnya kita akan menghitung *Euclidean distance*.

1. Jarak A ke B.

$$d_{AB} = \sqrt{(4-2)^2 + (7-3)^2} = \sqrt{2^2 + 4^2} = \sqrt{4+16} = \sqrt{20} \approx 4.47$$

2. Jarak A ke C.

$$d_{AC} = \sqrt{(3-2)^2 + (4-3)^2} = \sqrt{1^2 + 1^2} = \sqrt{1+1} = \sqrt{2} \approx 1.41$$

Karena $d_{AC} < d_{AB}$, sampel C merupakan tetangga terdekat dari sampel A. Setelah mengetahui tetangga terdekat, selanjutnya, kita akan membuat data sintetis dengan menggunakan rumus 2.4.

- Sampel A: (2,3).
- Sample C (tetangga terdekat): (3,4).
- $\lambda$ : 0.5 (dipilih secara acak)

Menghitung perbedaan sampel A dan sampel C:

$$x_{synthetic} = \text{Sampel A} + \lambda \cdot (\text{Sampel C} - \text{Sampel A})$$

$$x_{synthetic} = \text{Sampel A} + \lambda \cdot ((3-2), (4-3))$$

$$x_{synthetic} = \text{Sampel A} + \lambda \cdot (1, 1)$$

Mengkalikan dengan lamdha:

$$x_{synthetic} = \text{Sampel A} + 0.5 \cdot (1, 1)$$

$$x_{synthetic} = \text{Sampel A} + (0.5, 0.5)$$

Menghasilkan data sintetis:

$$x_{synthetic} = (2, 3) + (0.5, 0.5)$$

$$x_{synthetic} = (2.5, 3.5)$$

data sintetis yang telah berhasil dibuat adalah (2.5, 3.5).

### 3.2.9 Perhitungan ADASYN

Perhitungan ADASYN dimulai dengan mengetahui data mayoritas dan data minoritas. Kita asumsikan kita memliki dataset seperti berikut:

- Sampel Minoritas.
    1. Sampel A: (2,3).
    2. Sampel B: (3,7).
- Sampel Majoritas.
    1. Sampel M1: (2.5, 3.2).
    2. Sampel M2: (2.7, 3.1).
    3. Sample M3: (6,9).

Dan kita akan menginisiasi nilai k(*number of nearest neighbors* = 3 dan total data sintetis yang ingin kita buat adalah 9 ($G = 9$). Setelah itu kita akan mencari tetangga terdekat dari setiap data minoritas dengan rumus 3.3.2.8. Untuk data minortas pada sampel A(2, 3):

1. Jarak A ke M1.

$$d_{AM1} = \sqrt{(2.5 - 2)^2 + (3.2 - 3)^2} = \sqrt{0.5^2 + 0.2^2} = \sqrt{0.25 + 0.04} = \sqrt{0.29} \approx 0.54$$

2. Jarak A ke M2.

$$d_{AM2} = \sqrt{(2.7-2)^2 + (3.1-3)^2} = \sqrt{0.7^2 + 0.1^2} = \sqrt{0.49 + 0.01} = \sqrt{0.50} \approx 0.71$$

3. Jarak A ke M3.

$$d_{AM3} = \sqrt{(6-2)^2 + (9-3)^2} = \sqrt{4^2 + 6^2} = \sqrt{16 + 36} = \sqrt{52} \approx 7.21$$

4. Jarak A ke B.

$$d_{AB} = \sqrt{(4-2)^2 + (7-3)^2} = \sqrt{2^2 + 4^2} = \sqrt{4 + 16} = \sqrt{20} \approx 4.47$$

Pada sampel A 3 tetangga terdekat A($k = 3$) ialah M1,M2, dan B. yang berarti ratio dari sampel A adalah:

$$r_A = \frac{Majority}{k} = \frac{2}{3} \approx 0.67$$

Untuk data minoritas pada sampel B(4, 7):

1. Jarak B ke M1.

$$d_{BM1} = \sqrt{(4-2.5)^2 + (7-3.2)^2} = \sqrt{1.5^2 + 3.8^2} = \sqrt{2.25 + 14.44} = \sqrt{16.69} \approx 4.09$$

2. Jarak B ke M2.

$$d_{BM2} = \sqrt{(4-2.7)^2 + (7-3)^2} = \sqrt{1.3^2 + 3.9^2} = \sqrt{1.69 + 15.21} = \sqrt{16.9} \approx 4.11$$

3. Jarak B ke M3.

$$d_{BM3} = \sqrt{(4-6)^2 + (7-9)^2} = \sqrt{(-2)^2 + (-2)^2} = \sqrt{4 + 4} = \sqrt{16} \approx 2.83$$

4. Jarak A ke B.

$$d_{BA} \approx 4.47$$

Pada sampel B 3 tetangga terdekat ($k = 3$) ialah M3,M1, dan M2. yang berarti ratio dari sampel B adalah:

$$r_B = \frac{Majority}{k} = \frac{3}{3} = 1$$

Menentukan alokasi data sintetis:

$$R_total = r_a + r_b = 0.69 + 1 = 1.69$$

Menghitung *weight* pada setiap minority sampel:

- *weight* dari sampel A:

$$w_A = \frac{r_a}{R_total} = \frac{0.67}{1.67} \approx 0.40$$

- *weight* dari sampel B:

$$w_B = \frac{r_b}{R_total} = \frac{1}{1.67} \approx 0.60$$

Maka jumlah data sintetis pada masing data minoritas ialah:

- Data sintetis pada sampel A:

$$G_A = w_A \times G \approx 0.40 \times 9 \approx 3.6$$

- Data sintetis pada sample B:

$$G_B = w_B \times G \approx 0.60 \times 9 \approx 5.4$$

Jumlah dari masing-masing data sintetis ialah 4(dibulatkan) pada sampel A dan 5(dibulatkan) pada sampel B. Setelah mengalokasikan berapa data sintetis yang akan dibuat, dilanjutkan dengan membuat data sintetis pada setiap data minoritas dengan menggunakan rumus 2.4. Kita asumsikan bahwa kita memilih secara acak data minoritas dan yang terpilih ialah sampel A dan tetangga terdekat yang terpilih ialah sampel B(karena sampel B merupakan satu-satunya data minoritas lainnya).
Menghitung nilai pembeda dari sampel A dan B:

$$x_{synthetic} = \text{Sampel A} + \lambda \cdot (\text{Sampel B} - \text{Sampel A})$$
$$x_{synthetic} = \text{Sampel A} + \lambda \cdot ((4-2), (7-3))$$
$$x_{synthetic} = \text{Sampel A} + \lambda \cdot (2, 4)$$

Mengkalikan dengan lamdha ($\lambda = 0.5$):

$$x_{synthetic} = \text{Sampel A} + 0.5 \cdot (2, 4)$$
$$x_{synthetic} = \text{Sampel A} + (1, 2)$$

Menghasilkan data sintetis:

$$x_{synthetic} = (2, 3) + (1, 2)$$

$$x_{synthetic} = (3, 5)$$

data sintetis yang telah berhasil dibuat adalah (3, 5).

### 3.2.10 Alat dan Bahan Tugas Akhir

Adapun alat dan bahan yang digunakan dalam penelitian ini adalah sebagai berikut:

### 3.2.11 Alat

Berikut adalah alat yang digunakan dalam penelitian ini:

1. Spesifikasi Laptop, komputer atau notebook:

   (a) Minimum spesifikasi komputer, laptop, atau notebook agar dapat melakukan penelitian ini, sebagai berikut:

      i. Sistem operasi linux atau Windows 10 (64-bit), x86-64 based Menggunakan Processor Intel Core i5.

      ii. Menggunakan Processor Intel Core i5 atau AMD Ryzen 5.

      iii. Kebutuhan RAM paling minimal 8 GB.

      iv. Memiliki ruang memori yang bebas setidaknya 10 GB.

   (b) Pada penelitian ini penulis menggunakan laptop dengan spesifikasi sebagai berikut:

      i. ubuntu 22.04.4(64-bit),x86_64 based

      ii. AMD Ryzen 5 4600H with Radeon G

      iii. RAM 8.

      iv. SSD 512GB

      v. GPU: NVIDIA GeForce GTX 1650 Ti

2. Visual Studio Code

3. Scikit-Learn

### 3.2.12 Bahan

Berikut adalah bahan yang digunakan dalam penelitian ini:

1. Dataset yang digunakan ialah dataset Credit Card Fraud Detection yang dikumpulkan dan dianalisis oleh *machine learning group* Université Libre de Bruxelles(ULB)

2. Jurnal penelitian dari penelitian sebelumnya yang dijadikan dasar teori dan rangkaian konsep serta ide untuk mendukung penelitian.

# BAB IV

# HASIL DAN PEMBAHASAN

## 4.1 Hasil Penelitian

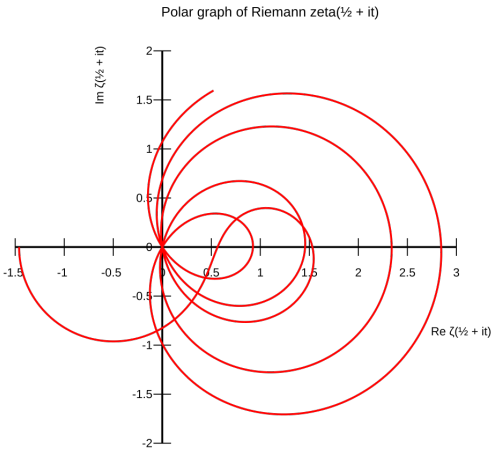
Berisi hasil penelitian berdasarkan rancangan yang sudah dijelaskan pada Bab III, terutama dari Subbab **??**. Bagi yang membuat alat, jelaskan alat yang jadi dalam bentuk apa. Bagi yang membuat aplikasi, jelaskan aplikasi yang jadi dalam bentuk seperti apa. Jabarkan dalam bentuk pseudocode dan dijelaskan per bagian kodenya. Gunakan gambar dan tabel sebagai alat bantu menjelaskan hasil.

## 4.2 Hasil Pengujian

Berikan hasil pengujian berdasarkan rancangan & skenario yang sudah direncanakan sebelumnya pada Subbab **??**.

Tabel 4.1: Contoh Hasil Pengujian

| **Pengujian** | **Metode A** | **Metode B** |
|---|---|---|
| Kecepatan | 10 ms | 12 ms |
| Memori | 10 MB | 7 MB |




Gambar 4.1: Contoh Graf Pengujian

## 4.3 Analisis Hasil Penelitian

Berikan analisis hasil penelitian & pengujian, berupa data yang didapatkan dari penelitian & pengujian Tugas Akhir yang sudah anda kerjakan. Gunakan gambar dan tabel sebagai alat bantu menjelaskan analisis hasil. Data luaran penelitian yang dapat dianalisis berupa:

1. Hasil pengujian
2. Hasil kuesioner
3. Aplikasi yang dikembangkan

Analisis dapat membandingkan dengan hasil penelitian sebelumnya yang memiliki kemiripan topik.

## 4.4 Pembahasan

Berisi pembahasan terkait hasil yang sudah didapatkan/dipaparkan sebelumnya, berupa penutup yang dapat menjelaskan mengenai kelebihan hasil tugas akhir dan kekurangannya dibandingkan dengan penelitian atau produk lain yang serupa atau mirip.

# BAB V

# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Berisi kesimpulan dari hasil dan pembahasan terkait penelitian yang dilakukan, dapat juga berupa temuan yang Anda dapatkan setelah melakukan penelitian atau analisis terhadap tugas akhir Anda. Berhubungan dengan poin pada Subbab 1.2 dan 1.3.

## 5.2 Saran

Berisi saran mengenai aspek Tugas Akhir atau temuan dalam penelitian. Diutamakan saran berdasarkan hasil analisis dari Subbab 4.3. Saran dapat dikembangkan dan diperkaya untuk Tugas Akhir selanjutnya.

# DAFTAR PUSTAKA


[1] Iman Prayogo Suryohadibroto **and** Djoko Prakoso. "Surat berharga: alat pembayaran dalam masyarakat modern". **in***(No Title)*: (1987).

[2] Decky Hendarsyah. "Analisis perilaku konsumen dan keamanan kartu kredit perbankan". **in***JPS (Jurnal Perbankan Syariah)*: 1.1 (2020), **pages** 85–96.

[3] *Identity Theft Statistics*. 2024. URL: https://www.experian.com/blogs/ask-experian/identity-theft-statistics/.

[4] Matt Rej. *Credit Card Fraud Statistics (2024)*. 2024. URL: https://merchantcostconsulting.com/lower-credit-card-processing-fees/credit-card-fraud-statistics/.

[5] Khyati Chaudhary, Jyoti Yadav **and** Bhawna Mallick. "A review of fraud detection techniques: Credit card". **in**45: 1 (2012), **pages** 39–44.

[6] Chase & J.P. Morgan. *Fraud Management in Commercial Cards: Proactive Vigilance and Collaboration Required*. 2016.

[7] Jarrod West **and** Maumita Bhattacharya. "Intelligent financial fraud detection: a comprehensive review". **in***Computers & security*: 57 (2016), **pages** 47–66.

[8] Eric WT Ngai, Yong Hu, Yiu Hing Wong **andothers**. "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature". **in***Decision support systems*: 50.3 (2011), **pages** 559–569.

[9] Sevdalina Georgieva, Maya Markova **and** Velizar Pavlov. "Using neural network for credit card fraud detection". **in***AIP Conference Proceedings*: **volume** 2159. 1. AIP Publishing. 2019.

[10] Ibtissam Benchaji, Samira Douzi **and** Bouabid El Ouahidi. "Using genetic algorithm to improve classification of imbalanced datasets for credit card fraud detection". **in***Smart Data and Computational Intelligence: Proceedings of the International Conference on Advanced Information Technology, Services and Systems (AIT2S-18) Held on October 17–18, 2018 in Mohammedia 3*: Springer. 2019, **pages** 220–229.

[11] KR Seeja **and** Masoumeh Zareapoor. “Fraudminer: A novel credit card fraud detection model based on frequent itemset mining”. **in***The Scientific World Journal*: 2014.1 (2014), **page** 252797.

[12] Sheo Kumar, Vinit Kumar Gunjan, Mohd Dilshad Ansari **andothers**. “Credit card fraud detection using support vector machine”. **in***Proceedings of the 2nd International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications: ICMISC 2021*: Springer. 2022, **pages** 27–37.

[13] Jyoti R Gaikwad, Amruta B Deshmane, Harshada V Somavanshi **andothers**. “Credit card fraud detection using decision tree induction algorithm”. **in***International Journal of Innovative Technology and Exploring Engineering (IJITEE)*: 4.6 (2014), **pages** 2278–3075.

[14] Ayoub Mniai, Mouna Tarik **and** Khalid Jebari. “A novel framework for credit card fraud detection”. **in***IEEE Access*: (2023).

[15] Putu Tirta Sari Ningsih, Muhammad Gusvarizon **and** Rudi Hermawan. “Analisis Sistem Pendeteksi Penipuan Transaksi Kartu Kredit dengan Algoritma Machine Learning”. **in***Jurnal Teknologi Informatika dan Komputer*: 8.2 (2022), **pages** 386–401.

[16] Thomas G Dietterich **andothers**. “Ensemble learning”. **in***The handbook of brain theory and neural networks*: 2.1 (2002), **pages** 110–125.

[17] Alexander Yun-chung Liu. “The effect of oversampling and undersampling on classifying imbalanced text datasets”. **in**(2004).

[18] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall **andothers**. “SMOTE: synthetic minority over-sampling technique”. **in***Journal of artificial intelligence research*: 16 (2002), **pages** 321–357.

[19] Haibo He, Yang Bai, Edwardo A. Garcia **andothers**. “ADASYN: Adaptive synthetic sampling approach for imbalanced learning”. **in***2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*: 2008, **pages** 1322–1328. DOI: 10.1109/IJCNN.2008.4633969.

[20] Jakob Brandt **and** Emil Lanzén. “A comparative review of SMOTE and ADASYN in imbalanced data classification”. **in**(2021).

[21] Andrea Dal Pozzolo, Olivier Caelen, Reid A Johnson **andothers**. "Calibrating probability with undersampling for unbalanced classification". **in***2015 IEEE symposium series on computational intelligence*: IEEE. 2015, **pages** 159–166.

[22] Machine Learning Group ULB. *Credit Card Fraud Detection*. `https://www.kaggle.com/mlg-ulb/creditcardfraud`. Created: 2018-03-23.

[23] RB Asha **and** Suresh Kumar KR. "Credit card fraud detection using artificial neural network". **in***Global Transitions Proceedings*: 2.1 (2021), **pages** 35–41.

[24] Mohammed Tayebi **and** Said El Kafhali. "Hyperparameter optimization using genetic algorithms to detect frauds transactions". **in***The International Conference on Artificial Intelligence and Computer Vision*: Springer. 2021, **pages** 288–297.

[25] Ruttala Sailusha, V. Gnaneswar, R. Ramesh **andothers**. "Credit Card Fraud Detection Using Machine Learning". **in***2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*: 2020, **pages** 1264–1270. DOI: `10.1109/ICICCS48265.2020.9121114`.

[26] Abdul Rehman Khalid, Nsikak Owoh, Omair Uthmani **andothers**. "Enhancing credit card fraud detection: an ensemble machine learning approach". **in***Big Data and Cognitive Computing*: 8.1 (2024), **page** 6.

[27] Archana Purwar **and** Ms Manju. "Credit card fraud detection using XGBoost for imbalanced data set". **in***Proceedings of the 2023 Fifteenth International Conference on Contemporary Computing*: 2023, **pages** 216–219.

[28] Ethem Alpaydin. *Machine learning*. MIT press, 2021.

[29] Alexander L Fradkov. "Early history of machine learning". **in***IFAC-PapersOnLine*: 53.2 (2020), **pages** 1385–1390.

[30] Michael I Jordan **and** Tom M Mitchell. "Machine learning: Trends, perspectives, and prospects". **in***Science*: 349.6245 (2015), **pages** 255–260.

[31] Pádraig Cunningham, Matthieu Cord **and** Sarah Jane Delany. "Supervised learning". **in***Machine learning techniques for multimedia: case studies on organization and retrieval*: Springer, 2008, **pages** 21–49.

[32] Sepp Hochreiter, A Steven Younger **and** Peter R Conwell. "Learning to learn using gradient descent". **in***Artificial Neural Networks—ICANN 2001:*

*International Conference Vienna, Austria, August 21–25, 2001 Proceedings 11*: Springer. 2001, **pages** 87–94.

[33] Sebastian Ruder. "An overview of gradient descent optimization algorithms". **in***arXiv preprint arXiv:1609.04747*: (2016).

[34] Chinwe Peace Igiri, Oscar Uzoma Anyama **and** Abasiama Ita Silas. "Effect of learning rate on artificial neural network in machine learning". **in**(2015).

[35] Matthew D Zeiler. "ADADELTA: an adaptive learning rate method". **in***arXiv preprint arXiv:1212.5701*: (2012).

[36] Leslie N Smith. "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay". **in***arXiv preprint arXiv:1803.09820*: (2018).

[37] Earl V Wooster. *Credit card*. US Patent 3,283,713. 1966.

[38] Daniel I Flitcroft **and** Graham O'donnell. *Credit card system and method*. US Patent 7,567,934. 2009.

[39] Gopesh Kumar. *Method for Providing Secured Card Transactions During Card Not Present (CNP) Transactions*. US Patent App. 14/717,735. 2016.

[40] Guo Haixiang, Li Yijing, Jennifer Shang **andothers**. "Learning from class-imbalanced data: Review of methods and applications". **in***Expert systems with applications*: 73 (2017), **pages** 220–239.

[41] Bartosz Krawczyk. "Learning from imbalanced data: open challenges and future directions". **in***Progress in artificial intelligence*: 5.4 (2016), **pages** 221–232.

[42] Li Yang **and** Abdallah Shami. "On hyperparameter optimization of machine learning algorithms: Theory and practice". **in***Neurocomputing*: 415 (2020), **pages** 295–316.

[43] Bernd Bischl, Martin Binder, Michel Lang **andothers**. "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges". **in***Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*: 13.2 (2023), e1484.

[44] Guozhu Dong **and** Huan Liu. *Feature engineering for machine learning and data analytics*. CRC press, 2018.

[45] Yan-Yan Song **and** LU Ying. “Decision tree methods: applications for classification and prediction”. **in***Shanghai archives of psychiatry*: 27.2 (2015), **page** 130.

[46] Wilson Andrés Castillo Rojas **and** Claudio Meneses Villegas. “Graphical representation and exploratory visualization for decision trees in the KDD process”. **in***2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI)*: IEEE. 2012, **pages** 1–10.

[47] Shan Suthaharan **and** Shan Suthaharan. “Decision tree learning”. **in***Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning*: (2016), **pages** 237–269.

[48] Leo Breiman. “Random forests”. **in***Machine learning*: 45 (2001), **pages** 5–32.

[49] Leo Breiman. “Bagging predictors”. **in***Machine learning*: 24 (1996), **pages** 123–140.

[50] T Chen. “Xgboost: extreme gradient boosting”. **in***R package version 0.4-2*: 1.4 (2015).

[51] Tianqi Chen **and** Carlos Guestrin. “Xgboost: A scalable tree boosting system”. **in***Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*: 2016, **pages** 785–794.

[52] XGBoost Developers. *XGBoost Parameter Tuning*. Accessed December 2, 2024. 2024. URL: `https://xgboost.readthedocs.io/en/stable/parameter.html`.

[53] Davide Chicco **and** Giuseppe Jurman. “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation”. **in***BMC genomics*: 21 (2020), **pages** 1–13.

[54] Davide Chicco **and** Giuseppe Jurman. “The Matthews correlation coefficient (MCC) should replace the ROC AUC as the standard metric for assessing binary classification”. **in***BioData Mining*: 16.1 (2023), **page** 4.

[55] Yann-Aël Le Borgne, Wissam Siblini, Bertrand Lebichot **andothers**. *Reproducible Machine Learning for Credit Card Fraud Detection - Practical Handbook*. Université Libre de Bruxelles, 2022. URL: `https://github.com/Fraud-Detection-Handbook/fraud-detection-handbook`.

[56] Wes McKinney **and** PD Team. “Pandas-Powerful python data analysis toolkit”. **version** 2.2.3. **in***Pandas—Powerful Python Data Analysis Toolkit*: (2024).

[57] Larry Gonick, Woollcott Smith **and** Woollcott Smith. *The cartoon guide to statistics*. HarperPerennial New York, 1993.

[58] Saleh Albahli. “Efficient hyperparameter tuning for predicting student performance with Bayesian optimization”. **in***Multimedia tools and applications*: 83.17 (2024), **pages** 52711–52735.

[59] Vibhu Verma. “Exploring Key XGBoost Hyperparameters: A Study on Optimal Search Spaces and Practical Recommendations for Regression and Classification”. **in***International Journal of All Research Education and Scientific Methods*: (2024). URL: https://api.semanticscholar.org/CorpusID:274319663.

# LAMPIRAN

## A Dataset

https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud