DETEKSI POTENSI *INFLATED BILLS* PADA DATA KLAIM LAYANAN BPJS KESEHATAN MENGGUNAKAN ALGORITMA *EXTREME GRADIENT BOOSTING* (XGBOOST)

SKRIPSI

KAYLA ALYSA ADRA

201402088

PROGRAM STUDI S1 TEKNOLOGI INFORMASI

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UNIVERSITAS SUMATERA UTARA

MEDAN

2025

DETEKSI POTENSI *INFLATED BILLS* PADA DATA KLAIM LAYANAN BPJS KESEHATAN MENGGUNAKAN ALGORITMA *EXTREME GRADIENT BOOSTING* (XGBOOST)

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazan Sarjana Teknologi Informasi

KAYLA ALYSA ADRA
201402088

PROGRAM STUDI S1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2025

iii

PERSETUJUAN

Judul	:	Deteksi Potensi *Inflated Bills* Pada Data Klaim Layanan BPJS Kesehatan Menggunakan Algoritma *Extreme Gradient Boosting* (Xgboost)
Kategori	:	SKRIPSI
Nama Mahasiswa	:	Kayla Alysa Adra
Nomor Induk Mahasiswa	:	201402088
Program Studi	:	Teknologi Informasi
Fakultas	:	Ilmu Komputer Dan Teknologi Informasi

Medan, 8 Januari 2025

Komisi Pembimbing :

Pembimbing 1,	Pembimbing 2,
Ulfi Andayani S.Kom., M.Kom	Ainul Hizriadi S.Kom., M.Sc
NIP : 198604192015042004	NIP : 198510272017061001

Diketahui/disetujui oleh

Ketua Program Studi S1 Teknologi Informasi,

Dedy Arisandi S.T., M.Kom.

NIP : 197908312009121002

PERNYATAAN

DETEKSI POTENSI *INFLATED BILLS* PADA DATA KLAIM LAYANAN BPJS KESEHATAN MENGGUNAKAN ALGORITMA *EXTREME GRADIENT BOOSTING* (XGBOOST)

SKRIPSI

Saya mengakui bahwa skripsi ini merupakan hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 8 Januari 2025

KAYLA ALYSA ADRA

201402088

DETEKSI POTENSI *INFLATED BILLS* PADA DATA KLAIM LAYANAN BPJS KESEHATAN MENGGUNAKAN ALGORITMA *EXTREME GRADIENT BOOSTING* (XGBOOST)

ABSTRAK

Inflated Bills atau tagihan yang melebihi biaya yang seharusnya merupakan salah satu bentuk kecurangan yang umum terjadi di Fasilitas Kesehatan Rujukan Tingkat Lanjut (FKRTL) dalam pelaksanaan program BPJS Kesehatan. Kecurangan tersebut dapat menyebabkan kerugian finansial bagi BPJS Kesehatan, memicu defisit anggaran, dan menggangu keberlangsungan program Jaminan Kesehatan Nasional (JKN). Tujuan penelitian ini adalah mengembangkan sistem yang dapat mendeteksi adanya potensi *inflated bills* secara otomatis guna mencegah terjadinya defisit anggaran tersebut. Metode yang digunakan dalam penelitian ini adalah algoritma *Extreme Gradient Boosting* (XGBoost). Data yang digunakan adalah data sampel klaim pelayanan kesehatan di FKRTL untuk periode 2016 hingga 2022 yang terdiri dari 1.176.438 baris. Data tersebut dibagi ke dalam subset *data training, data validation,* dan *data testing* dengan pemilihan 25 variabel berdasarkan Peraturan Menteri Kesehatan Republik Indonesia No. 3 Tahun 2023 tentang Standar Tarif Pelayanan Kesehatan dalam Program JKN. Proses *parameter tuning* dilakukan sebanyak 48 kali dengan menguji kombinasi parameter *n_estimators, learning_rate*, dan *max_depth*. Kombinasi terbaik dimiliki oleh *n_estimators* sebesar 100, *learning_rate* sebesar 0.1, dan *max_depth* sebesar 8 dengan nilai *Mean Absolute Error* (MAE) sebesar 0.005 pada *data training* dan 0.008 pada *data validation*. Hasil pengujian pada penelitian ini menunjukkan bahwa model memiliki nilai MAE sebesar 0.008, MSE sebesar 0.015, dan RMSE sebesar 0.12 yang diuji pada *data testing*. Nilai-nilai tersebut menunjukkan bahwa model memiliki tingkat kesalahan yang kecil dalam memprediksi biaya verifikasi dan dapat digunakan sebagai alat untuk mendeteksi potensi *inflated bills.*

Kata Kunci: *Inflated bills*, BPJS Kesehatan, *Extreme Gradient Boosting*

DETECTION OF POTENTIAL INFLATED BILLS IN BPJS HEALTH CLAIM DATA USING EXTREME GRADIENT BOOSTING (XGBOOST)

ABSTRACT

Inflated bills or bills that exceed the actual cost are a common type of fraud happen in advanced referral health facilities in the implementation of BPJS Heath program. Such fraud can lead to financial losses for BPJS Health, trigger budget deficits, and disrupt the sustainability of the National Health Insurance (JKN) program. This study aims to develop a system that capable to automatically detecting potential inflated bills to prevent such budget deficits. the data used consists of a sample of health service claims in FKRTL from 2016 to 2022, comprising 1,176,438 rows. The data is divided into training, validation, and testing subsets with 25 variables selected based on the Indonesian Minister of Health Regulation No. 3 of 2023 about Healthcare Tariff Standards In The JKN Program. The parameter tuning process was conducted 48 times, testing the combination of n_estimators, learning_rate, and max_depth parameters. The best combination achieved was n_estimators at 100, learning_rate at 0.1, and max_depth at 8 with Mean Absolute Error (MAE) of 0.005 on the training data and 0.008 on the validation data. The testing results in this study indicate that the model achieved an MAE of 0.008, MSE of 0.015, and RMSE of 0.12 on the testing data. These values demonstrate that the model has a low error rate in predicting the verification costs and can be used as a tool for detecting potential inflated bills.

Keywords*: *Inflated Bills, BPJS Health, Extreme Gradient Boosting

UCAPAN TERIMA KASIH

Segala puji dan syukur penulis panjakatkan kepada Allah SWT atas berkat, rahmat, dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi ini sebagai salah satu syarat dalam memperoleh gelar Sarjana Komputer di Program Studi S1 Teknologi Informasi Universitas Sumatera Utara dengan judul "Deteksi Potensi *Inflated Bills* Pada Data Klaim Layanan BPJS Kesehatan Menggunakan Algoritma *Extreme Gradient Boosting* (XGBOOST)".

Pencapaian ini tidak akan terwujud tanpa dukungan, bimbingan, bantuan, serta doa dari berbagai pihak. Oleh karena itu, penulis dengan sangat tulus mengucapkan terima kasih yang mendalam kepada:

1. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
2. Bapak Dedy Arisandi S.T., M.Kom selaku Ketua Program Studi S1 Teknologi Informasi Universitas Sumatera Utara sekaligus dosen penguji 1 yang sudah memberikan saran dan arahan yang berharga bagi penulis.
3. Ibu Ulfi Andayani S.Kom., M.Kom dan Bapak Ainul Hizriadi S.Kom., M.Sc selaku Dosen Pembimbing 1 dan 2 yang dengan sabar dan penuh dedikasi memberikan bimbingan, masukan berharga, serta membantu penulis dalam menyelesaikan skripsi ini dengan baik.
4. Bapak Ivan Jaya S.Si., M.Kom selaku dosen penguji 2 yang sudah memberikan saran dan arahan yang berharga bagi penulis.
5. Seluruh dosen, staff, dan pengawai Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara atas ilmu yang telah diberikan serta segala bantuan dalam hal administrasi maupun perkuliahan selama ini.
6. Kedua orang tua tercinta, Muhammad Taufik, S.H. (alm) dan Dra. Herlina Maharani, yang selalu memberikan doa dan kasih sayang tanpa henti. Penulis ingin mengucapkan terima kasih atas cinta, pengorbanan, dan dukungan yang tidak ternilai sepanjang hidup penulis.

7. Saudara kandung penulis, Aqfie Jumidhal Ardha, S.T. dan Randy Hisyam Abya, yang selalu memberikan semangat, dukungan, dan doa yang tulus selama ini. Keberadaan kalian sangat berarti bagi penulis.
8. Sahabat Penulis, Annisa Putri S.Pd dan Rahma Alia A.Md.RMIK untuk banyak masukan, pengertian, serta dukungan emosional yang diperlukan selama proses penulisan skripsi. Terima kasih atas kebersamaan dan kebahagiaan yang kita bagi.
9. Wanda Khalishah yang telah menjadi rekan dalam penulisan skripsi ini. Dukungan dan kenangan tersebut tidak akan penulis lupakan.
10. Seluruh rekan dari Program Studi S1 Teknologi Informasi Angkatan 2020, khususnya KOM A.
11. Seluruh pihak yang belum disebutkan, namun telah memberikan dukungan dan motivasi dalam menyelesaikan skripsi ini. Setiap bantuan kalian sangat berarti bagi penulis.

Penulis menyadari bahwa skripsi ini masih memiliki banyak kekurangan. Oleh karena itu, penulis sangat mengharapkan masukan dan kritik yang membangun guna memperbaiki dan menyempurnakan skripsi ini. Semoga skripsi ini dapat membawa manfaat dan kontribusi positif untuk semua pihak yang membutuhkan.

Medan, 8 Januari 2025

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR PERSAMAAN

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Setiap orang di Indonesia memiliki hak yang melekat dan tidak dapat dicabut untuk mendapatkan kesehatan yang baik. Hak atas kesehatan fisik dan mental, perumahan, dan lingkungan yang layak dijamin bagi semua warga negara Republik Indonesia oleh undang-undang (Negara Republik Indonesia, 1945). Oleh karena itu, pemerintah Indonesia menetapkan program Jaminan Kesehatan Nasional (JKN) untuk memastikan bahwa semua warga negara memiliki akses terhadap perawatan kesehatan dasar yang memadai. BPJS Kesehatan, juga dikenal sebagai Badan Penyelenggara Jaminan Sosial Kesehatan, adalah badan resmi di Indonesia yang bertanggung jawab kepada Presiden dan bertugas untuk melaksanakan program JKN yang menyediakan asuransi kesehatan untuk semua warga negara (Adinda et al., 2022).

Peluncuran resmi BPJS Kesehatan adalah pada 1 Januari 2014, dan jumlah pesertanya terus meningkat sejak saat itu. Untuk layanan nonspesialis, BPJS Kesehatan bekerja sama dengan Fasilitas Kesehatan Rujukan Primer (FKTP), dan untuk layanan ahli dan subspesialis, bermitra dengan Fasilitas Kesehatan Rujukan Tingkat Lanjut (FKRTL). Menurut data BPJS Kesehatan (2024) per 31 Januari 2024, jumlah peserta BPJS Kesehatan adalah 267.784.196 orang, fasilitas kesehatan rujukan tingkat pertama sebanyak 23.250, dan fasilitas kesehatan rujukan tingkat lanjut sebanyak 3.095. Keberadaan BPJS Kesehatan memberikan dampak positif bagi masyarakat, terlebih pada masyarakat kurang mampu. Diantaranya dengan mengurangi pengeluaran masyarakat dalam mengakses layanan kesehatan di Indonesia (Eka et al., 2020).

Namun, program BPJS Kesehatan tidak selalu berjalan dengan baik. Berdasarkan laporan analisis keuangan BPJS Kesehatan pada tahun 2023 lalu, BPJS Kesehatan dinyatakan kembali mengalami defisit anggaran hingga mencapai Rp7,9 triliun (Kompas, 2024). Defisit BPJS Kesehatan terjadi akibat adanya ketidakcocokan dan ketidakefektifan antara jumlah tanggungan jaminan kesehatan oleh BPJS Kesehatan dan uang yang diperoleh dari setoran kepersertaan (Raisa et al., 2021).

Salah satu penyebab defisit anggaran BPJS Kesehatan adalah adanya tindakan kecurangan atau praktik *fraud* dalam pelaksanaan program tersebut. Tindakan yang disengaja untuk melanggar ketentuan demi memperoleh keuntungan dari program Jaminan Kesehatan Nasional (JKN) melalui kecurangan adalah salah satu bentuk penipuan (*fraud*) dalam program tersebut (Peraturan Menteri Kesehatan Nomor 16 Tahun 2019). Pemalsuan tagihan merupakan salah satu bentuk kecurangan atau penipuan yang umum terjadi di Fasilitas Kesehatan Rujukan Tingkat Lanjut (FKRTL) (Peraturan Menteri Kesehatan Nomor 16 Tahun 2019). *Inflated* bills merupakan jenis pemalsuan tagihan dilakukan oleh fasilitas kesehatan dan rumah sakit dengan mengenakan biaya lebih dari yang seharusnya untuk biaya pengobatan dan alat kesehatan (Gede et al., 2024).

Pembayaran klaim BPJS Kesehatan merupakan prosedur yang rumit. BPJS kesehatan mengandalkan para verifikator untuk memeriksa keabsahan klaim yang diterima. Proses tersebut bertujuan untuk memastikan bahwa layanan yang diberikan kepada pasien yang terdaftar sebagai peserta BPJS Kesehatan telah memenuhi standar dan prosedur yang telah ditetapkan, mengelola biaya, mencegah penyalahgunaan dana anggaran, dan mendeteksi potensi kecurangan. Namun, karena masih dilakukan secara konvensional, prosedur verifikasi dapat memakan waktu yang lama. Verifikasi manual dan pendekatan tradisional lainnya tidak mampu mengimbangi jumlah klaim dan data yang terus meningkat. Di sisi lain, diperlukan percepatan proses penyelesaian klaim dengan tim verifikator yang kecil (Rofiq, 2023). Dengan demikian, diperlukan sistem yang dapat memeriksa klaim secara otomatis dan mengidentifikasi adanya kecurangan dengan tepat dan akurat.

Terdapat beberapa penelitian terdahulu yang membahas tentang identifikasi dan prediksi praktik inefisiensi dan *fraud* dalam klaim layanan yang diajukan oleh rumah sakit atau fasilitas kesehatan. Penelitian terkait dilakukan oleh Kevin et al. (2023) menggunakan algoritma *Light Gradient Boosting Machine* (LGBM) dalam

memprediksi inefisiensi pada klaim BPJS Kesehatan menggunakan algoritma LGBM. Hasil studi tersebut menunjukkan bahwa algoritma LGBM berhasil mendapatkan skor *Area Under Curve* (AUC) sebesar 0,75.

Penelitian lain dilakukan oleh Rofiq (2023) dengan menggunakan algoritma *machine learning* seperti *Gradient Boosting Classifier, Decision Tree, Random Forest,* SVM*, Naive Bayes, CatBoost,* dan *XGBoost* untuk mendeteksi klaim yang tidak efisien yang diajukan ke BPJS Kesehatan oleh pihak fasilitas kesehatan. Hasil dari penelitian tersebut menunjukkan bahwa algoritma *Random Forest* yang dikombinasikan dengan teknik *Tomek Links* adalah yang terbaik dengan *F1 Score* sebesar 19,53 dan terdapat lima variabel yang paling berpengaruh terhadap inefisiensi klaim diantaranya fasilitas medis, usia peserta, diagnosa awal, diagnosa primer peserta, dan tipe sarana medis.

Demikian pula, Alan & Mohammad (2023) menggunakan algoritma SVM dan XGBoost untuk menyelidiki data terkait penipuan klaim asuransi kesehatan. Dengan nilai *recall* sebsear 0,9994 dan *balance accuracy* sebesar 0,9995, metode XGBoost mengungguli SVM dalam penelitian ini. Penelitian lain oleh Humasak et al. (2021) menggunakan metode Jaringan Syaraf Tiruan dalam pendeteksian penipuan dalam klaim layanan rumah sakit. Metode jaringan syaraf tiruan yang dibangun mencapai nilai *recall* sebesar 78% dan nilai akurasi dan presisi di atas 50%.

Berdasarkan latar belakang tersebut dan beberapa penelitian sebelumnya dapat dinyatakan bahwa kasus *inflated bills* pada praktik BPJS Kesehatan merupakan masalah serius yang dapat menyebabkan defisit anggaran BPJS Kesehatan. Adapun algoritma yang diusulkan dalam penelitian ini adalah *Extreme Gradient Boosting*. Algoritma *Extreme Gradient Boosting* (XGBoost) dipilih dalam membangun sistem verifikasi klaim dan deteksi potensi *inflated bills* mengingat volume data klaim BPJS Kesehatan yang besar dan kebutuhan untuk memverifikasikannya dengan cepat. Selain itu, algoritma *Extreme Gradient Boosting* (XGBoost) juga memiliki kemampuan untuk mengidentifikasi fitur penting suatu data (Rimbun et al., 2022). Dengan adanya kemampuan tersebut, algoritma *Extreme Gradient Boosting* (XGBoost) dapat meninjau pola atau indikator yang mengarah pada potensi *inflated bills* dengan lebih baik. Implementasi sistem tersebut diharapkan mampu meningkatkan efisiensi dalam mendeteksi potensi *inflated bills* pada data klaim layanan BPJS Kesehatan sehingga dapat membantu mencegah potensi defisit anggaran kedepannya. Oleh karena itu, penulis memutuskan untuk melakukan penelitian dan eksplorasi dalam melakukan

deteksi potensi *inflated bills* pada data klaim BPJS Kesehatan dengan memanfaatkan algoritma *Exterme Gradient Boosting*.

1.2 Rumusan Masalah

Praktik *inflated bills* merupakan suatu permasalahan serius dalam implementasi program Jaminan Kesehatan Nasional (JKN) oleh BPJS Kesehatan yang dapat menyebabkan kerugian finansial, menghambat jalannya program, serta defisit anggaran BPJS Kesehatan. Tantangan utama adalah terdapat kompleksitas pada data klaim layanan dengan adanya variabel yang beragam. Oleh karena itu itu, dibutuhkan sebuah sistem yang secara otomatis bisa mendeteksi potensi *inflated bills* pada data klaim layanan BPJS Kesehatan menggunakan algoritma *Extreme Gradient* Boosting (XGBoost) sehingga manajemen BPJS Kesehatan dapat *monitoring* anggaran yang diberikan dengan lebih baik.

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah mendeteksi potensi *inflated bills* pada data klaim layanan BPJS Kesehatan menggunakan algoritma *Extreme Gradient Boosting* (XGBoost).

1.4 Batasan Masalah

Batasan masalah dalam penelitian ini, antara lain:

1. Dataset yang digunakan merupakan dataset sampel BPJS Kesehatan untuk periode 2016-2022.
2. Data yang digunakan merupakan data sampel histori klaim layanan pengobatan di Fasilitas Kesehatan Rujukan Tingkat lanjut (FKRTL).
3. Menggunakan 25 variabel sebagai indikator penentuan *inflated bills* diantaranya, tipe FKRTL, tingkat pelayanan, kelas iuran, kode ICD-10, kode INACBGs, jenis prosedur, tarif regional INACBGs, *group* tarif INACBGS, kode dan tarif-tarif tambahan, biaya tagih, lama masa rawat (*length of stay*), biaya verifikasi dan lain sebagainya.

1.5 Manfaat Penelitian

Manfaat dalam penelitian ini, antara lain:

1. Memberikan kontribusi terhadap bidang keilmuan *machine learning* dan *data science*, khususnya dalam sektor layanan kesehatan.
2. Menerapkan algoritma *Extreme Gradient Boosting* (XGBoost) dalam melakukan regresi biaya verifikasi yang digunakan sebagai alat untuk mendeteksi potensi *inflated bills* pada data klaim layanan BPJS Kesehatan.

1.6 Metodologi Penelitian

Beberapa tahap yang akan dilakukan dalam penelitian ini adalah:

1. Studi Pustaka

   Penelitian ini dimulai dengan pencarian sumber informasi yang bervariasi dan kredibel serta melakukan tinjauan pustaka dari berbagai jenis referensi, termasuk buku, undang-undang, jurnal, *e-book*, dan artikel yang berkaitan dengan topik *inflated bills*, klaim dan peraturan BPJS Kesehatan, algoritma *Extreme Gradient Boosting* (XGBoost).
2. Analisis Permasalahan

   Pada tahap analisis permasalahan, penulis melakukan analisis dan pemahaman konsep dari algoritma *Extreme Gradient Boosting* dan *inflated bills* berdasarkan studi pustaka yang telah dilakukan sebelumnya.
3. Pengumpulan Data

   Data historis pelayanan kesehatan BPJS Kesehatan diperoleh melalui *situs* portal data Jaminan Kesehatan nasional (JKN). Data tersebut mencakup informasi detail mengenai klaim layanan kesehatan yang telah diajukan oleh fasilitas kesehatan.
4. Perancangan Sistem

   Pada tahap ini, penulis melakukan perancangan sistem seperti arsitektur umum untuk algoritma *Extreme Gradient Boosting* dan desain antarmuka pengguna berdasarkan analisis permasalahan yang telah dilakukan sebelumnya.

5. Implementasi

   Pada tahap ini, sistem yang telah dirancang sebelumnya diimplementasikan sehingga menghasilkan sebuah sistem yang sesuai dengan tujuan penelitian dengan menggunakan teknik *Extreme Gradient Boosting* berbasis *Python.*

6. Pengujian

   Pada tahap ini, penulis melakukan pengujian terhadap sistem yang telah dibangun untuk melihat seberapa baik kinerja sistem dalam mendeteksi potensi *inflated bills* pada data klaim layanan BPJS Kesehatan menggunakan algoritma *Extreme Gradient Boosting* (XGBoost). Proses pengujian tersebut dilakukan dengan menggunakan beberapa metrik evaluasi

7. Dokumentasi

   Langkah terakhir dalam penelitian ini ialah dokumentasi. Pada tahap ini, seluruh rangkaian penelitian, mulai dari tahap studi pustaka hingga tahap penyelesaian akan didokumentasikan dalam format skripsi.

1.7 Sistematika Penulisan

Berikut ini merupakan sistematika penulisan skripsi ini yang terdiri dari beberapa bab utama diantaranya seperti yang dijelaskans sebagai berikut:

BAB I PENDAHULUAN

Bab ini memuat tentang latar belakang penelitian dengan judul "Deteksi Potensi *Inflated Bills* pada Data Klaim Layanan BPJS Kesehatan Menggunakan Algoritma *Extreme Gradient Boosting* (XGBoost)", perumusan masalah, batasan dalam masalah, tujuan dari penelitian, manfaat dari penelitian, metode penelitian, dan sistematika penulisan.

BAB 2 LANDASAN TEORI

Bab ini berisi tentang teori-teori yang relevan dengan penelitian ini seperti praktik *inflated bills* pada data klaim layanan BPJS Kesehatan, *machine learning,* teori dalam *preprocessing data*, Teknik *Ensemble Learning* seperti *Boosting, Stacking,* dan *Bagging,* algoritma *Extreme Gradient Boosting*, serta metrik evaluasi seperti *Mean Squared Error* (MSE), *Root Mean Squared Error* (RMSE), dan *Mean Absolute Error* (MAE).

BAB 3 ANALISIS DAN PERANCANGAN SISTEM

Bab ini memuat informasi dan penjelasan mengenai data yang digunakan, arsitektur umum, proses *preprocessing* yang dilakuakan pada data, pembelajaran algoritma, serta rancangan desain antarmuka sistem.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Metodologi, alat, dan teknik yang digunakan selama proses analisis dan desain sistem, serta hasil evaluasi kinerja model, dirinci dalam bab ini.

BAB 5 KESIMPULAN

Penelitian yang telah selesai dirangkum dalam bab terakhir ini, yang juga mencakup saran untuk peluang dalam penelitian lainya.

BAB 2

LANDASAN TEORI

2.1 BPJS Kesehatan

BPJS Kesehatan adalah sebuah lembaga resmi di Indonesia yang memiliki tanggung jawab kepada Presiden dan memiliki kewajiban dalam melaksanakan program JKN yang memberikan jaminan atau tanggungan kesehatan bagi seluruh warga negara Indonesia (Adinda et al., 2022). Semua warga negara Indonesia dianggap sebagai anggota BPJS Kesehatan sesuai dengan nilai-nilai organisasi yang dianutnya, yaitu keadilan sosial, manfaat, dan kemanusiaan. Setiap warga negara Indonesia berhak atas kehidupan yang layak, dan BPJS Kesehatan berupaya untuk memastikan bahwa setiap orang mendapatkannya.

2.2 *Inflated Bills*

Praktik *Inflated bills* di Fasilitas Kesehatan Rujukan Tingkat Lanjut (FKRTL) merupakan salah satu bentuk penipuan Jaminan Kesehatan Nasional (JKN) (Peraturan Menteri Kesehatan Nomor 16, 2019). Pembengkakan tagihan atau yang dikenal juga dengan penggelembungan biaya farmasi dan alat kesehatan terjadi karena biaya obat dan alat kesehatan yang sebenarnya dibebankan terlalu tinggi. (Gede et al., 2024). *Inflated bills* terjadi ketika penyedia layanan kesehatan menyediakan produk dengan harga murah namun mengklaimnya dengan harga yang lebih tinggi (Sarah et al., 2021).

Kecurangan dalam implementasi program BPJS Kesehatan tersebut menjadi salah satu penyebab terjadinya defisit BPJS Kesehatan. Defisit keuangan BPJS Kesehatan merupakan situasi di mana BPJS Kesehatan harus menanggung biaya jaminan kesehatan nasional dibandingkan dengan pendapatan yang diterimanya dari iuran peserta. (Raisa et al., 2021).

2.3 *Data Preprocessing*

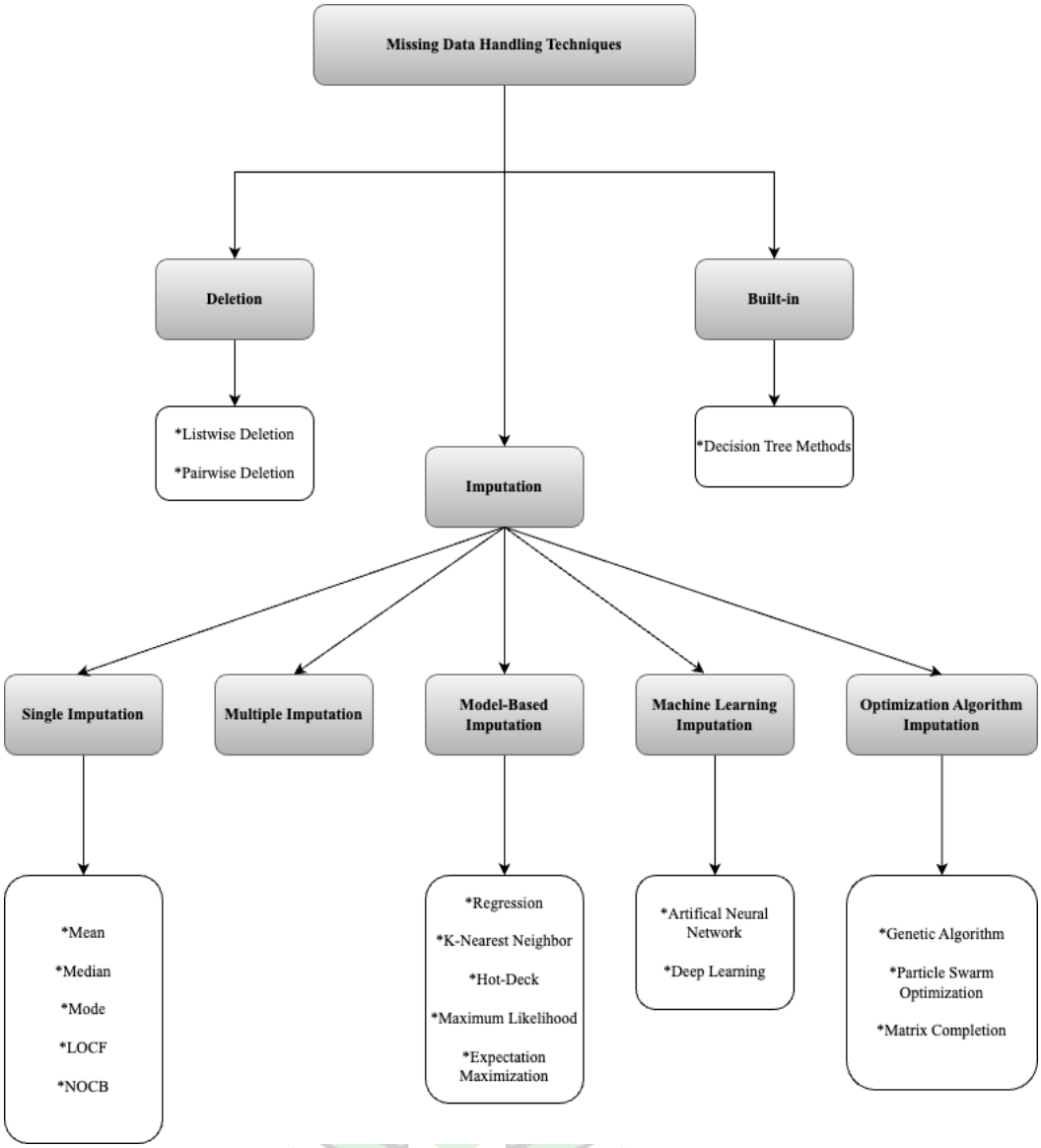
Data preprocessing merupakan langkah awal dalam pengolahan data. Proses ini bertujuan untuk mengkonversi data mentah atau data asli menjadi data yang lebih baik secara kualitas sehingga layak untuk diolah pada tahap selanjutnya (Alghifari & Juardi, 2021). Proses ini penting untuk memastikan bahwa data yang diolah sudah bersih, berkualitas dan terstruktur sehingga tahapan selanjutnya dapat berjalan dengan baik. Terdapat beberapa teknik *data preprocessing* yang diimplementasikan dalam penelitian ini, antara lain:

2.3.1 *Data Cleaning*

Data cleaning atau proses pembersihan data merupakan tahapan yang melibatkan perbaikan masalah atau *error* sistematis dalam data yang berantakan (Jason, 2020). Terdapat beberapa langkah yang dapat diambil untuk membersihkan data diantaranya, menggunakan statistik untuk mendefinisikan data normal dan mengidentifikasi *outliers*, mengidentifikasi kolom-kolom yang memiliki nilai sama atau tidak ada varian dan menghapusnya, mengidentifikasi dan menghapus baris yang duplikat, menandai data yang kosong sebagai *missing value,* serta mengisi *empty value*.

a. *Missing Values*

   Missing values merupakan kondisi dimana terdapat nilai yang hilang dalam sebuah dataset. *Missing values* merupakan hal yang umum terjadi pada suatu data akibat beberapa faktor seperti kerusakan alat, perhitungan yang kurang tepat, data yang tidak terdokumentasi dengan baik, serta beberapa masalah teknis lainnya (Prasetya et al., 2023). Keberadaan *missing values* dapat mempengaruhi kualitas dari suatu model yang akan dibangun sehingga diperlukan teknik penanganan *missing values* yang tepat. Terdapat beberapa teknik dalam menangani *missing values* diantaranya penghapusan (*deletion*), imputasi (*imputation*), dan juga *built-in* (Joel et al., 2022). Ilustrasi teknik penanganan *missing values* terlihat seperti Gambar 2.1 dibawah ini:



Gambar 2.1 Teknik Penanganan *Missing Values* (Joel et al., 2022)

b. *Duplicated Values*

Duplicate values atau nilai duplikat adalah kondisi dimana terdapat data dengan nilai yang identik atau serupa dalam sebuah dataset. *Duplicated values* dalam dataset dapat membahayakan keunikan dari data (Zhao et al., 2024). Keberadaan *duplicated values* tersebut dapat menggangu keunikan dan integritas data serta memiliki potensi untuk menimbulkan masalah pada hasil analisisnya. Metode yang umum digunakan dalam mengatasi *duplicate values* adalah dengan menghapus setiap entri data yang identik atau duplikat.

2.3.2 *Feature Engineering*

Feature Engineering melibatkan pengambilan data mentah dan ditransformasi menjadi format yang mungkin digunakan model pembelajaran mesin dengan mengekstraksi atribut. (Sitorus et al., 2020). *Feature engineering* bertujuan untuk meningkatkan kinerja model dalam melakukan prediksi dan klasifikasi. Menurut Sinan & Divya (2018), terdapat beberapa aspek dalam *feature engineering,* diantaranya:

a. *Feature Understanding*

   Feature understanding atau melakukan pemahaman terhadap data berdasarkan kondisi kualitas dan kuantitasnya dengan cara analisis data terstruktur dan tidak terstruktur, mengidentifikasi nilai yang hilang (*missing value*), eksplorasi data, statistik deskriptif, dan visualisasi data.

b. *Feature Improvement*

   Feature Improvement bertujuan untuk meningkatkan kualitas fitur-fitur dalam dataset. Langkah pertama yang diambil adalah membersihkan data yang mencakup penghapusan atau perbaikan data yang tidak *valid*, duplikat, atau mengandung kesalahan, sehingga dataset menjadi lebih akurat dan dapat diandalkan. Selanjutnya, langkah ini juga melibatkan pengisian nilai yang kosong (*missing values*) agar tidak mengganggu proses analisis dan pembuatan model. Pengisian ini dapat dilakukan dengan berbagai metode, seperti imputasi berdasarkan rata-rata, *median*, atau menggunakan algoritma tertentu. Selain itu, transformasi data yang tidak terstruktur, seperti teks atau gambar yang dilakukan untuk mengubahnya menjadi format yang lebih sesuai untuk analisis dan model. Selain itu, pada proses ini juga dapat dilakukan normalisasi data untuk mengatasi *range* data yang jauh.

c. *Feature Selection*

   Feature selection merupakan proses pemilihan fitur yang sesuai dengan model yang ingin dibangun. Proses ini bertujuan untuk meningkatkan kualitas dari model. Terdapat beberapa metode dalam melakukan *feature selection* diantaranya korelasi koefisien, mengidentifikasi dan menghapus multikolinearitas, *chi-squared test, anova test,* interpretasi nilai *p-value*, pemilihan fitur berulang, serta menggunakan *machine learning* untuk mengukur entropi dan mendapatkan informasi tambahan.

d. *Feature Construction*

Pada tahap ini diciptakan fitur-fitur baru dan dimasukkan ke dalam dataset. Fitur-fitur baru tersebut akan menambah informasi dan pola baru yang dapat digunakan oleh *machine learning* untuk dieksplorasi sehingga dapat meningkatkan kualitas kinerja model yang akan diciptakan.

e. *Feature Transformation*

Feature transformation atau transformasi fitur merupakan tahapan mengekstraksi struktur laten atau tersembunyi dari *dataset* dengan tujuan mengubah *dataset* secara matematis menjadi *dataset* yang lebih baik.

f. *Feature Learning*

Tahap *feature learning* merupakan tahapan yang melibatkan *machine learning* untuk melihat data dari sudut pandang baru yang akan membuka masalah baru untuk dipecahkan.

2.3.3 *Label Encoding*

Proses pengubahan data kategorikal dari teks menjadi angka disebut dengan *label encoding* (Cevi et al., 2024). Salah satu implementasi yang populer adalah fitur *label encoder* yang tersedia dalam *library scikit-learn*. Fitur tersebut dapat mengubah data kategorik menjadi numerik dalam satu kolom. Tabel 2.1 menunjukkan penerapan dari *label encoding*, yaitu:

Tabel 2.1 Contoh *Label Encoding*

Sebelum *Label Encoding*	**Setelah *Label Encoding***
Kelas I	0
Kelas II	1
Kelas III	2

2.3.4 *Normalization*

Normalization atau normalisasi data merupakan proses memodifikasi nilai dalam data sehingga berada di rentang yang sama sehingga dapat mempermudah proses analitik (Muhammad et al., 2022). Normalisasi data bertujuan untuk mengatasi dua permasalahan utama dalam proses algoritma pembelajaran mesin, yaitu adanya fitur

dominan dan *outlier* (Dalwinder & Birmohan, 2020). Terdapat beberapa teknik untuk menormalisasi data diantaranya *min-max , z-score,* dan juga *decimal scaling.*

a. *Min-Max Normalization*

Metode *Min-Max Normalization* bertujuan untuk mengubah *range* atau rentang nilai menjadi 0 dan 1 (Inggih & Febi, 2022). *Min-Max Normalization* dilakukan dengan cara mengurangi nilai paling kecil atau minimum dari setiap data, kemudian membaginya dengan selisih antara nilai paling besar atau maksimum dan nilai paling kecil atau minimum dalam dataset. Dengan demikian, semua nilai akan terdistribusi dalam rentang yang lebih terkontrol. Persamaan yang digunakan untuk melakukan *Min-Max Normalization* ditunjukkan pada Persamaan 2.1, yaitu:

$$x' = \frac{x_i - min(x)}{max(x) - min(x)} \tag{2.1}$$

Dimana x' adalah nilai dihasilkan selama proses normalisasi, x_i adalah nilai yang akan dinormalisasi, nilai yang paling rendah ditunjukkan oleh *min(x)* , dan nilai yang paling tinggi dari keseluruhan data disebut *max(x)*.

b. *Z-Score*

Metode *Z-score* atau *zero mean normalization* adalah teknik untuk menormalkan data berdasarkan nilai rata-rata dan standar deviasi pada setiap atribut. Teknik ini bertujuan agar proses penskalaan fitur mendekati standar yang terdistribusi secara normal (Fikri et al., 2021). Proses *Z-score normalization* dilakukan melalui beberapa tahap perhitungan. Pertama, langkah awal adalah menghitung nilai rata-rata ($\bar{x}$) dari atribut yang akan dinormalisasi. Nilai rata-rata ini diperoleh dengan menambahkan semua nilai dalam atribut tersebut, kemudian membaginya dengan jumlah data observasi. Rumus matematis dari rata-rata disajikan pada Persamaan 2.2 dibawah ini:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.2}$$

Setelah mendapatkan nilai rata-rata, perhitungan *z-score* dilanjutkan dengan perhitungan nilai dari standar deviasi (σ). Standar deviasi menunjukkan seberapa jauh nilai-nilai dalam kumpulan data menyebar dari nilai rata-ratanya. Nilai sampel dapat diperoleh dengan menggunakan rumus di mana setiap nilai dikurangi dari rata-rata, kemudian dikuadratkan hasilnya, dijumlahkan semua kuadrat tersebut, dan kemudian dibagikan dengan jumlah total data dikurangi satu. Rumus matematis dari standar deviasi disajikan pada Persamaan 2.3 dibawah ini:

$$\boldsymbol{\sigma} = \sqrt{\frac{\mathbf{1}}{\boldsymbol{n} - \mathbf{1}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}} (\boldsymbol{x_i} - \overline{\boldsymbol{x}})^{\mathbf{2}}} \tag{2.3}$$

Proses normalisasi dapat dilakukan setelah kedua nilai ini diperoleh. Setelah kedua nilai ini diperoleh, proses normalisasi dapat dilakukan. Nilai baru (x') untuk setiap data dalam atribut dihitung dengan menggunakan Persamaan 2.4 berikut:

$$\boldsymbol{x'} = \frac{\boldsymbol{x_i} - \overline{\boldsymbol{x}}}{\boldsymbol{\sigma}} \tag{2.4}$$

Dimana nilai yang dihasilkan selama proses normalisasi ditunjukkan oleh x', nilai asli yang akan dinormalisasi dilambangkan oleh x_i, nilai rata-rata dari seluruh data observasi atau atribut data dilambangkan dengan $\bar{x}$, dan nilai standar deviasi dari atribut yang bersangkutan dilambangkan dengan σ.

c. *Decimal Scaling*

Metode *decimal scaling* merupakan teknik normalisasi data yang memindahkan nilai atau posisi desimal data ke arah yang diinginkan sehingga nilai-nilai tersebut berada pada rentang yang dinginkan, umumnya -1 dan 1. Dalam metode ini, nilai-nilai dalam dataset diubah dengan cara membagi angka tersebut dengan 10 pangkat j (10^j), di mana j adalah jumlah digit terbesar dari nilai maksimum dalam *dataset*. *Decimal scaling* ditentukan menggunakan persamaan 2.5, yaitu:

$$v'_i = \frac{v_i}{10^j} \tag{2.5}$$

Dimana v'_i melambangkan hasil atau nilai baru yang didapat dari proses normalisasi, v_i melambangkan nilai atribut, dan *j* melambangkan bilangan bilat terkecil hingga maksimum (|v'|)<1

2.4 *Machine Learning*

Soebroto (2019) mendefinisikan *machine learning* atau pembelajaran mesin sebagai bagian dari *artificial intelligence* yang berfokus pada pembelajaran dari data atau berkonsentrasi pada pengembangan sistem secara otomatis tanpa memerlukan pemrograman berulang. Pengertian lain dari *machine learning* menurut Dios (2022) adalah sebuah disiplin ilmu yang mempelajari algoritma komputer yang dirancang untuk mengenali pola dalam data, dengan tujuan untuk mengkonversi berbagai data menjadi tindakan konkret dengan seminim mungkin keterlibatan manusia

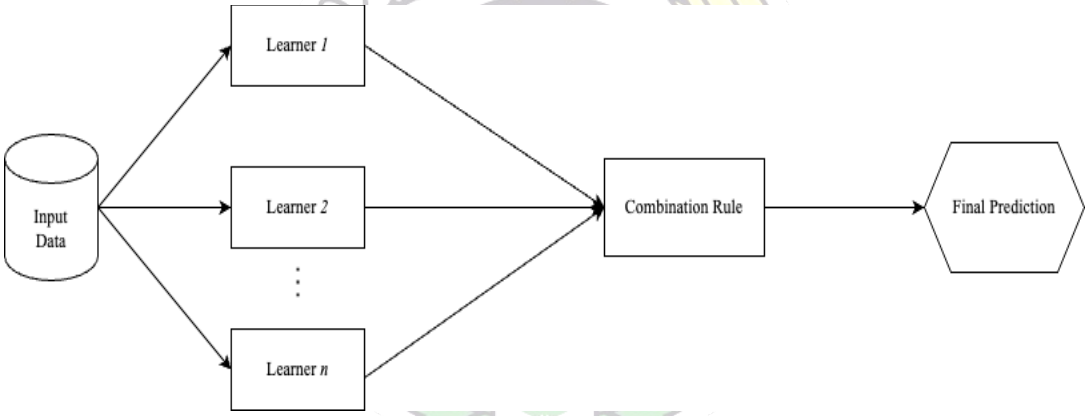
Machine learning dikategorikan mejadi tiga di antaranya adalah s*upervised learning, unsupervised learning,* dan *reinforcement learning,* dimana setiap kategori tersebut memiliki tipe *dataset* dan karakteristik yang berbeda. *Supervised learning* adalah tipe pertama dalam *machine learning* yang memanfaatkan data latih (*training*) yang telah memiliki label untuk melakukan pembelajaran pada mesin. Hal tersebut dilakukan agar mesin dapat mengidentifikasi input berdasarkan fitur yang tersedia untuk membentuk sebuah prediksi maupun klasifikasi (Endang & Rully, 2020).

Unsupervised learning merupakan metode *machine learning* yang bersifat deskriptif. Teknik umumnya digunakan untuk mengelompokkan, mengkategorikan, dan mengklasifikasikan data. Algoritma ini tidak memiliki *training dataset* karena tidak bersifat prediktif, sehingga membutuhkan pembelajaran dari data yang telah ada (Puput et al., 2021). *Unsupervised learning* berkaitan dengan data yang tidak memiliki label. *Reinforcement learning* merupakan salah satu pendekatan dalam teori pembelajaran mesin dimana agen dapat mempelajari melalui tindakan yang diambilnya dan juga hasil dari tindakannya serta berusaha untuk memaksimalkan *reward* yang diterima melalui interaksi lingkungan berupa *reward* bernilai positif atau negatif (Dharma & Tambunan, 2021).

2.5 *Ensemble Learning*

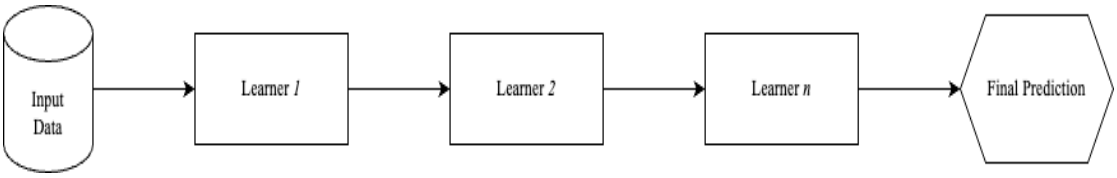
Salah satu metode pembelajaran mesin atau *machine learning* adalah *ensemble learning* yang memungkinkan pelatihan beberapa model untuk menyelesaikan masalah yang sama. Kemudian hasil dari model-model tersebut digabungkan untuk mencapai hasil yang lebih optimal (Linggar & Adi, 2022). Tujuan utama dari *ensemble learning* adalah menggabungkan beberapa model yang lemah untuk mendapatkan model dengan tingkat akurasi lebih tinggi.

Terdapat dua cara dalam menggabungkan model dalam ensemble learning yakni, *paralel* dan *sequential.* Metode *paralel* melatih beberapa dasar (*base*) klasifikasi secara mandiri lalu menggabungkan seluruh model tersebut. Alur proses *paralel* diilustrasikan seperti pada Gambar 2.2 dibawah ini:



Gambar 2.2 Proses Paralel Pada *Ensemble Learning* (Mienye & Sun, 2022)

Sementara itu metode *sequential* melatih model dasar (*base model)* secara berurutan sehingga model dalam setiap iterasi dapat memperbaiki kesalahan atau *error* yang dibuat oleh model sebelumnya (Mienye & Sun, 2022). Alur proses *sequential* diilustrasikan seperti pada Gambar 2.3 dibawah ini:

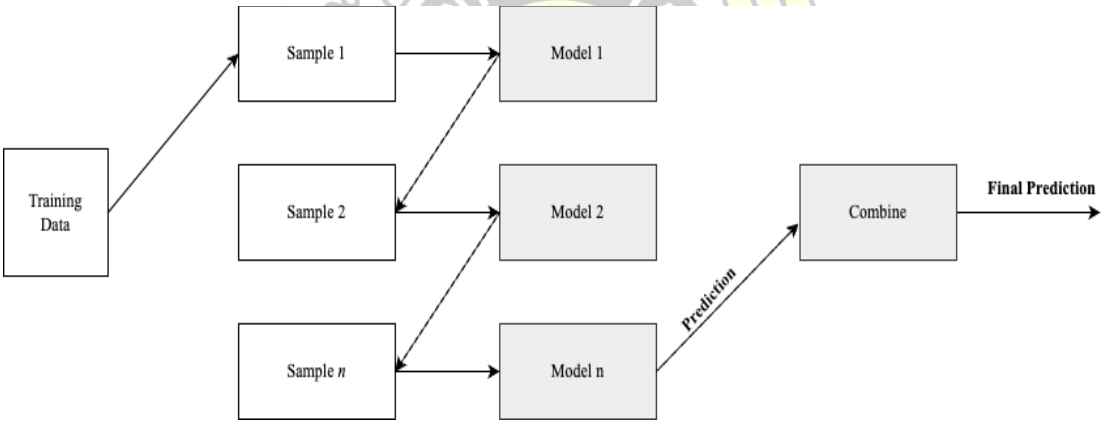


Gambar 2.3 Proses *Sequential* Pada *Ensemble Learning* (Mienye & Sun, 2022)

Ensemble learning terbagi menjadi 3 yakni *boosting, bagging,* dan *stacking* yang masing-masing menggunakan pendekatan yang berbeda dalam menggabungkan model-model dasar (*base model*) menjadi model prediktif yang lebih baik.

2.5.1 *Boosting*

Boosting merupakan salah satu tipe dari *ensemble learning* yang dapat dimanfaatkan untuk menghasilkan model prediksi atau klasifikasi. Metode ini hanya menggunakan satu *base model* atau model dasar dan proses pembelajarannya dilakukan secara berurutan (*sequential)* dan adaptif, di mana hasil model bergantung pada model dasar sebelumnya. Setelah itu, seluruhnya digabungkan untuk mendapatkan hasil yang terbaik (Linggar & Adi, 2022). Ilustrasi dari metode *boosting* terlihat seperti yang disajikan Gambar 2.4, yaitu:



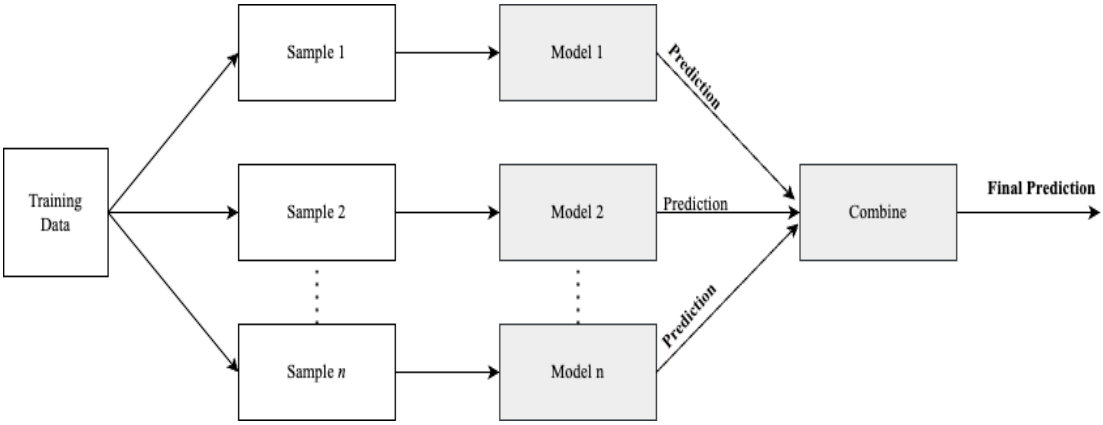
Gambar 2.4 *Boosting Ensemble Learning* (Linggar & Adi, 2022)

Terdapat beberapa model dari algoritma *boosting* diantaranya, *Adaptive Boosting (*AdaBoost), *Gradient Boosting Machines* (GBM), *Extreme Gradient Boosting Machine* (XGBoost), *Light Gradient Boosting Machine* (LGBM), dan CatBoost.

2.5.2 *Bagging*

Bagging atau *bootstrap aggregating* merupakan metode *ensemble learning* yang menggunakan satu tipe model dasar atau *base model* serta melakukan pembelajaran secara paralel dan independen di setiap *base model* lalu digabungkan untuk

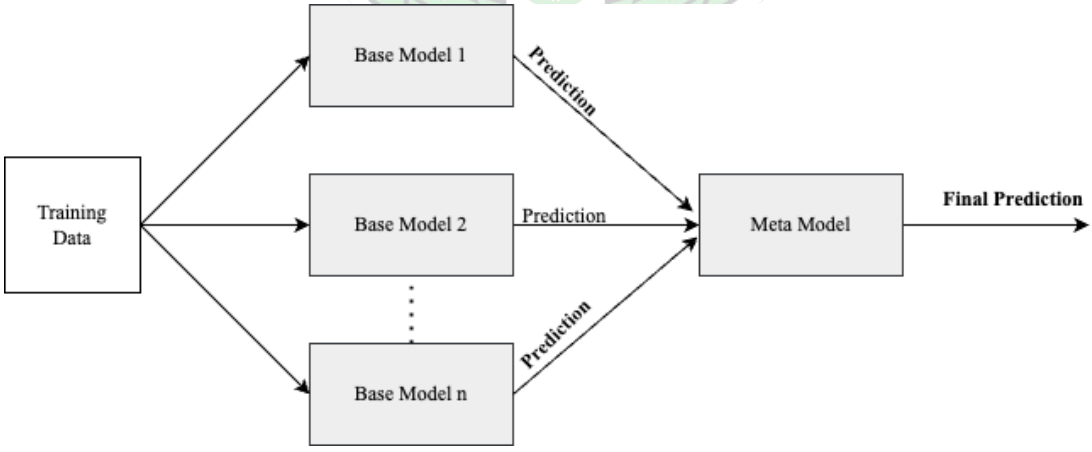
mendapatkan hasil yang terbaik. Ilustrasi proses *bagging* terlihat seperti yang disajikan Gambar 2.5 berikut:



Gambar 2.5 *Bagging Ensemble Learning* (Linggar & Adi, 2022)

2.5.3 *Stacking*

Stacking merupakan metode *ensemble learning* yang menggunakan beberapa model dasar atau *base model* untuk melakukan pembelajaran secara mandiri dan paralel. Setelah itu, seluruh model dasar atau *base model* digabungkan menggunakan algoritma *meta-learning* untuk menghasilkan *output* akhir (Linggar & Adi, 2022) Ilustrasi proses metode *stacking* terlihat seperti pada Gambar 2.6, yaitu:



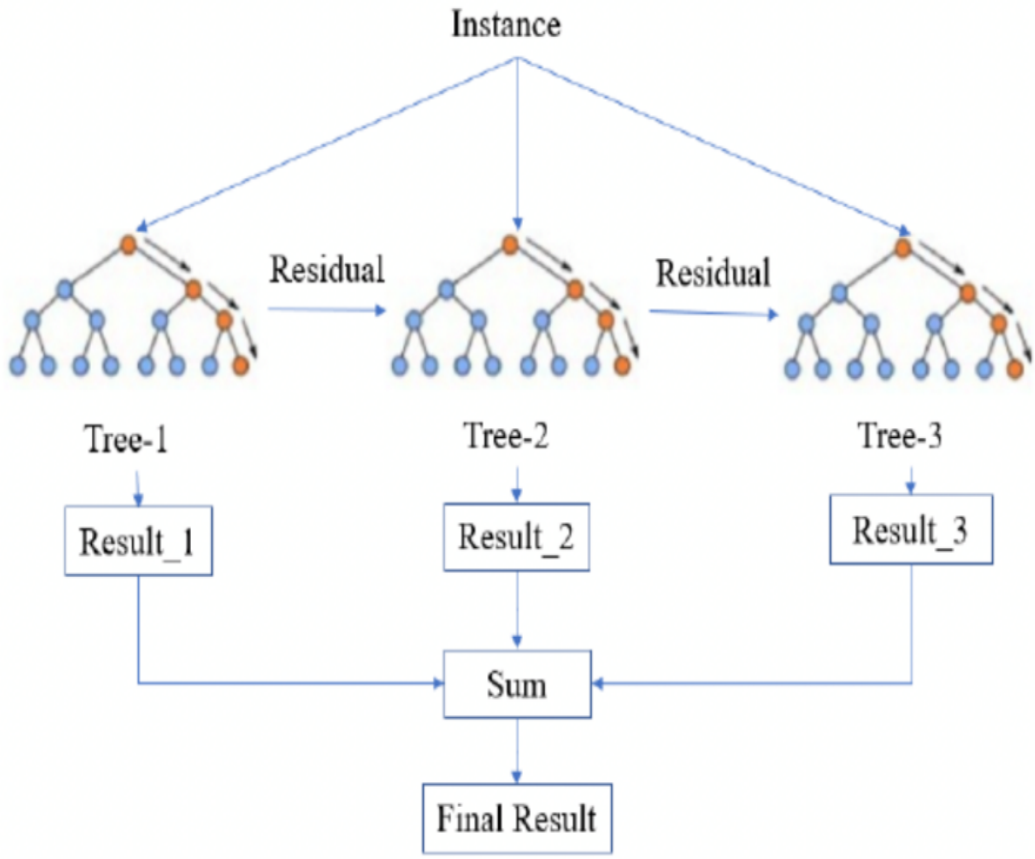
Gambar 2.6 *Stacking Ensemble Learning* (Linggar & Adi, 2022)

2.6 *Extreme Gradient Boosting* (XGBoost)

Extreme Gradient Boosting (XGBoost) adalah salah satu jenis *Gradient Tree Boosting* yang mampu mengatasi masalah dengan skala besar dan umum digunakan untuk menyelesaikan permasalahan klasifikasi dan regresi. XGBoost pertama kali diciptakan oleh Tianqi Chen pada tahun 2014. Sejak diperkenalkan pada 2014 lalu, XGBoost dapat menjuarai berbagai kompetisi *kaggle* dengan menjadi solusi yang efektif, kuat, dan efisien terutama pada permasalahan klasifikasi (Septiana Rizky et al., 2022).

XGBoost merupakan bagian dari metode *boosting* dalam machine learning yang terdiri dari beberapa pohon keputusan (*Decision Tree*) dimana untuk membangun pohon berikutnya akan bergantung pada pohon sebelumnya (Syukron et al., 2020). XGBoost merupakan peningkatan dari metode *Gradient Boosted Decision Tree* (GBDT) yang mampu menjalankan komputasi secara paralel, membangun *tree* secara aproksimatif, serta memproses data *sparse* dengan lebih efisien (Kurnia et al., 2023). Algoritma ini menambahkan beberapa teknik yang efisien untuk mengurangi permasalahan *overfitting*, *split finding*, dan mengatasi permasalahan *missing value* dalam fase pelatihan (*training*) (Chen & Guestrin, 2016). Dalam mengatasi permasalahan *overfitting,* XGBoost menggunakan fungsi objektif dan regularisasi (Jafar et al., 2020).

XGBoost memiliki peningkatan dalam segi implementasinya dengan menggunakan metode pembelajaran paralel, komputasi *out-of-core*, dan akses *cache-aware* untuk meminimalkan kompleksitas dalam waktu pembelajaran, sehingga algoritma ini cocok digunakan untuk dataset yang besar (Jafar et al., 2020). Konsep dasar algoritma *Extreme Gradient Boosting* (XGBoost) terlihat seperti yang disajikan pada Gambar 2.7 dibawah ini:



Gambar 2.7 Konsep Dasar XGBoost (Wang et al., 2021)

XGBoost merupakan algoritma berbasis *gradient boosting* yang menggunakan berbagai fungsi pelengkap untuk memperkirakan hasilnya (Shahani et al., 2021). Hasil prediksi didapat menggunakan 2.4 dibawah ini:

$$\bar{y} = y_i^0 + \eta \sum_{K=1}^{n} f_k(U_i) \tag{2.6}$$

Dimana hasil prediksi untuk data ke-i ditunjukkan dengan $\bar{y}$, dihasilkan melalui vektor parameter U_i. Proses prediksi melibatkan sejumlah *estimators* yang ditunjukkan dengan n, dimana setiap f_k merupakan struktur pohon keputusan independen yang bekerja dari K=1 hingga K=n. Hipotesis awal atau *primer* dilambangkan dengan y_i^0 yang merupakan rata-rata dari parameter asli pada data pelatihan (*data training*). Selanjutnya, η menunjukkan laju pembelajaran atau *learning rate*, yang mengatur seberapa besar kontribusi pohon keputusan terhadap model keseluruhan. Dengan

mengontrol *learning rate*, dapat membantu kinerja model secara bertahap dan mencegah terjadinya *overfitting*.

Pada tahap ke-*k* pada proses *boosting*, *estimators* ke-*k* dihubungkan dengan model dan hasil prediksi y_i^{-k} dihitung berdasarkan hasil estimasi $y_i^{-(k-1)}$ pada langkah berikutnya. *Estimators* komplementer ke-*k* yang ditunjukkan oleh f_k dirumuskan seperti Persamaan 2.7 dibawah ini:

$$y_i^{-k} = y_i^{-(k-1)} + \eta f_k \tag{2.7}$$

Dimana f_k merepresentasikan bobot daun yang didapat dari mengurangi fungsi objektif pohon ke-*k.* Fungsi Objektif dirumuskan menggunakan Persamaan 2.8 dibawah ini:

$$f_{obj} = \gamma Z + \sum_{a=1}^{Z} \left[g_a \omega_a + \frac{1}{2}(h_a + \lambda)\omega_a^2 \right] \tag{2.8}$$

Dimana *Z* merupakan jumlah simpul daun pada pohon keputusan. Parameter γ dan λ merupakan parameter regularisasi yang digunakan untuk mencegah *overfitting* dan mengatur kompleksitas model. γ menunjukkan parameter kompleksitas dan λ merupakan koefisien konstan. g_a dan h_a merupakan penjumlahan parameter untuk seluruh dataset yang berkaitan dengan gardien dari fungsi kerugian awal dan sebelumnya, secara berurutan. Dalam membentuk pohon ke-*k*, daun dibagi menjadi beberapa daun. Hal tersebut dilakukan menggunakan parameter *gain* yang dirumuskan pada Persamaan 2.9 dibawah ini:

$$G = \frac{1}{2}\left[\frac{O_L^2}{P_L + \lambda} + \frac{O_R^2}{P_R + \lambda} + \frac{(O_L + O_R)^2}{P_R + P_L + \lambda} \right] \tag{2.9}$$

Dimana *G* menunjukkan *parameter gain*, dan O_R dan P_R menunjukkan daun kanan. Sedangkan, O_L dan P_L menunjukkan daun kiri. γ dan λ merupakan parameter

regularisasi yang berperan dalam mengontrol kompleksitas model dan secara tidak langsung bergantung pada parameter *gain*. Ketika *parameter gain* mendekati nol, pembagian yang telah dilakukan dianggap baik dan dapat diterima menjadi kriteria pemisahan yang baik.

2.7 *Parameter Tuning*

Parameter tuning merupakan salah satu proses penting dalam pengembangan model *machine learning* yang bertujuan untuk menemukan parameter terbaik dalam kumpulan parameter (Adiwiguna et al., 2022). Proses ini digunakan untuk mengembangkan performa kinerja suatu model *machine learning* dengan pengoptimalan parameter-parameter tersebut (Yulianti et al., 2022). Beberapa contoh parameter yang terdapat pada algoritma XGBoost antara lain *n_estimators, max_depth, min_child_weight, learning_rate, gamma, sub_sample,* dan lain sebagainya. Setiap parameter tersebut memiliki peran dalam menentukan seberapa baik model dapat mempelajari data. Fungsi atau kegunaan dari beberapa parameter yang terdapat pada algoritma *Extreme Gradient Boosting* terlihat seperti Tabel 2.2 berikut ini:

Tabel 2.2 Parameter dan Fungsinya

Parameter	**Kegunaan**
n_estimators	Menentukan jumlah pohon keputusan yang akan dibangun dalam proses pelatihan model.
eta (learning_rate)	Mengatur langkah yang akan diambil oleh XGBoost dalam pembaharuan bobot selama pelatihan serta membantu mempersingkat langkah dalam setiap iterasi atau pengulangan.
max_depth	Menentukan kedalaman terbanyak atau maksimum dari setiap pohon keputusan yang dibangun.

Parameter	Kegunaan
min_child_weight	Menetapkan bobot minimum yang diperlukan dalam setiap *node* anak untuk melakukan pembagian
gamma	Mengontrol dan meninimumkan pengurangan kerugian yang diperlukan untuk melakukan pemisahan.
subsample	Mengatur proporsi data pelatihan yang digunakan untuk membangun setiap pohon. Menggunakan subset data dapat mengurangi varians dan mencegah overfitting, tetapi nilai yang terlalu kecil dapat menyebabkan model kehilangan informasi penting.
colsample_bylevel	Mengatur rasio fitur yang digunakan dalam membangun pohon dari *data training*

2.8 *Mean Squared Error* (MSE)

Mean Squared Error (MSE) adalah metrik yang melihat perbedaan kuadrat antara nilai yang diprediksi oleh model dan nilai sebenarnya yang diamati. metrik ini digunakan untuk mengevaluasi kinerja model pembelajaran mesin, khususnya regresi (Almaliki et al., 2023). MSE dapat dihitung menggunakan Persamaan 2.10 dibawah ini:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.10}$$

Dimana *n* menunjukkan total data, $\hat{y}_i$ merupakan nilai hasil prediksi, dan y_i merupakan nilai sebenarnya. Tahapan perhitungan MSE dimulai dengan mengurangi nilai prediksi ($\hat{y}_i$) dari nilai aktual (y_i) untuk setiap data. Selanjutnya, perbedaan nilai tersebut dikuadratkan untuk menghilangkan tanda negatif. Setelah itu, perbedaan nilai

untuk setiap data dijumlahkan dan kemudian dibagi dengan jumlah total data (n) untuk menghasilkan rata-rata kesalahan kuadrat (MSE).

Kinerja yang lebih baik oleh model ditunjukkan dengan nilai MSE yang lebih rendah karena prediksi yang dihasilkan oleh model lebih dekat dengan nilai aktual. Sebaliknya, nilai MSE yang lebih tinggi menunjukkan kesalahan prediksi yang lebih besar oleh model. Hal tersebut terjadi akibat MSE yang sensitif terhadap *outlier*, nilai kesalahan yang besar dapat mendominasi perhitungan dan memperbesar nilai MSE secara keseluruhan (Pipin et al., 2023).

2.9 *Root Mean Squared Error* (RMSE)

Root Mean Squared Error (RMSE) adalah indikator populer untuk menilai seberapa baik kinerja model pembelajaran mesin untuk menemukan akar dari MSE. RMSE adalah metrik lain yang dapat menunjukkan seberapa tidak akuratnya model prediksi (Ruswanti, 2020). RMSE dapat dihitung menggunakan Persamaan 2.11 dibawah ini:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.11}$$

Dimana jumlah data observasi atau data poin dilambangkan dengan n, nilai aktual dari data observasi ke-i dilambangkan dengan y_i, nilai prediksi untuk data observasi ke-i dilambangkan dengan $\hat{y}_i$. Tahapan dalam perhitungan RMSE dimulai dengan menghitung selisih antara nilai prediksi ($\hat{y}_i$) dari nilai aktual (y_i) untuk setiap observasi. Selisih ini kemudian dikuadratkan untuk menghilangkan tanda negatif. Setelah itu, hasil kuadrat dari semua selisih ini dijumlahkan dan dibagi dengan jumlah total observasi (n) untuk mendapatkan rata-rata kesalahan kuadrat (MSE). Langkah terakhir adalah mengambil akar kuadrat dari hasil rata-rata ini untuk mengembalikan kesalahan ke skala asli dari data.

RMSE yang besar menunjukkan bahwa model memiliki kesalahan prediksi yang signifikan dan menyimpang jauh dari nilai aktual, yang menunjukkan bahwa kinerja model tidak optimal. Sebaliknya, RMSE yang kecil mengindikasikan bahwa prediksi model mendekati nilai sebenarnya, sehingga model dianggap lebih akurat dan dapat diandalkan (Vermaysha & Nurmalitasari, 2023). Metrik ini sangat berguna karena

menyajikan kesalahan dalam satuan yang sama dengan target aslinya, memudahkan interpretasi kinerja model dalam konteks masalah yang dihadapi.

2.10 *Mean Absolute Error* (MAE)

Mean Absolute Error (MAE) mewakili rata-rata perbedaan absolut antara nilai yang diharapkan dan nilai aktual (Nurani et al., 2023). Akurasi prediktif model statistik dapat dievaluasi dengan bantuan MAE, yang merangkum jumlah rata-rata kesalahan yang dihasilkan oleh model. Perhitungan MAE dapat dilakukan dengan menggunakan Persamaan 2.12 berikut:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i] \tag{2.12}$$

Dimana n merupakan banyaknya data yang diuji, y_i adalah nilai aktual dari data observasi ke-i, *dan* $\hat{y}_i$ menunjukkan nilai prediksi untuk data observasi ke-i. Komponen $[y_i - \hat{y}_i]$ menampilkan nilai kesalahan dengan membandingkan nilai yang diantisipasi dengan nilai sebenarnya. Langkah-langkah dalam menghitung MAE adalah sebagai berikut:

1. Langkah pertama dalam perhitungan MAE adalah mengambil selisih antara nilai prediksi ($\hat{y}_i$) dari nilai aktual (y_i). Selisih tersebut menunjukkan seberapa besar perbedaan antara data yang diprediksi dengan data asli.
2. Langkah selanjutnya adalah menghitung nilai absolut yaitu mengabaikan nilai positif atau negatif sehingga seluruh nilai dianggap sebagai positif dan menjumlahkan seluruh nilai selisih absolut.
3. Setelah semua selisih absolut dijumlahkan, total ini kemudian dibagi dengan jumlah data observasi (n) untuk mendapatkan rata-rata kesalahan absolut. Hasil rata-rata inilah yang disebut sebagai MAE

Skor MAE yang lebih rendah menunjukkan bahwa model tersebut berkinerja lebih baik dalam hal membuat prediksi.

2.11 Penelitian Terdahulu

Untuk menyelesaikan penelitian ini, penulis merujuk pada sejumlah penelitian sebelumnya. Hal tersebut terlihat seperti pada Tabel 2.3 dibawah ini:

Tabel 2.3 Penelitian Terdahulu

No	Penulis	Judul	Tahun	Hasil
1	Kulbir S.	*Healthcare Fraudulence: Leveraging Advanced Artificial Intelligence Techniques for Detection*	2024	Penelitian ini menggunakan *logistic regression* untuk deteksi kecurangan atau penipuan pada asuransi kesehatan. Penelitian ini berhasil mencapai AUC sebesar 0,9471 pada data pelatihan dan 0,9518 pada data pengujian, serta skor F1 masing-masing 0,5689 dan 0,5509. AUC merupakan metrik yang mengukur kemampuan model dalam membedakan antara klaim yang valid dan klaim yang berpotensi penipuan, dengan nilai mendekati 1 menunjukkan model memiliki performa klasifikasi yang sangat baik.

No	Penulis	Judul	Tahun	Hasil
2	Kevin V.N., Binti S., dan Abdul R.	*Prediction of Inefficiency in Health Insurance Administration Institutions in Indonesia using Light Gradient Boosting Machine*	2023	Penelitian ini memanfaatkan algoritma LGBM untuk menprediksi tidak efisiennya klaim administrasi dan menghasilkan AUC sebesar 0,75. Hasil tersebut potensi algoritma dalam meningkatkan efisiensi proses klaim di institusi asuransi kesehatan di Indonesia.
3	Hanif N.R.	Deteksi Inefisiensi pada Klaim BPJS Kesehatan dengan menggunakan *Machine Learning*	2023	Penelitian ini mendeteksi inefisiensi pada klaim BPJS Kesehatan menggunakan *Random Forest* yang dikombinasikan dengan *Tomek Links* dengan skor F1 sebesar 19.53 sebagai hasil terbaik.

No	Penulis	Judul	Tahun	Hasil
4	Alan C.N., dan M.I. Irawan.	Komparasi Deteksi Kecurangan pada Data Klaim Asuransi Pelayanan Kesehatan Menggunakan Metode *Support Vector Machine* (SVM) dan *Extreme Gradient Boosting* (XGBoost)	2023	Penelitian ini membandingkan algoritma SVM dan XGBoost dalam mengenali kecurangan dalam informasi klaim asuransi bidang kesehatan. XGBoost terbukti unggul dengan nilai akurasi keseimbangan (*balance accuracy*) sebesar 0,9995 dan *recall* sebesar 0,9994. Hal tersebut menunjukkan efektivitasnya dalam mengidentifikasi klaim yang mencurigakan.
5	Eman N., dan Abdullah A.	*Fraud Detection in Healthcare Insurance Claims Using Machine Learning*	2023	*Random Forest* diidentifikasi sebagai algoritma terbaik dalam mendeteksi kecurangan asuransi kesehatan dengan akurasi 98.21%, presisi 98.08%, *recall* 100%, dan skor F1 99.03%.

No	Penulis	Judul	Tahun	Hasil
6	Saravanan P., Arun R.L., Abdul A.K., K.J. Sathick, dan Vaishnavi J.	*Detection of Health Insurance Fraud using Bayesian Optimized XGBoost*	2023	Penelitian ini menggunakan XGBoost untuk mendeteksi kecurangan pada klaim asuransi kesehatan dengan hasil akurasi 98%, AUC 0.994, precision 98%, recall 97%, dan skor F1 97.5%. Hasil ini menunjukkan model sangat efektif dalam membedakan klaim curang dan valid dengan keseimbangan optimal antara *precision* dan *recall*.
7	Laiqa R., Waqas H.B., Kashid N., dan Sana N.	*Prediction of Insurance Fraud Detection using Machine Learning Algorithms*	2022	Hasil penelitian ini ditunjukkan dengan *Decisition Tree* yang menjadi algoritma terbaik dengan tingkat akurasinya yang mencapai 78%.
8	Humasak T.A.S., Marojahan S., dan Heppy M.S.	Deteksi *Fraud* Pada Klaim Layanan Rumah Sakit Menggunakan Model *Neural Network*	2021	Penelitian ini mengimplementasikan *Neural Network* untuk mendeteksi kecurangan klaim rumah sakit, dengan akurasi dan presisi di atas 50%, serta *recall* sebesar 78%.

No	Penulis	Judul	Tahun	Hasil
9	Rohan Y.G., Satya S.S.M, dan Pallav K.B.	*A Comparative Study of Using Various Machine Learning and Deep Learning-Based Fraud Detection Models For Universal Health Coverage Schemes*	2021	*Neural Network* yang dikombinasikan dengan metode *undersampling* menunjukkan kinerja terbaik dengan skor F1 sebesar 0.95 dalam mendeteksi kecurangan pada skema *universal health coverage.*
10	Prof. Edison W. L. dan Said A.J.	*Establishing A Machine Learning Model For Fraud Detection In Medical Insurance Claims*	2021	*Naive Bayes* Classifier diidentifikasi sebagai model terbaik dalam mendeteksi kecurangan asuransi kesehatan, dengan *recall* sebesar 98%.

Penelitian terkait deteksi penipuan atau kecurangan di fasilitas kesehatan dilakukan oleh Kulbir (2024) menggunakan pendekatan algoritma *Logistic Regression*, *Decision Tree,* dan *Random Forest*. Penelitian ini menunjukkan bahwa *logistic regression* memberikan hasil terbaik dengan AUC pada data pelatihan sebesar 0.9471 dan AUC pada data pengujian sebesar 0.9518. *F1 Score* pada data pelatihan mencapai 0.5686, sedangkan pada data pengujian sebesar 0.5509.

Penelitian selanjutnya dilakukan oleh Kevin et al. (2023) berfokus pada pendeteksian inefisiensi dalam administrasi asuransi kesehatan di Indonesia dengan memanfaatkan algoritma LGBM. Penelitian ini menggunakan sembilan variabel

diantaranya kode diagnosa, kode prosedur, usia pasien, hubungan kepesertaan, jenis pelayanan, biaya, kode jenis pulang, kode kelas rawat, dan label. Label akan merepresentasikan status dari klaim administrasi asuransi kesehatan, apakah mengindikasi inefisiensi atau tidak. Hasil dari penelitian tersebut menunjukkan bahwa algoritma LGBM berhasil mendapatkan skor AUC sebesar 0,75.

Penelitian yang berfokus pada deteksi inefisiensi dalam klaim asuransi kesehatan juga telah dilakukan oleh Rofiq (2023). Dalam penelitian yang dilakukannya, peneliti dengan menggunakan memanfaatkan beberapa algoritma *machine learning learning* seperti *Gradient Boosting Classifier, Decision Tree, Random Forest,* SVM*, Naive Bayes, CatBoost,* dan *XGBoost*. Lebih jauh, penelitian ini menggunakan strategi penanganan data yang tidak seimbang, khususnya *oversampling* dan *undersampling*. Lokasi, jenis, usia, jenis kelamin, dan hubungan partisipan hanyalah beberapa dari tujuh belas variabel yang digunakan dalam penelitian ini, jenis pelayanan, diagnosa dari fasilitas kesehatan tingkat pertama, kondisi peserta saat pulang, kelas perawatan, diagnosa primer peserta, jumlah diagnosa sekunder peserta, CMG (*Casemix Main Groups*), tipe kelompok kasus (*Case Groups*), spesifikasi kelompok kasus, tingkat keparahan kelompok kasus, lama perawatan, dan label potensi inefisiensi klaim. Hasil dari penelitian tersebut menunjukkan bahwa algoritma *Random Forest* yang dikombinasikan dengan teknik *Tomek Links* adalah yang terbaik dengan skor F1 sebesar 19,53 dan terdapat lima variabel yang paling berpengaruh terhadap inefisiensi klaim diantaranya fasilitas kesehatan, usia peserta, diagnosa dari fasilitas kesehatan tingkat pertama, diagnosis medis utama seseorang dan jenis pusat medis yang mereka kunjungi.

Serupa dengan itu, Alan & Mohammad (2023) menggunakan algoritma SVM dan XGBoost untuk menyelidiki data terkait penipuan klaim asuransi kesehatan. Penelitian tersebut menerapkan berbagai proses pengolahan data termasuk pembersihan data (*data cleaning*) untuk menghilangkan *noise* dan *outlier*, rekayasa fitur untuk menghasilkan fitur dari serta mengolah fitur sesuai dengan kebutuhan model, serta pengodean *one-hot* pada fitur dengan tipe kategoris untuk mempermudah proses pelatihan model. Selain itu, metode *oversampling* digunakan untuk menangani masalah ketidakseimbangan pada data dan normalisasi data menggunakan *z-score* untuk memastikan data pada distribusi yang sesuai. Terakhir, pengkategorian data dilakukan untuk membedakan antara klaim yang dianggap sebagai penipuan dan yang tidak. Hasil akhir menunjukkan

bahwa metode XGBoost berhasil mencapai nilai *recall* sebesar 0,9994 dan akurasi seimbang 0,9995, sehingga mengungguli performa SVM dalam penelitian ini.

Nabrawi & Alanazi (2023) juga melakukan penelitian yang memanfaatkan berbagai metode pembelajaran mesin, termasuk Jaringan Saraf Tiruan (JST) atau *Artificial Neural Network* (ANN), Regresi Logistik, dan *Random Forest* untuk mengeksplorasi deteksi penipuan dalam klaim asuransi kesehatan. Data yang digunakan dalam studi ini diperoleh dari tiga fasilitas kesehatan di Arab Saudi selama periode dari Januari 2022 hingga Mei 2022. Variabel studi meliputi berikut ini: tanggal (*date*), gender, usia (*age*), kebijakan (*policy*), tipe polis (P_Type), penyedia layanan (*provider*), departmenet (*dept*), total klaim (*total claims*), penolakan (*rejection*), pendidikan (*education*), dan rasio (*ratio*). Hasil keluaran berupa label apakah data klaim terjadi *fraud* atau tidak. Hasil penelitian menunjukkan bahwa *Random Forest* mendapatkan tingkat akurasi sebesar 98.21%, presisi 98.08%, *recall* 100%, *F1 Score* sebesar 99.03%, tingkat spesifisitas (*specificity*) sebesar 80%, dan AUC 90% Sedangkan model *Logistic Regression* berhasil mendapatkan akurasi sebesar 80.36%, presisi sebesar 97.62% , *recall* sebesar 80.39%, *F1 Score* sebesar 88.17%, tingkat spesifisitas (*specificity*) sebesar 80%, dan AUC sebesar 80.20%. Sementara itu, *Artificial Neural Network* (ANN) berhasil mendapatkan akurasi sebesar 94.64%, presisi sebesar 98.00%, *recall* sebesar 96.08%, *F1 Score* sebesar 97.03%, tingkat spesifisitas (*specificity*) sebesar 80%, dan AUC sebesar 88.04%.

Penelitian selanjutnya yang berkaitan dengan deteksi *fraud* di asuransi kesehatan dilakukan oleh Saravanan et al. (2023) dengan mengimplementasikan *Bayesian Optimized XGBoost* (BOXGBoost). Penelitian ini menggunakan beberapa variabel seperti Bene ID (nomor identitas pasien), nomor *provider* fasilitas kesehatan, dokter yang menangani, dokter lain (*other physician*), kode diagnosis klaim masuk, kode prosedur klaim 3, kode prosedur klaim 4, kode diagnosis klaim 2, kode diagnosis klaim 4, tahun klaim, dan label *fraud*. Kesimpulan dari penelitian tersebut ini menunjukkan bahwa model yang diteliti berhasil mencapai akurasi sebesar 98%, AUC sebesar 0.994, *precision* sebesar 98%, *recall* sebesar 97% , dan *F1 Score* sebesar 97.5%.

Penelitian selanjutnya dilakukan oleh Laiqa et al. (2022) untuk mendeteksi *fraud* asuransi menggunakan algoritma *machine learning* diantaranya, SVM, *Random Forest, Decision Tree, Adaboost, Linear Regression, Naive Bayes,* MLP. Tujuan utama dari penelitian ini adalah untuk mengevaluasi efektifitas dari masing-masing algoritma

machine learning dalam mengidentifikasi dan mendeteksi kasus kecurangan yang sering terjadi di sektor asuransi. Dari penelitian yang telah dilakukan, diperoleh hasil yang menunjukkan bahwa algoritma *Decision Tree* adalah yang terbaik dengan tingkat akurasi mencapai 78%. Hal tersebut menunjukkan bahwa *Decision Tree* memiliki kemampuan yang lebih baik dibandingkan dengan algoritma lainnya dalam mendeteksi potensi kecurangan dalam klaim asuransi.

Penelitian terkait deteksi kecurangan asuransi kesehatan juga dilakukan oleh Humasak et al. (2021) dengan mengimplementasikan algoritma *Neural Netrwork*. Pada penelitian tersebut, para peneliti melacak dan mengembangkan model *Neural Network* untuk menganalisis data klaim asuransi kesehatan dengan tujuan utama untuk mendeteksi potensi kecurangan. Hasil penelitian tersebut menunjukkan bahwa algoritma *Neural Network* yang dibangun berhasil mendapatkan nilai akurasi dan *precision* diatas 50% yang mana menandakan efektivitas model dalam mendeteksi kecurangan. Selain itu, nilai *recall* yang mencapai 78% menunjukkan kemampuan model dalam mengidentifikasi kasus positif (*true positive*) atau kasus yang benar-benar merupakan kecurangan dengan baik.

Penelitian lain dilakukan oleh Rohan et al. (2021) yang bertujuan untuk mendeteksi *fraud* menggunakan *machine learning* dan *deep learning*. Penelitian ini menggunakan dataset layanan kesehatan dari Ayusman Bharat (PM-JAY). Penelitian ini menggunakan beberapa algoritma diantaranya, *Decision Tree, Random Forest, XGBoost, LightGBM,* GBM, dan *Neural Network* dan juga menggunakan beberapa teknik dalam menangani data yang tidak seimbang (*imbalance dataset*) yakni SMOTE, ADASYN, dan TGANS. Dari penelitian ini didapatkan bahwa algoritma *Neural Network* yang dikombinasikan dengan metode *undersampled* merupakan yang terbaik dari seluruh model yang diterapkan. Model ini berhasil mendapatkan *F1 Score* sebesar 0.95.

Penelitian lain dilakukan oleh Lubua & Juma (2021) bertujuan untuk mengembangkan model menggunakan teknik *machine learning* guna mendeteksi kecurangan atau *fraud* dalam klaim asuransi kesehatan. Penelitian ini menggunakan data dari NHIF Database. Penelitian ini menggunakan beberapa algoritma *machine learning* diantaranya, KNN, *Decision Tree*, *Logistic Regression*, dan *Naive Bayes Classifier*. Hasil penelitian ini menunjukkan bahwa algoritma KNN berhasil mendapatkan *recall* sebesar 81%, *Decision Tree* dengan *recall* 87%, *Logistic*

Regression dengan *recall* 82%, dan *Naive Bayes Classifier* dengan recall 98%. Oleh karena itu, *Naive Bayes Classifier* merupakan algoritma paling efektif dalam mendeteksi kecurangan (*fraud*) dalam klaim asuransi kesehatan berdasarkan data yang tersedia dari NHIF Database.

2.12 Perbedaan Penelitian

Terdapat beberapa penelitian terdahulu yang memiliki kolerasi dengan penelitian ini dalam mendeteksi inefisiensi dan *fraud* pada layanan asuransi kesehatan yang dilakukan dengan menggunakan metode dan pendekatan yang berbeda. Namun, tidak banyak penelitian yang secara mendalam membahas tentang deteksi *inflated bills*, yang merupakan salah satu bentuk kecurangan (*fraud*) di mana tagihan atas layanan medis yang sengaja dinaikkan untuk mendapatkan pembayaran yang lebih besar dari perusahan asuransi, dalam hal ini BPJS Kesehatan. Penelitian ini menunjukkan beberapa perbedaan dibandingkan beberapa penelitian terdahulu.

Perbedaan pertama terletak pada fokus penelitian. Seperti, penelitian yang dilakukan oleh Kulbir (2024) berfokus pada deteksi *fraud* pada asuransi kesehatan secara umum. Penelitian yang dilakukan oleh Kevin et al. (2023) dan Rofiq (2023) berfokus pada deteksi inefisiensi klaim BPJS Kesehatan secara umum. Penelitian-penelitian tersebut tidak spesifik pada deteksi *inflated bills*.

Perbedaan selanjutnya terdapat pada sumber data yang digunakan pada penelitian terdahulu. Misalnya, penelitian yang dilakukan oleh Nabrawi & Alanazi (2023) menggunakan data dari tiga fasilitas kesehatan di Arab Saudi. Selanjutnya, penelitian oleh Lubua & Juma (2021) yang menggunakan data dari NHIF *Database*. Data yang digunakan berasal dari BPJS Kesehatan di mana data tersebut memiliki karakteristik dan regulasi yang berbeda.

Perbedaan penelitian selanjutnya terletak pada metode yang digunakan. Penelitian yang dilakukan oleh Kulbir (2024) menggunakan metode *Logistic Regression*, *Decision Tree,* dan *Random Forest*. Sementara itu, penelitian yang dilakukan oleh Kevin et al. (2023) menggunakan metode *Light Gradient Boosting Machine* (LGBM). Penelitian yang dilakukan oleh Nabrawi & Alanazi (2023) menggunakan algoritma *machine learning* seperti *Random Forest, Logistic Regression,* dan *Artificial Neural Network*.

Penelitian ini akan menerapkan algoritma *Extreme Gradient Boosting* untuk mendeteksi potensi *inflated bills* pada data klaim layanan BPJS Kesehatan.

Perbedaan penting lainnya adalah pendekatan dalam mendeteksi *inflated bills*. Penelitian ini dilakuakn dengan memprediksi biaya verifikasi menggunakan *XGBoost Regression* (XGBRegressor) terlebih dahulu untuk selanjutnya dibandingkan hasilnya dengan biaya tagih oleh fasilitas kesehatan kepada pihak BPJS Kesehatan. Jika biaya tagih lebih besar dari prediksi biaya verifikasi, maka klaim tersebut terdeteksi berpotensi *inflated bills*. Jika tidak, maka klaim tersebut terdeteksi normal.

Dengan demikian, penelitian terdapat beberapa kebaruan dibandingkan penelitian-penelitian sebelumnya diantaranya fokus penelitian, sumber data, serta metode atau algoritma yang akan digunakan. Fokus yang spesifik pada deteksi potensi *inflated bills*, menggunakan data dari BPJS Kesehatan, dan teknik *preprocessing data* yang akan disesuaikaan menjadikan penelitian ini berbeda dari penelitian-penelitian sebelumnya.

BAB 3

ANALISIS DAN PERANCANGAN SISTEM

3.1 Data

Dataset yang digunakan untuk penelitian ini adalah dataset sampel pelayanan di Fasilitas Kesehatan Rujukan Tingkat Lanjut (FKRTL) dati tahun 2016 hingga 2022. Dataset dapat diakses dan diunduh setelah melalui pengajuan lewat website portal data JKN https://data.bpjs-kesehatan.go.id/. Dataset tersebut memiliki format stata (.dta) dengan 1.176.438 baris dan 55 kolom. Dataset sampel pelayanan di FKRTL untuk periode 2016-2022 terlihat seperti pada Gambar 3.1, dibawah ini:

	PSTV01	PSTV02	PSTV15	FKP02	FKL02	FKL03	FKL04	FKL05	FKL06	FKL07	...	FKL39	FKL40	FKL4
0	180205299	180205298	3.662519		100080122V000022	2022-01-03	2022-01-03	PAPUA	JAYAPURA	Pemerintah kab/kota	...		0	NON
1	314289762	314289762	14.453729		100080122V000028	2022-01-02	2022-01-02	PAPUA	JAYAPURA	Pemerintah kab/kota	...		0	NON
2	385562166	314289762	15.544407		100080122V000029	2022-01-02	2022-01-02	PAPUA	JAYAPURA	Pemerintah kab/kota	...		0	NON
3	69505864	69505864	2.836411		100080122V000183	2022-01-05	2022-01-05	PAPUA	JAYAPURA	Pemerintah kab/kota	...		0	NON
4	135231252	135231252	3.361672		100080122V000263	2022-01-07	2022-01-07	PAPUA	JAYAPURA	Pemerintah kab/kota	...		0	NON
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1176433	20647133	20647133	14.917422	345150922Y000546	99591122V000223	2022-11-24	2022-11-24	DKI JAKARTA	KEPULAUAN SERIBU	Pemerintah provinsi	...		0	NON
1176434	71048782	32664628	3.991986	99531122Y000575	99591122V000234	2022-11-25	2022-11-25	DKI JAKARTA	KEPULAUAN SERIBU	Pemerintah provinsi	...		0	NON
1176435	60464841	60464841	74.061844	345151122Y000751	99591122V000238	2022-11-25	2022-11-25	DKI JAKARTA	KEPULAUAN SERIBU	Pemerintah provinsi	...		0	NON
1176436	466097827	60655067	5.043636	99551222Y000104	99591222V000136	2022-12-14	2022-12-14	DKI JAKARTA	KEPULAUAN SERIBU	Pemerintah provinsi	...		0	NON
1176437	68356963	68356963	18.174042	345151222Y000683	99591222V000245	2022-12-29	2022-12-29	DKI JAKARTA	KEPULAUAN SERIBU	Pemerintah provinsi	...		0	NON

Gambar 3.1 Dataset Sampel FKRTL 2016-2022

Dataset mengenai histori klaim layanan pengobatan di Fasilitas Kesehatan Rujukan Tingkat Lanjut atau FKRTL tersebut memiiki banyak variabel yang memberikan informasi-informasi detail mengenai pelayanan yang diterima oleh pasien. Variabel-variabel tersebut meliputi nomor peserta, nomor kunjungan, nomor keluarga, bobot sampel, informasi mengenai rujukan asal dan kunjugan, tanggal kunjungan, tanggal pulang, lokasi FKRTL (seperti provinsi, kabupaten/kota, kepemilikan, jenis, dan tipe), tingkat pelayanan, jenis poli, segmen peserta, kelas iuran, premi peserta, status pulang, serta beberapa kode dan nama diagnosis masuk dan primer dengan menggunakan standar ICD 10.

Selain itu, terdapat informasi mengenai kode INACBGs yang mencakup kelompok kasus, tingkat keparahan, dan tarif regional. Selain itu, terdapat variabel lainnya yang terdiri atas jenis prosedur yang diterima, kode dan tarif untuk *sub-acute groups*, prosedur khusus (*special procedure*), *prosthesis,* investigasi khusus (*special investigation*), dan obat-obatan khusus. Terakhir, Dataset ini juga mencakup biaya tagih oleh fasilitas atau *provider* kesehatan dan biaya verifikasi oleh BPJS Kesehatan. Informasi lengkap mengenai variabel yang terdapat dalam dataset tersebut seperti yang diperlihatkan pada Tabel 3.1 berikut ini:

Tabel 3.1 Informasi Variabel Dataset

Variable	**Keterangan**
PSTV01	Nomor peserta
PSTV02	Nomor keluarga
PSTV15	Bobot
FKP02	No Asal Rujukan (ID Kunjungan FKTP)
FKL02	ID Kunjungan
FKL03	Tanggal datang kunjungan FKRTL
FKL04	Tanggal pulang kunjungan FKRTL
FKL05	Provinsi FKRTL
FKL06	Kabupaten/Kota FKRTL
FKL07	Kepemilikan FKRTL
FKL08	Jenis FKRTL
FKL09	Tipe FKRTL
FKL10	Tingkat Pelayanan FKRTL

Variable	Keterangan
FKL11	Jenis Poli FKRTL
FKL12	Segmen Peserta saat akses layanan FKRTL
FKL13	Kelas iuran premi peserta saat akses layanan FKRTL
FKL14	Status pulang dari FKRTL
FKL15	Kode dan nama diagnosis masuk ICD 10 (3 digit)
FKL15A	Kode diagnosis masuk ICD 10 (3 digit)
FKL16	Kode ICD 10 diagnosis masuk FKRTL (3-6digit)
FKL16A	Nama diagnosis masuk FKRTL (3-6digit)
FKL17	Kode dan nama diagnosis primer ICD 10 (3 digit)
FKL17A	Kode diagnosis primer ICD 10 (3 digit)
FKL18	Kode ICD 10 diagnosis primer FKRTL (3-6digit)
FKL18A	Nama diagnosis primer FKRTL (3-6digit)
FKL19	Kode INACBGs
FKL19A	Deskripsi kode INACBGs
FKL20	INACBGs - Kode Casemix utama (Digit ke-1)
FKL21	INACBGs - Tipe kelompok kasus atau jenis kasus(Digit ke-2)
FKL22	INACBGs - Spesifikasi kelompok kasus (Digit ke-3)
FKL23	INACBGs - Tingkat keparahan(Digit ke-4)
FKL25	Provinsi faskes perujuk
FKL26	Kabupaten/Kota faskes perujuk
FKL27	Kepemilikan faskes perujuk
FKL28	Jenis faskes perujuk
FKL29	Tipe faskes perujuk
FKL30	Jenis prosedur
FKL31	Tarif regional INACBGs
FKL32	Group Tarif INACBGs
FKL33	Kode *special sub acute* (SA)
FKL34	Tarif *special sub acute* (SA)
FKL35	Kode *special procedure* (SP)
FKL36	Deskripsi *special procedure* (SP)
FKL37	Tarif *special procedure* (SP)
FKL38	Kode *special prosthesis* (RR)

Variable	Keterangan
FKL39	Deskripsi *special prosthesis* (RR)
FKL40	Tarif *special prosthesis* (RR)
FKL41	Kode *special investigation*(SI)
FKL42	Deskripsi *special investigation*(SI)
FKL43	Tarif *special investigation*(SI)
FKL44	Kode *special drugs* (SD)
FKL45	Deskripsi *special drugs* (SD)
FKL46	Tarif *special drugs* (SD)
FKL47	Biaya Tagih - oleh fasilitas kesehatan (provider)
FKL48	Biaya Verifikasi - BPJS Kesehatan setelah dilakukan verifikasi

3.2 Variabel yang Digunakan

Dari keseluruhan variabel yang terdapat pada dataset sampel klaim layanan pengobatan di FKRTL, terdapat beberapa variabel yang akan digunakan dalam proses pembelajaran menggunakan algoritma *Extreme Gradient Boosting* (XGBoost). Dibawah ini adalah Tabel 3.2 yang mencakup detail variabel yang digunakan pada penelitian ini:

Tabel 3.2 Informasi Variabel yang Digunakan

Variabel	Keterangan
FKL09	Tipe FKRTL
FKL10	Tingkat pelayanan di FKRTL
FKL13	Kelas iuran
FKL17	Kode dan nama diagnosis primer icd
FKL18	Kode ICD-10 diagnosis primer FKRTL
FKL20	INACBGs – Kode *Casemix Main Groups*
FKL21	INACBGs – Tipe kelompok kasus atau *case groups*
FKL22	INACBGS – Spesifikasi kelompok kasus
FKL23	INACBGS – Tingkat keparahan kasus
FKL30	Jenis prosedur yang diberikan
FKL31	Tarif regional
FKL32	*Group* tarif regional

Variabel	Keterangan
FKL33	Kode *special sub acute* (SA)
FKL34	Tarif *special sub acute* (SA)
FKL35	Kode *special procedure* (SP)
FKL37	Tarif *special procedure* (SP)
FKL38	Kode *special prothesis* (RR)
FKL40	Tarif *special prothesis* (RR)
FKL41	Kode *special investigation* (SI)
FKL43	Tarif *special investigation* (SI)
FKL44	Kode *special drugs* (SD)
FKL46	Tarif *special drugs* (SD
FKL47	Biaya tagih
LOS	*Length of Stay* (LOS) atau lama masa rawat
FKL48	Biaya Verifikasi

Keputusan untuk menggunakan variabel-variabel tersebut diambil berdasarkan Peraturan Menteri Kesehatan Republik Indonesia No.3 Tahun 2023 tentang Standar Tarif Pelayanan Kesehatan Dalam Penyelenggaran Program Jaminan Kesehatan Nasional.

Variabel tipe dan tingkat pelayanan Fasilitas Kesehatan Rujukan Tingkat Lanjut atau FKRTL (FKL09 dan FKL10) diambil berdasarkan Pasal 28 Ayat 1 yang mengatur tarif untuk pelayanan rawat jalan dan rawat inap berdasarkan kelas dan jenis FKRTL. Selanjutnya, variabel kelas iuran (FKL13) diambil berdasarkan Pasal 28 Ayat 3 yang membagi tarif rawat inap ke dalam kelas 1, 2, dan 3 tergantung pada tingkat layanan yang diikuti peserta BPJS Kesehatan.

Variabel selanjutnya adalah Variabel INACBGs yang mencakup kode *casemix main groups* (FKL20), tipe kelompok kasus atau *case groups* (FKL21), spesifikasi kelompok kasus (FKL22), dan tingkat keparahan kasus (FKL23). Variabel-variabel tersebut dipilih berdasarkan Pasal 27 Ayat 1.

Beberapa variabel tambahan terkait *Special Casemix Main Groups* (CMG) seperti *special procedure* (FKL35 dan FKL37), *special drugs* (FKL44 dan FKL46), *special investigation* (FKL41 dan FKL43), *special prosthesis* (FKL38 dan FKL40), *subacute case*, dan *chronic case* mengacu pada Pasal 27 Ayat 2. Pasal ini menjelaskan jenis

layanan yang termasuk dalam *Special Casemix Main Groups* dan dapat memperoleh tambahan biaya atau *top up payment*. Selain itu, variabel tarif regional (FKL31) dan group tarif regional (FKL32) didasarkan pada Pasal 28 Ayat 2 yang membagi tarif ke dalam lima regional berbeda untuk menyesuaikan tarif berdasarkan wilayah.

Variabel lainnya seperti kode dan nama diagnosis primer ICD-10 (FKL17 dan FKL18), jenis prosedur yang diberikan (FKL30), dan *length of stay* atau lama masa rawat (LOS). Variabel-variabel tersebut diatur dalam lampiran Peraturan Menteri Kesehatan Republik Indonesia No.3 Tahun 2023. Penggunaan ICD-10 memungkinkan verifikasi diagnosis dan prosedur berdasarkan standar internasional, sedangkan *length of stay* (LOS) menunjukkan durasi perawatan yang dapat memengaruhi total biaya klaim

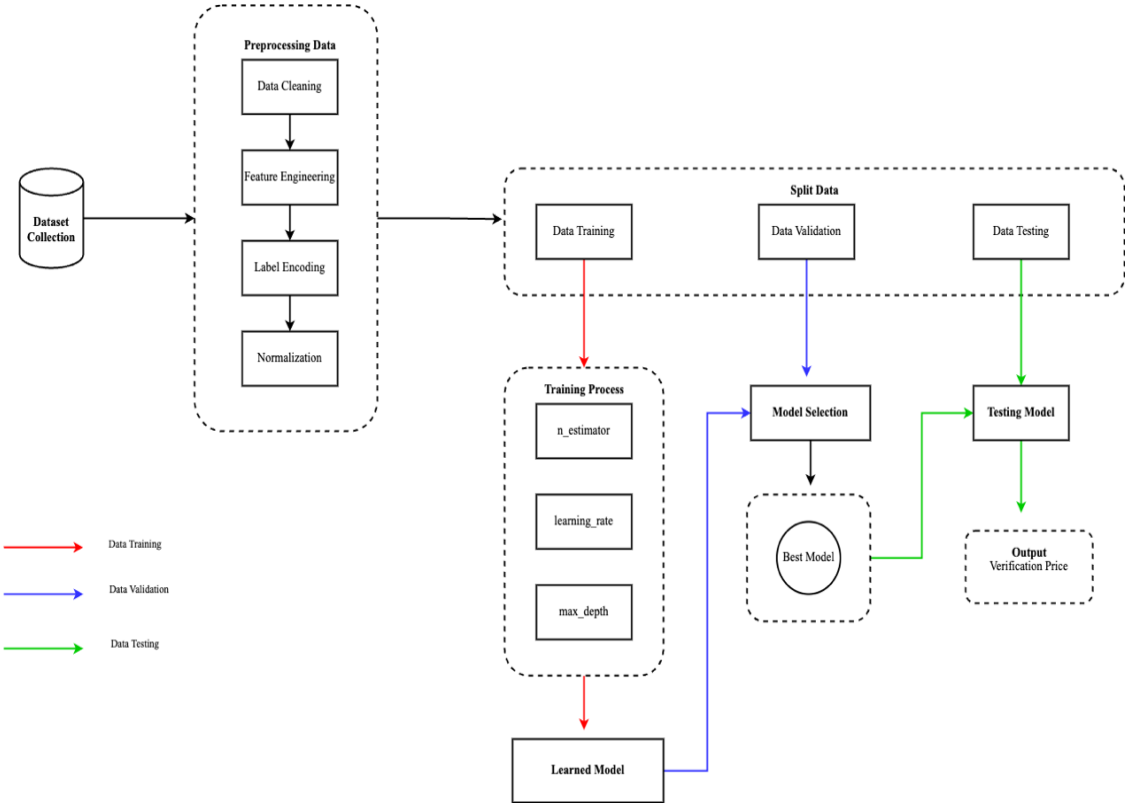
3.3 Arsitektur Umum

Pada penelitian ini dibentuk sebuah sistem yang mampu mendeteksi klaim yang berpotensi mengalami pembengkakakan tagihan (*inflated bills*) pada data klaim layanan BPJS Kesehatan. Klaim yang memiliki potensi *inflated bills* memiliki ciri-ciri di mana biaya yang ditagihkan oleh penyedia layanan kesehatan lebih besar dibandingkan dengan biaya yang diverifikasi oleh BPJS Kesehatan. Sistem tersebut dirancang menggunakan algoritma *Extreme Gradient Boosting* (XGBoost). Untuk membangun sistem tersebut, terdapat beberapa tahapan yang harus dilakukan.

Tahap pertama yang dilakukan dalam penelitian ini adalah mengumpulkan data yang diperlukan (*dataset collection*). Setelah data dikumpulkan, tahap selanjutnya adalah *preprocessing data* atau tahap pra-pengolahan data yang mencakup pembersihan data (*data cleaning*), rakayasa fitur (*feature engineering*), mengubah data kategorikal menjadi numerikal (*label encoding*), serta normalisasi data (*normalization*). Selanjutnya, data yang telah diolah masuk ke dalam tahap pembagian data (*split dataset*) di mana pada tahap ini data dibagi menjadi tiga kelompok yaitu *data training* (untuk melatih model)*, data validation* (untuk mengevaluasi kinerja model selama pelatihan), dan *data testing* (untuk menguji kinerja model pada data yang baru).

Tahap selanjutnya adalah pelatihan atau *training process.* Pada tahap ini, *data training* digunakan untuk melatih model dengan parameter tertentu yaitu *n_estimators*, *learning_rate*, dan *max_depth*. Hasil dari proses ini adalah model yang telah terlatih

(*learned model*). Setelah model terlatih, seleksi model (*model selection*) dilakuakn untuk memilih model terbaik (*best model*) menggunakan *data validation*. Setelah model terbaik ditemukan, tahap selanjutnya adalah pengujian model (*testing model*) yang dilakukan menggunakan *data testing* untuk memastikan model yang dipilih sudah berhasil memprediksi secara akurat pada data baru yang tidak terlihat selama proses pelatihan. Keseluruhan arsitektur umum dari proses ini dapat dilihat pada Gambar 3.2 berikut ini:



Gambar 3.2 Arsitektur Umum

3.3.1 *Dataset Collection*

Pada penelitian ini, penulis memanfaatkan dataset histori klaim pelayanan yang berasal dari Fasilitas Kesehatan Rujukan Tingkat Lanjut (FKRTL) mencakup periode tahun 2016 hingga 2022. Dataset tersebut berisi informasi mendetail mengenai berbagai klaim layanan kesehatan yang diajukan oleh FKRTL kepada BPJS Kesehatan selama kurun waktu tersebut. Setiap klaim dalam dataset memuat berbagai informasi penting, seperti

data terkait pasien, jenis layanan yang diberikan, diagnosa, prosedur yang dilakukan, hingga biaya yang ditagihkan.

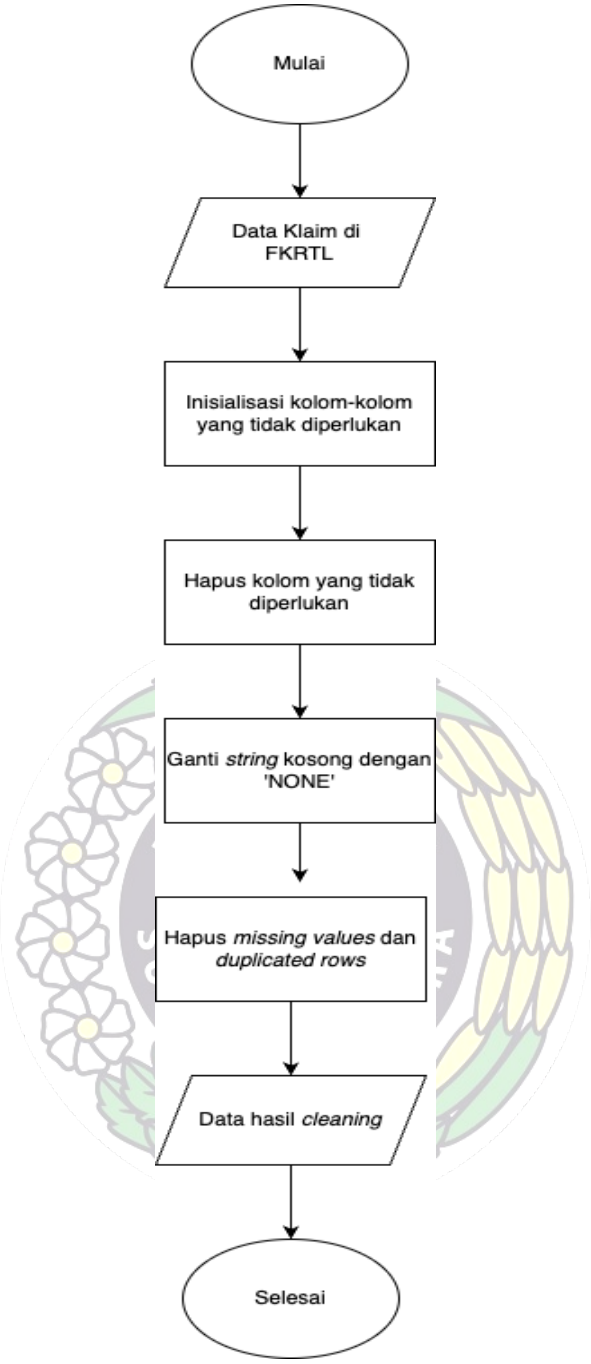
Penulis memperoleh data tersebut melalui pengajuan resmi pada *website* portal data Jaminan Kesehatan Nasional (JKN), yang menyediakan akses terbuka terhadap data-data terkait pelayanan kesehatan di Indonesia. Data yang digunakan terdiri atas 55 kolom yang memuat berbagai variabet atau atribut serta 1.176.438 baris yang mewakili setiap klaim layanan yang diajukan. Jumlah data yang cukup besar ini mencakup berbagai macam layanan kesehatan, jenis penyakit, dan kategori prosedur medis, sehingga memungkinkan penulis untuk melakukan analisis mendalam terkait pola klaim, efisiensi pelayanan, hingga potensi adanya kecurangan (fraud) dalam klaim yang diajukan.

3.3.2 *Preprocessing Dataset*

Tahap ini bertujuan untuk mempersiapkan data secara optimal sebelum melanjutkan ke proses berikutnya dan memastikan bahwa data yang digunakan memiliki kualitas yang lebih baik sehingga dapat mendukung pembentukan model dengan kinerja yang lebih efektif dan akurat. Tahap *preprocessing data* meliputi pembersihan data (*data cleaning*) untuk menghilangkan kesalahan atau anomali pada data, rekayasa fitur (*feature engineering*) yang bertujuan mengekstrak informasi relevan dari atribut yang ada atau menciptakan fitur baru yang dapat meningkatkan performa model, *label encoding* untuk mengonversi data dengan tipe kategori menjadi bentuk numerik yang dapat diproses oleh algoritma *machine learning,* serta normalisasi data (*data normalization*) yang berfungsi menyeimbangkan distribusi nilai-nilai atribut sehingga model yang dibangun dapat bekerja dengan lebih efisien dan konsisten dalam berbagai kondisi.

3.3.2.1 *Data Cleaning*

Proses pembersihan dataset sampel histori klaim pelayanan di FKRTL merupakan salah satu tahap krusial dalam penelitian ini. *Flowchart* tersebut dapat ditunjukkan pada Gambar 3.3 dibawah ini:



Gambar 3.3 *Flowchart Data Cleaning*

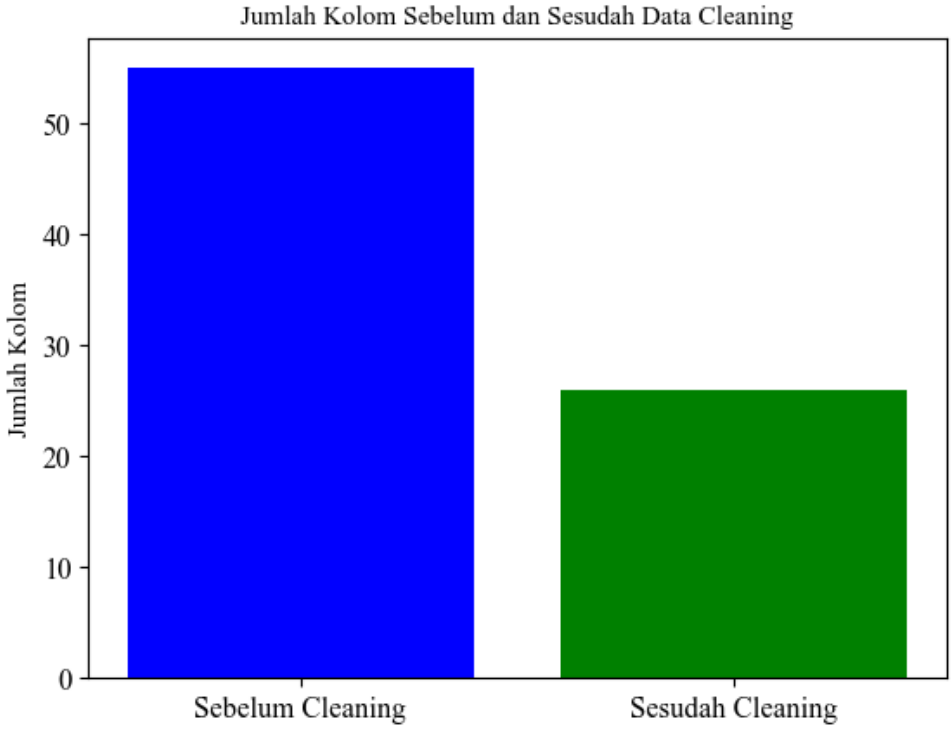
Pada tahap ini, penulis memulai proses pembersihan data dengan menginisialisasi dataset dan mengidentifikasi kolom-kolom yang tidak relevan untuk analisis lebih lanjut. Setelah menentukan dan menghapus kolom-kolom tersebut, penulis melanjutkan dengan mengganti nilai *string* yang kosong dengan istilah *'NONE'* untuk memastikan konsistensi dalam dataset. Selanjutnya, penulis menghapus semua baris yang mengandung nilai hilang (*missing values*) untuk menjaga integritas data, serta mendeteksi dan menghapus duplikat yang dapat mengganggu hasil analisis. Proses

pembersihan data ini sangat penting untuk memastikan bahwa analisis yang dilakukan nantinya berbasis pada data yang bersih dan berkualitas tinggi. *Pseudocode* yang menggambarkan langkah-langkah dalam proses *data cleaning* ini dapat di bawah ini

```
Function clean_data:
    Initialize columns to remove
    Remove unused columns
    Replace empty strings with 'NONE'
    Remove rows with missing values
    Remove duplicated rows
    Reset index
    Return cleaned data
```

Sebelum proses data cleaning dilakukan, dataset awal terdiri dari 55 kolom yang mencakup berbagai informasi penting terkait klaim pelayanan kesehatan. Kolom-kolom ini meliputi data demografis peserta, rincian layanan yang diberikan, serta informasi keuangan yang relevan. Namun, setelah melalui tahap *data cleaning*, penulis melakukan evaluasi menyeluruh terhadap setiap kolom untuk menentukan relevansinya dengan analisis yang akan dilakukan.

Proses ini sangat penting mengingat adanya Peraturan Menteri Kesehatan yang mengatur standar tarif pelayanan kesehatan. Oleh karena itu, kolom-kolom yang dianggap tidak diperlukan atau tidak relevan dengan tujuan analisis dihapus untuk menyederhanakan dataset. Dengan demikian, jumlah kolom yang tersisa kini berkurang menjadi hanya 26 kolom. Visualisasi yang menggambarkan perubahan jumlah kolom dalam dataset ini dapat dilihat pada Gambar 3.4 di bawah ini. Gambar tersebut menunjukkan perbandingan antara jumlah kolom sebelum dan sesudah proses *data cleaning.*



Gambar 3.4 Perbandingkan Kolom Sebelum dan Sesudah *Data Cleaning*

Setelah menghapus kolom-kolom yang tidak relevan tersebut, langkah selanjutnya adalah mengganti nilai yang kosong (' ') menjadi '*NONE*' untuk memudahkan dalam analisis lebih lanjut. Selain itu, penulis juga menghapus 2 baris dengan nilai yang *missing* serta 452.960 nilai yang duplikat. Proses ini dilakukan dengan tujuan agar kualitas data menjadi lebih baik. Informasi lengkap mengenai perubahan sebelum dan sesudah proses pembersihan data dapat dilihat pada Tabel 3.3 dibawah ini:

Tabel 3.3 Perbandingan Sebelum dan Sesudah *Data Cleaning*

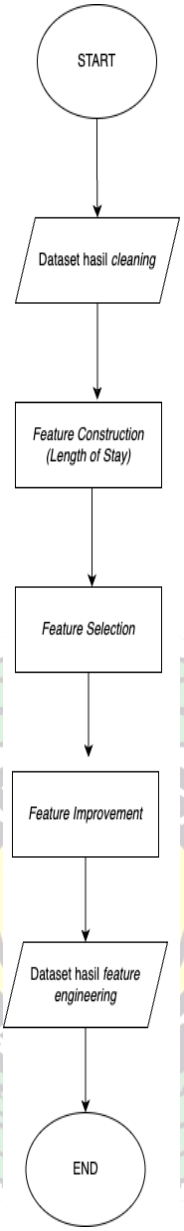
	Sebelum *Data Cleaning*	**Setelah *Data Cleaning***
Jumlah *missing values*	2	0
Jumlah *duplicated values*	452.960	0

Tabel 3.3 di atas menunjukkan kondisi data sebelum dan sesudah proses pembersihan atau *data cleaning*. Dari tabel tersebut, dapat dilihat bahwa setelah proses perbaikan data melalui *data cleaning* di mana tidak ada lagi data yang hilang maupun terduplikasi. Hal ini sangat kontras dengan kondisi awal di mana terdapat 2 data yang hilang dan jumlah yang signifikan yaitu 452.960 untuk data duplikat. Pembersihan data

ini tidak hanya meningkatkan kualitas dataset, tetapi juga memastikan bahwa analisis yang dilakukan akan memberikan hasil yang lebih *valid*, representatif, serta menurunkan kemungkinan *overfit* maupun *underfit*.

3.3.2.2 *Feature Engineering*

Langkah selanjutnya dalam *preprocessing data* merupakan *feature engineering* atau rekayasa fitur. Pada tahap ini, penulis melakukan beberapa langkah diantaranya pembuatan fitur baru dari fitur-fitur yang sudah ada (*feature construction*), pemilihan fitur yang relevan (*feature selection*), serta memperbaiki kualitas fitur (*feature improvement*). Gambar 3.5 dibawah ini menampilkan *flowchart* dari proses rekayasa fitur atau *feature engineering*:



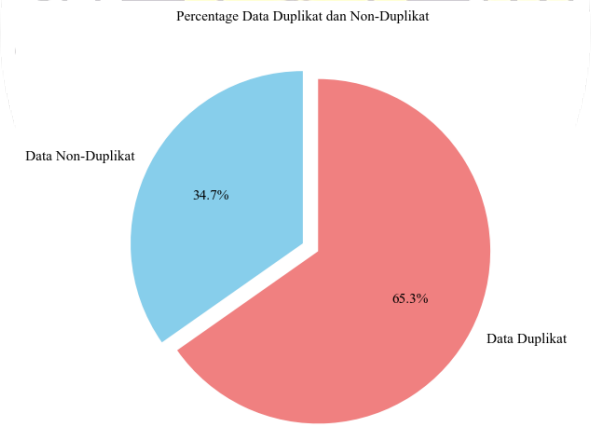
Gambar 3.5 *Flowchart Feature Engineering*

Setelah selesai membersihkan data pada tahap sebelumnya, langkah pertama dalam proses *feature engineering* atau rekayasa fitur adalah *feature construction.* Pada tahap ini, penulis membuat sebuah fitur baru yaitu "*Length of Stay" (*LOS). Fitur "*Length of Stay*" dibuat dengan memanfaatkan data tanggal kunjungan pasien (FKL03) dan tanggal pulang kunjungan (FKL04), sehingga dapat dihitung lama waktu pasien menginap dirumah sakit.

Langkah pertama dalam membentuk fitur *Length of Stay* adalah mengonversi kolom 'FKL03' yang berisi tanggal kunjungan pasien dan kolom 'FKL04' yang berisi tanggal

pulang kunjungan pasien menjadi tipe data *datetime*. Hal tersebut bertujuan untuk memastikan data tersebut dikenali sebagai tanggal oleh bahasa pemrograman *python* sehingga perhitungan selisih hari dapat terbentuk. Selanjutnya, kolom 'LOS' dibuat dengan menghitung selisih antara tanggal pulang kunjungan dan tanggal datang kunjungan. Terakhir, kolom 'LOS' diubah menjadi *integer* untuk memudahkan proses pembelajaran mesin.

Langkah selanjutnya setelah *feature construction* adalah *feature selection*. Pada tahap ini, dilakukan pemilihan ulang terhadap fitur-fitur yang akan digunakan dalam pembelajaran model. Penelitian memutuskan untuk menghapus tanggal kunjungan pasien (FKL03), dan tanggal pulang kunjungan (FKL04) karena fitur tersebut sudah diwakilkan oleh fitur *Length of Stay*. Langkah terakhir dalam proses *feature engineering* adalah *feature improvement*. Pada tahap ini, dilakukan pembersihan data kembali untuk memastikan kualitas dari dataset. Langkah-langkah yang diambil seperti penghapusan data duplikat. Gambar 3.6 dibawah ini merupakan visualisasi yang menggambarkan distribusi dan persentase dari data yang duplikat.



Gambar 3.6 Persentase Data Duplikat dan Non-Duplikat

Berdasarkan Gambar 3.6 diatas terlihat bahwa 65,3% dari data merupakan duplikat, sementara 34,7% lainnya merupakan data yang tidak duplikat. Dengan proporsi yang sangat tinggi dari data duplikat ini maka penting untuk menghapus entri-entri tersebut. Data duplikat dapat menyebabkan bias dalam analisis, mengurangi performa model yang dikembangkan, serta memperlambat proses komputasi. *Pseudocode* yang menggambarkan proses feature engineering dapat dilihat di bawah ini:

```
Function feature_engineering:
    Read cleaned data
    Construct feature Length of Stay (LOS)
    Change Length of Stay (LOS) data type to integer
    Feature selection (remove unused columns)
    Feature improvement (clean data)
    Return feature engineered data
```

Perubahan yang terjadi selama fase *feature engineering* dapat dilihat secara rinci melalui perbandingan yang ditunjukkan pada Tabel 3.4 dibawah ini:

Tabel 3.4 Perbandingan Sebelum dan Sesudah *Feature Engineering*

	Sebelum *Feature Engineering*	**Setelah *Feature Engineering***
Jumlah Data	723.476	251.341
Jumlah Kolom	26	25

Tabel 3.4 di atas memperlihatkan perubahan signifikan dalam jumlah kolom dan data (baris) setelah tahap *feature engineering* dilakukan. Sebelum proses ini, dataset terdiri dari 723.476 baris dan 26 kolom. Namun setelah melalui tahap *feature engineering*, jumlah data berkurang secara drastis menjadi 251.341 baris dengan jumlah kolom yang tersisa sebanyak 25. Pengurangan ini menunjukkan bahwa proses *feature engineering* berhasil menyaring data yang tidak relevan dan mengoptimalkan fitur yang ada sehingga menghasilkan dataset yang lebih efisien dan siap untuk analisis lebih lanjut.

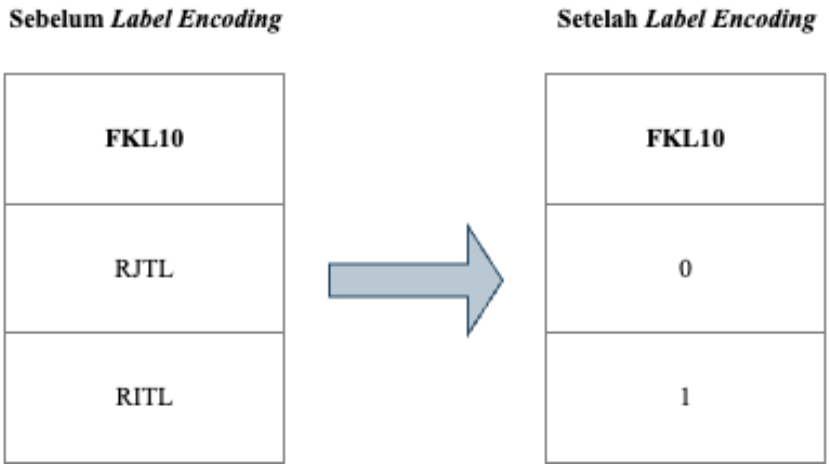
3.3.2.3 *Label Encoding*

Salah satu proses penting dan *preprocessing data* adalah *label encoding*, terutama dalam mengatasi data dengan tipe kategorikal. Data kategorik tersebut tidak dapat langsung digunakan oleh algoritma *Extreme Gradient Boosting* (XGBoost) karena algoritma tersebut hanya dapat bekerja dengan *input* numerik. Dengan menggunakan teknik *label encoding*, setiap kategorik unik dalam data akan dikonversi menjadi

numerik. Dalam penelitian ini, penulis menggunakan kelas *LabelEncoder* yang disediakan oleh modul *sklearn.preprocessing*. *LabelEncoder* bekerja dengan menganti setiap kategori unik pada data menjadi nilai *integer* yang terurut. Berikut ini merupakan *pseudocode* dari *label encoding*.

```
Function label_encoding:
        initialize LabelEncoder
        For each column in dataframe with object and category data type:
                Apply LabelEncoder to transform the column
        Return modified dataframe
```

Pseudocode di atas menjelaskan proses *label encoding* yang digunakan dalam menangani data dengan tipe kategorik dalam sebuah *dataframe*. Proses ini dimulai dengan menginisiasi objek *LabelEncoder* dari modul *sklearn.preprocessing* yang akan digunakan untuk konversi data dari kategorik menjadi numerik. Setelah itu, setiap kolom pada *dataframe* yang memiliki tipe data *object* dan *category* diidentifikasi dan dipilih untuk diterapkan *label encoding*. Pada langkah selanjutnya, *LabelEncoder* digunakan untuk mengganti nilai kategorik menjadi numerik. Setelah kolom kategorik dikonversi menjadi numerik, *dataframe* yang telah dimodifikasi tersebut kemudian dikembalikan, dengan seluruh datanya menjadi numerik dan siap untuk analisis lebih lanjut. Sebagai ilustrasi, Gambar 3.7 di bawah ini menunjukkan proses label encoding pada kolom FKL10 atau Tingkat Pelayanan di FKRTL:



Gambar 3.7 Penerapan *Label Encoding* Pada Kolom FKL10

Pada Gambar 3.7, terlihat bahwa kolom FKL10 yang sebelumnya memiliki data dengan tipe kategoris dengan 2 kategoris, yaitu Rawat Jalan Tingkat Lanjut (RJTL), dan Rawat Inap Tingkat Lanjut (RITL). Setelah dilakukan *label encoding*, RJTL berubah menjadi 0 dan RITL berubah menjadi 1.

3.3.2.4 *Normalization*

Dataset yang digunakan dalam ini terdiri dari berberapa variabel yang memiliki rentang nilai yang berbeda-beda. Perbedaan rentang ini dapat menyebabkan model *machine learning* menjadi kurang optimal akibat variabel dengan nilai yang lebih besar mendominasi variabel dengan rentang nilai yang lebih kecil. Untuk mengatasi masalah ini, dilakukan normalisasi data yang merupakan proses transformasi data agar berda pada skala yang sama.

Dalam penelitian ini, *Z-Score* atau normalisasi berbasis standar deviasi yang digunakan untuk mengubah semua nilai dataset berdasarkan distribusi standar. Nilai hasil *z-score* memiliki rata-rata sebesar 0 dan standar deviasi sebesar 1. Pada tahapan ini, nilai selisih antara masing-masing nilai dan nilai rata-rata dihitung lalu dibagi dengan nilai standar deviasi.

StandardScaler adalah alat atau *scaler* yang digunakan dalam penelitian ini. *Z-score* sendiri diimplementasikan menggunaka *StandardScaler* yang terdapat pada pustaka *scikit-learn. Pseudocode* dibawah menunjukkan tahapan proses normalisasi data menggunakan *StandardScaler*.

```
Function normalization:
    Define feature and target variable
    Create scalers for the features and target
    Scale the feature variables
    Scale the target variable
    Save the scalers for future used
    Convert the scaled data back to dataframe
    Combine the scaled feature and target into a single dataframe
    Return the scaled data
```

Pseudocode diatas menunjukkan *pseudocode* dari proses normalisasi data menggunakan *StandardScaler*. Proses ini dimulai dengan mendefinisikan sebuah fungsi dengan nama *normalization*. Pada tahap awal, dilakukan pemilihan variabel untuk fitur dan target. Setelah itu, dibuat *scaler* untuk fitur dan target. Langkah selanjutnya adalah melakukan proses normalisasi terhadap variabel-variabel fitur dan juga variabel target menggunakan masing-masing *scaler* yang telah dibangun. Setelah proses tersebut selesai, *scaler* yang diguankan akan disimpan untuk keperluan selanjutnya untuk menjaga konsistensi dari proses normalisasi data tersebut. Kemudian, seluruh variabel fitur dan target yang telah dinormalisasi digabungkan kembali menjadi sebuah *dataframe*. Adapun Tabel 3.5 dibawah ini menunjukkan contoh variabel-variabel fitur yang memiliki rentang beragam

Tabel 3.5 Contoh Variabel Fitur

FKL22	**FKL30**
13	42188
14	38739
17	38739
17	38739
27	43016

Tabel 3.5 diatas menunjukkan dua variabel fitur yang memiliki rentang yang sangat beragam, diantaranya FKL22 atau INACBGs - Spesifikasi kelompok kasus (Digit ke-3) dan FKL 30 atau Jenis prosedur. Langkah-langkah untuk melakukan normalisasi data pada kedua variabel tersebut dilakukan menggunakan metode *z-score normalization* dengan bantuan *StandardScaler()* yang terdapat di *Python*. Metode tersebut dilakukan menggunakan Persamaan 2.4 seperti yang tertulis dibawah ini:

$$x' = \frac{x_i - \bar{x}}{\sigma}$$

Dimana x' menunjukkan hasil normalisasi, x_i menunjukkan nilai yang dihasilkan dari proses normalisasi, $\bar{x}$ menunjukkan rata-rata dari atribut, dan σ menunjukkan standar deviasi dari atribut. Langkah awal dalam normalisasi menggunakan *z-score*

adalah menghitung rata-rata dari setiap kolom. Perhitungan rata-rata dapat dilakuakan dengan menjumlahkan seluruh nilai di kolom tersebut, kemudian dibagi dengan jumlah observasi (n). Untuk kolom FKL22, rata-rata ($\bar{x}$) dihitung menggunakan Persamaan 2.2. Hasil perhitungan nilai rata-rata ditunjukkan sebagai berikut:

$$\bar{x}_{FKL22} = \frac{13 + 14 + 17 + 17 + 27}{5} = \frac{88}{5} = 17{,}6$$

Sedangkan rata-rata untuk kolom FKL30 dihitung sebagai berikut:

$$\bar{x}_{FKL30} = \frac{42188 + 38739 + 38739 + 38739 + 43016}{5} = \frac{5201421}{5} = 40284{,}2$$

Setelah menemukan rata-rata dari kedua kolom, langkah selanjutnya adalah menghitung standar deviasi untuk masing-masing kolom. Rumus perhitungan nilai standar deviasi dilakukan menggunakan Persamaan 2.3 seperti yang ditunjukkan berikut ini:

$$\sigma = \sqrt{\frac{\sum(x_i - \bar{x})^2}{N}}$$

Untuk kolom FKL22, standar deviasinya dihitung dengan cara berikut:

$$\sigma_{FKL22} = \sqrt{\frac{(13-17{,}6)^2+(14-17{,}6)^2+(17-17{,}6)^2+(17-17{,}6)^2+(27-17{,}6)^2}{5}}$$

Setelah dikalkulasi:

$$\sigma_{FKL22} = \sqrt{\frac{21.16+12.96+0.36+0.36+87.36}{5}} = \sqrt{\frac{122.2}{5}} = \sqrt{24.44} = 4{,}963$$

Sedangkan untuk FKL30, standar deviasinya dihitung sebagai berikut:

$$\sigma_{FKL30} = \sqrt{\frac{(42188-40284.2)^2+(38739-40284.2)^2+(38739-40284.2)^2+(38739-40284.2)^2+(43016-40284.2)^2}{5}}$$

Hasil akhirnya adalah:

$$\sigma_{\text{FKL30}} = 1910{,}5$$

Setelah mendapatkan rata-rata dan standar deviasi, langkah terakhir adalah melakukan normalisasi setiap nilai menggunakan rumus z-score. Sebagai contoh, untuk nilai $x_i = 13$ pada Kolom FKL22 dinormalisasi dihitung sebagai berikut:

$$x' = \frac{13 - 17{,}6}{4{,}963} = \frac{-4.6}{4{,}963} = -0{,}93$$

Sedangkan untuk $x_i = 42188$ pada kolom FKL30 dinormalisasi sebagai berikut:

$$x' = \frac{42188 - 40284{,}2}{1910{,}5} = \frac{1903.8}{1910{,}5} = 0{,}99$$

Proses tersebut diulang untuk setiap nilainya dalam kolom FKL22 dan FKL30 sehingga seluruh nilai pada kedua kolom tersebut telah berhasil dinormalisasikan. Setelah semua nilai dinormalisasi, data yang telah diolah dapat digunakan untuk tahap selanjutnya. Hasil akhir contoh tersebut terlihat seperti Gambar 3.12 dibawah ini:

Sebelum Normalisasi

FKL22	FKL30
13	42188
14	38739
17	38739
17	38739
27	43016

Setelah Normalisasi

FKL22	FKL30
-0,93	1,042
-0,73	-0,84
-0,12	-0,84
-0,12	-0,84
1,9	1,5

Gambar 3.8 Data Setelah Normalisasi

Gambar 3.8 di atas menunjukkan perbedaan antara data sebelum dan setelah dinormalisasi. Sebelum dinormalisasi, rentang data bervariasi secara signifikan, dengan nilai yang tersebar luas di seluruh skala. Namun setelah diterapkan normalisasi z-score, rentang data menjadi lebih terstandarisasi dengan rata-rata bernilai nol (0) dan standar deviasi dengan nilai 1. Kini seluruh nilai dalam data memiliki distribusi yang lebih seragam dan skala yang lebih konsisten. Hal tersebut mempermudah perbandingan antar data serta mencegah penurunan kualitas model yang dihasilkan seperti bias, *overfit*, maupun *underfit*.

3.3.3 *Splitting Dataset*

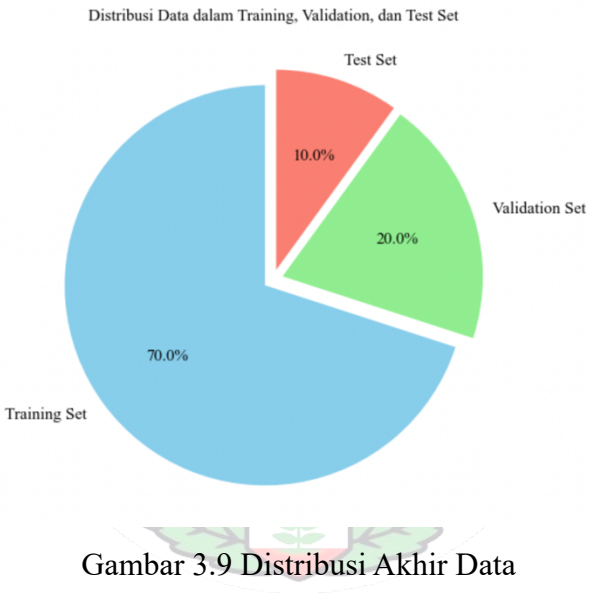
Setelah menyelesaikan tahap *preprocessing data*, penulis lebih dulu membagi data menjadi tiga subset yaitu *data training*, *data validation* dan *data testing*. Proses ini dimulai dengan pembagian awal antara *data training* dan *data testing* dengan proporsi perbandingan 90:10, dimana 90% dari total dataset dianggap sebagai *data training* lalu 10% sisanya dianggap sebagai *data testing*. *Data testing* ini disimpan secara terpisah dan tidak dilibatkan dalam proses pelatihan model (*training model*) untuk nantinya digunakan dalam mengukur performa dan kinerja model. Kedua *subset* tersebut selanjutnya disimpan dalam format *Comma Separated Values* (.CSV) untuk memudahkan akses saat proses *deployment* nanti.

Setelah memisahkan *data testing, data training* yang telah terbentuk sebelumnya kembali dipisahkan menjadi dua subset yaitu *data training* dan *data validation*. Pada tahap ini, *data validation* dialokasikan sebesar 2/9 dari *data training* yang telah dibagi sebelumnya. Pembagian ini bertujuan untuk menggunakan *data validation* sebagai upaya evaluasi kinerja model selama proses pelatihan (*training*). Kini data telah berhasil dibagi menjadi 3 subset yairu *data training, data testing,* dan *data validation*. Rincian lengkap dari pembagian dataset ini ditampilkan pada Tabel 3.6 di bawah ini.

Tabel 3.6 Pembagian Data

Data Training		***Data Testing***
Data Training	***Data Validation***	
70%	20%	10%

Data training terdiri dari 70% dari dataset awal. Sementara itu, *data testing* terdiri atas 10% dari dataset awal. Selain itu, *data validation* diambil sebesar 2/9 dari total *data training* atau 20% dari dataset awal. *Data validation* merupakan bagian dari *data training* yang telah ditentukan jumlahnya. Berikut Gambar 3.9 yang menunjukkan visualisasi pembagian data dalam konteks keseluruhan dataset



Gambar 3.9 Distribusi Akhir Data

Proses pembagian data tersebut dilakukan dengan memanfaatkan salah satu *library scikit-learn* yaitu *train_test_split* yang terdapat dalam modul *sklearn.model_selection*. Berikut ini merupakan *pseudocode* dari proses pembagian data atau *splitting data*:

```
Define features set as `X`
Define target variable and set as `y`
Split `X` and `y` into train and test sets using 90:10 split amd random seed 42
Assign the result of training features to `X_train` and training labels to `y_train`
Assign the result of testing features to `X_test` and testing labels to `y_test`
Combine `X_train` and `y_train` into a dataset called `train_data`
Combine `X_test` and `y_test` into a dataset called `test_data`
Save the `train_data` and `test_data` for future use
Further split `X_train` and `y_train` into training and validation sets using 2:9 split
and random seed 42
Assign the result of training features to `X_train` and training labels to `y_train`
Assign the result of validation features to `X_val` and validation labels to `y_val`
```

Pada *pseudocode* diatas terlihat bahwa langkah pertama dalam proses pembagian data (*data splitting*) adalah mendefinisikan variabel target sebagai `y` dan fitur-fitur sebagai `X`. Setelah itu, data dibagi menjadi *training set* dan *validation set* dengan perbandingan 90:10 menggunakan fungsi *train_test_split*. Hasil pembagian tersebut disimpan sebagai *`X_train'* dan *`y_train`* untuk *data training* serta *`X_test`* dan *`y_test`* untuk *data testing.* Langkah selanjutnya adalah menyimpan kedua data tersebut (*data training* dan *data testing*) dalam format *file CSV* untuk kebutuhan proses *deployment.* Kemudian, *data training* (*`X_train'* dan *`y_train`*) kembali dibagi menjadi *training set* dan *validation set* dengan perbandingan 2:9. Hasil pembagian ini disimpan sebagai *`X_train`, `y_train`, `X_val`*, dan *`y_val`*.

3.3.4 *Training Process*

Langkah selanjutnya dari penelitian ini adalah melatih model menggunakan *data training*. Dalam penelitian ini, algoritma yang dipilih merupakan algoritma *Extreme Gradient Boosting* (XGBoost) karena kemampuannya dalam mengolah data yang besar dan kompleks. XGBoost mampu mendeteksi pola-pola yang menunjukkan potensi *inflated bills* dengan memprediksi lonjakan biaya tagih. Tabel 3.7 dibawah ini

merupakan contoh data yang digunakan dalam proses pelatihan model menggunakan algoritma *Extreme Gradient Boosting* (XGBoost)

Tabel 3.7 Contoh *Data Training*

Variable	**U1**	**U2**	**U3**
FKL09	1,916910278	-0,3863239964	-0,6166474238
FKL10	-1,047623392	-1,047623392	-1,047623392
FKL13	-2,529347128	-1,013090169	0,50316679
FKL17	-1,843370843	-1,651625617	-1,096056119
FKL18	-1,60665937	-1,501771633	-1,127800173
FKL20	-0,5336502392	-2,270504006	-0,7073356159
FKL21	-0,2594784258	-0,2594784258	-1,383580068
FKL22	-0,4658500041	-0,7626883202	-0,2432212671
FKL23	0,6980264424	0,6980264424	2,122305254
FKL30	0,8418900352	0,1222889768	-0,03841400984
FKL31	0,5084362977	0,5084362977	-0,9547483309
FKL32	-0,1136320726	-0,1334362614	2,974535154
FKL33	-0,01319628443	-0,01319628443	-0,01319628443
FKL34	-0,01151475107	-0,01151475107	-0,01151475107
FKL35	-0,07833712891	-0,07833712891	-0,07833712891
FKL37	-0,06940314038	-0,06940314038	-0,06940314038
FKL38	-0,02699323539	-0,02699323539	-0,02699323539
FKL40	-0,02274097303	-0,02274097303	-0,02274097303
FKL41	0,04323988538	0,04323988538	0,04323988538
FKL43	-0,04249989236	-0,04249989236	-0,04249989236
FKL44	0,02861652043	0,02861652043	0,02861652043
FKL46	-0,02897008307	-0,02897008307	-0,02897008307
FKL47	-0,1205793538	-0,1393312878	2,803504447
LOS	0,3836372693	0,08700075662	-0,2096357561
FKL48 (Target)	-0,1198934437	-0,1386803586	2,809645127

Pada Tabel 3.7 tersebut terlihat bahwa terdapat 24 variabel *input* dan 1 variabel *output* (FKL48) atau *target* yang ingin di prediksi. Data tersebut juga terdapat 3 baris yang diwakilkan oleh *U1, U2*, dan *U3*.

Langkah pertama dalam memprediksi biaya verifikasi menggunakan algoritma *Extreme Gradient Boosting* atau XGBoost adalah menentukan nilai awal atau *base model* y_i^0, yang umumnya merupakan nilai-rata-rata dari *target*. Dari data yang diberikan, y_i^0 dapat dihitung sebagai berikut:

$$y_i^0 = \frac{-0{,}1198934437 + -0{,}1386803586 + 2{,}809645127}{3} = \frac{2{,}5511}{3} = 0{,}85$$

Hasil y_i^0 sebesar 0,85 akan digunakan sebagai prediksi awal untuk semua data. Langkah selanjutnya adalah melakukan perhitungan *gradient* (g_i) dan *hessian* (h_i). *Gradient* dihitung sebagai selisih antara nilai sebenarnya atau nilai aktual dari target (y) dan nilai prediksi awal atau *base model* (y_i^0). *Gradient* juga merupakan turunan pertama dari *loss function.* Secara matematis, *gradient* ditulis sebagai berikut:

$$g_i = \frac{\partial L}{\partial \hat{y}} = y - \hat{y}$$

Sedangkan *hessian* adalah turunan kedua dari *loss function*. Secara matematis, *hessian* dihitung dengan persamaan:

$$h_i = \frac{\partial^2 L}{\partial \hat{y}^2} = 1$$

Berikut ini merupakan perhitungan *gradient* dan *hessian* untuk setiap data:

- Data 1 (U1)

  y = -0,1198934437 = -0.1199

  $\hat{y}$ = 0,85

  $g_i = \hat{y} - \text{y} = 0{,}85 - (-0{,}1199) = 0.85 + 0{,}1199 = 0{,}9699 = 0{,}97$

  $h_1 = 1$

- Data 2 (U2)

  y = -0,1386803586 = -0,139

  $\hat{y}$ = 0,85

$g_2 = \hat{y} - \mathrm{y} = 0{,}85 - (-0{,}139) = 0.85 + 0{,}139 = 0{,}989 = 0{,}99$

$h_2 = 1$

- Data 3 (U3)

  y = 2,809645127 = 2,81

  $\hat{y}$ = 0,85

  $g_3 = \hat{y} - \mathrm{y} = 0{,}85 - 2{,}81 = -1{,}96$

  $h_3 = 1$

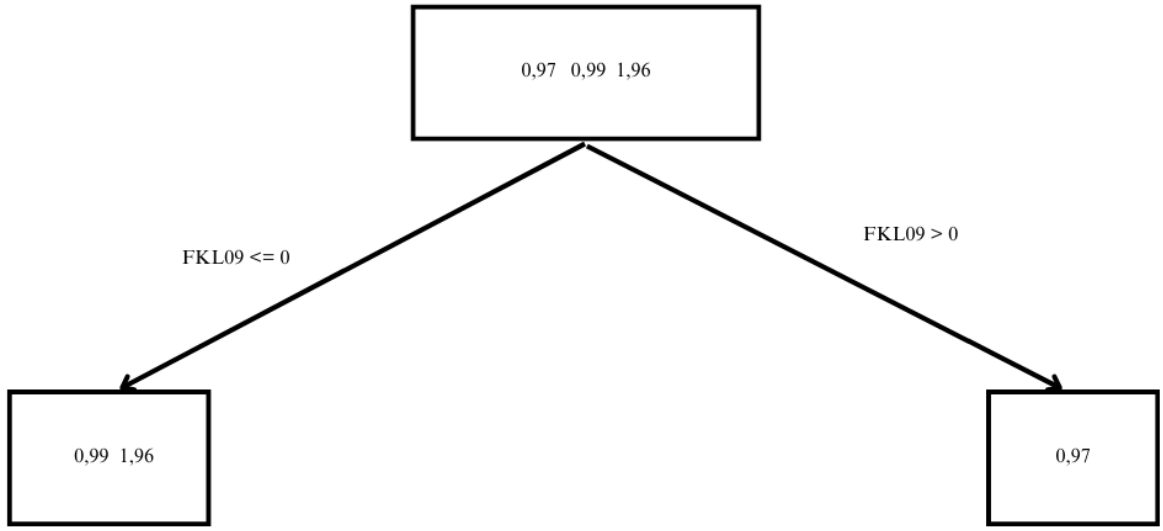
Setelah menghitung *gradien* dan *hessian*. Langkah selanjutnya adalah menghitung bobot (w_a) dari daun atau *node* dalam pohon keputusan yang akan dibangun oleh XGBoost dengan menggunakan rumus berikut ini:

$$w_a = -\frac{\sum g_a}{\sum h_a + \lambda}$$

Jika parameter regularisasi λ = 1, maka bobot (w_a) dapat dihitung sebagai berikut:

$$w_a = -\frac{(0{,}97 + 0{,}99 + (-1{,}96)}{(1+1+1)+1} = 0$$

Hasil perhitungan tersebut menunjukkan bahwa (w_a) = 0 yang menandakan bahwa belum ada perubahan signifikan pada daun. Langkah selanjutnya adalah melakukan pembagian daun berdasarkan nilai pada kolom FKL09 dengan kriteria FKL09 ≤ 0 dan FKL09 > 0. Pembagian daun tersebut terlihat seperti Gambar 3.10 dibawah ini:



Gambar 3.10 Pembagian Daun (*Node*)

Pada Gambar 3.15 diatas terlihat bahwa pada daun kiri terdapat dua residual yaitu 0,99 dan 1,96. Sedangkan pada daun kanan, terdapat satu residual sebesai 0,97. Untuk daun kiri, nilai residu total O_L dan P_L dihitung sebagai berikut:

$$O_L = 0{,}99 + (-1{,}96) = -0{,}97$$

$$P_L = 2 \text{ (Karena terdapat 2 data)}$$

Sedangkan untuk daun kanan, nilai O_R dan P_R dihitung sebagai berikut:

$$O_R = 0{,}97$$

$$P_R = 1 \text{ (Karena terdapat 1 data)}$$

Setelah itu, dihitung nilai *information gain* menggunakan Persamaan 2.9 seperti yang tertulis dibawah ini:

$$G = \frac{1}{2}\left[\frac{O_L^2}{P_L + \lambda} + \frac{O_R^2}{P_R + \lambda} + \frac{(O_L + O_R)^2}{P_R + P_L + \lambda}\right]$$

Dengan nilai λ = 1, maka nilai *gain* menjadi:

$$G = \frac{1}{2}\left[\frac{(-0{,}97)^2}{2 + 1} + \frac{0{,}97^2}{1+1} + \frac{(0{,}97 + (-0{,}97))^2}{2+1+1}\right]$$

$$G = 0{,}392$$

Setelah mendapatkan nilai *gain*, bobot daun untuk masing-masing sisi juga dihitung kembali. Untuk sisi kiri, bobot daun menjadi:

$$w_L = -\frac{0{,}99 + (-1{,}96)}{(1+1)+1} = 0{,}323$$

Sedangkan untuk daun kanan:

$$w_R = -\frac{0{,}97}{1+1} = -0{,}485$$

Setelah itu, fungsi objektif dihitung dengan menggunakan Persamaan 2.8 yang tertulis di bawah ini"

$$f_{obj} = \gamma Z + \sum_{a=1}^{Z}\left[g_a\omega_a + \frac{1}{2}(h_a+\lambda)\omega_a^2\right]$$

Dengan parameter γ = 0.1, Z = 2, dan λ = 1, Maka fungsi objektif dihitung sebagai berikut:

$$f_{obj} = 0.1(2) + ((0{,}99(0{,}323) + (-1{,}96)(0{,}323) + \tfrac{1}{2}(2+1)(0{,}323)^2)) + ((0{,}97(-0{,}485) + \tfrac{1}{2}(1+1)(-0{,}485)^2))$$

Setelah melakukan perhitungan, hasil akhir fungsi objektif adalah sebagai berikut:

$$f_{obj} = -0{,}192$$

Hasil akhir pada fungsi objektif menunjukkan estimasi setelah pembagian, dimana hal tersebut akan digunakan untuk iterasi selanjutnya. Setelah dilakukan beberapa iterasi atau pengulangan, prediksi akhir untuk setiap data dapat dihitung dengan Persamaan 2.6 seperti yang tertulis dibawah ini:

$$\bar{y} = y_i^0 + \eta\sum_{K=1}^{n} f_k(U_i)$$

dimana y_i^0 adalah *base model* (nilai awal = 0,85) , η adalah *learning rate*, dan $f_k(U_i)$ merupakan *output* dari masing-masing pohon pada iterasi ke-*k*. Hasil prediksi

akan diperbaharui pada setiap iterasi, dimana prediksi akhir merupakan hasil akumulasi dari nilai awal atau *base model* (y_i^0) dan perbaikan yang ada pada setiap pohon keputusan yang dibangun dalam proses *boosting*. Dengan cara tersebut, algoritma *Extreme Gradient Boosting* (XGBoost) akan terus mengurangi *error* dengan menambahkan pohon keputusan baru yang menangkap sisa *error* dari iterasi sebelumnya sehingga menghasilkan prediksi yang lebih akurat. *Pseudocode* dari proses pelatihan (*training process*) terlihat seperti dibawah ini:

```
SET PARAMS. = {
        Set 'objective`
        Set `eval_metrics` to `mae`
        Set `learning_rate`
        Set `n_estimators`
        Set `max_depth`}

CREATE regressor AS XGBRegressor WITH enable_categorical=True AND
params

CALL regressor.fit WITH arguments:
        training data(X_train, y_train)
        evaluation set (X_train, y_train, X_val, and y_val)
```

Pada *pseudocode* diatas yang menjelaskan proses *training model* menggunakan XGBoost. Proses ini diawali dengan menetapkan parameter utama untuk model. Parameter tersebut mencakup *objective***,** *eval_metric* (metrik evaluasi yang digunakan yaitu MAE), *learning_rate***,** *n_estimators***,** dan *max_depth***.** Selanjutnya, dibuat sebuah variabel untuk menyimpan model regresi yang menggunakan XGBRegressor. Langkah berikutnya adalah memanggil metode fit() pada model regresi untuk memulai proses pelatihan. Proses ini memerlukan argumen berupa *data training* (X_train dan y_train) serta *data validation* (X_val dan y_val).

3.3.4.1 *Parameter Tuning*

Parameter tuning merupakan proses untuk menemukan kombinasi parameter paling optimal dan terbaik untuk meningkatkan kinerja model. *Parameter tuning* dilakukan untuk mencegah terjadinya *overfitting* dan *underfitting*. Dalam penelitian ini, penulis melakukan *tuning* terhadap beberapa parameter pada model, yakni:

1. *n_estimators* (Jumlah pohon keputusan)

   Parameter ini menentukan jumlah pohon keputusan yang akan digunakan oleh model. Semakin banyak pohon keputusan yang digunakan, semakin besar kemungkinan model untuk menangkap pola dalam data. Namun, jika pohon terlalu banyak, model berpotensi untuk terlalu kompleks dan rentan terhadap *overfitting*. Oleh karena itu, *tuning* pada parameter ini membantu untuk menemukan keseimbangan jumlah pohon agar model yang dihasilkan tidak terbebani. Pada penelitian ini, nilai *n_estimators* yang diuji adalah 50, 100, 200, dan 300.

2. *max_depth* (kedalaman maksimal pohon keputusan)

   Parameter ini berfungsi untuk menentukan seberapa detail model belajar dari data. Semakin besar kedalamannya, semakin mampu pula model dalam menangkap lebih banyak interaksi kompleks antar fitur. Namun, jika kedalamannya tidak seimbang, maka hal tersebut dapat menyebabkan model menangkap *noise* dan menjadi *overfitting*. Pada penelitian ini, nilai *max_depth* yang diuji adalah 2, 4, 6, dan 8

3. *learning_rate* (kecepatan belajar)

   Parameter mengontrol seberapa cepat model belajar dari kesalahan yang terjadi pada setiap iterasi atau pengulangan. *Learning rate* yang terlalu tinggi dapat menyebabkan model melewati solusi yang optimal, sementara *learning rate* yang terlalu rendah dapat menyebabkan model menjadi lambat dan tidak efisien. Pada penelitian ini, nilai *learning_rate* yang diuji adalah 0.01, 0.05, dan 0.1.

Proses *parameter tuning* dilakukan sebanyak 48 kali (4 nilai pada *n_estimators* x 3 nilai pada *learning_rate* x 4 nilai pada *max_depth*) dengan mencoba seluruh kombinasi dari tiga parameter yaitu *n_estimators, learning_rate,* dan *max_depth*. Pada parameter *n_estimators*, dilakukan pengujian pada nilai 50, 100, 200, dan 300. Pada parameter *learning_rate*, dilakukan pengujian pada 3 nilai diantaranya 0.01, 0.05, dan 0.1.

Semantara itu, pada parameter *max_depth* dilakukan pengujian pada nilai 2, 4, 6, dan 8.

48 kombinasi tersebut diuji secara menyeluruh untuk mendapatkan kombinasi parameter yang menghasilkan kinerja model terbaik. Kriteria pemilihan kombinasi parameter terbaik dilakukan dengan menggunakan nilai *Mean Absolute Error* (MAE) pada data latih (*data training*) dan data validasi (*data validation*).

3.3.5 *Model Selection*

Setelah proses pelatihan model (*training process*) dan penyesuaian parameter (*parameter tuning*) dengan berbagai kombinasi parameter, langkah selanjutnya adalah memilih model terbaik (*best model*) berdasarkan hasil evaluasinya pada *data validation*. Pemilihan model (*model selection*) ini diambil berdasarkan nilai *Mean Absolute Error* (MAE) pada *data validation.* Hal tersebut bertujuan untuk menemukan model dengan tingkat kesalahan yang paling rendah pada data baru sehingga dapat digunakan untuk membuat data prediksi yang optimal.

3.3.6 *Testing Model*

Setelah menemukan model terbaik dengan kombinasi parameter yang paling optimal, langkah selanjutnya adalah melakukan pengujian model (*testing model*) yang dilakukan menggunakan *subset* data uji (*data testing*). *Data testing* adalah kumpulan data yang terpisah dari *data training* dan *data validation* sehingga tidak terlihat selama proses pelatihan (*training process*) sebelumnya. *Testing model* ini bertujuan untuk melihat seberapa baik model bekerja pada data yang baru. Hasil dari *testing model* ini berupa prediksi biaya verifikasi (*verification price*) yang kemudian dapat dibandingkan dengan data asli untuk mengukur ketepatan model dalam memperkirakan biaya. *Pseudocode* untuk proses pengujian (*testing model*) terlihat seperti dibawah ini

```
SET y_pred TO regressor.predict(X_test)
```

3.3.6 *Evaluation*

Tahap evaluasi (*evaluation*) diperlukan untuk menilai seberapa baik hasil pengujian model (*testing model*) dalam melakukan prediksi. Dalam penelitian ini, kinerja model diukur menggunakan metrik evaluasi seperti *Mean Absolute Error* (MAE), *Mean Square Error* (MSE), dan *Root Mean Squared Error* (RMSE). Ketiga metrik tersebut digunakan untuk mengevaluasi seberapa jauh perbedaan antara nilai yang dihasilkan oleh model dengan nilai aktualnya. Metrik MSE dan RMSE memberikan bobot yang besar pada perbedaan yang besar karena mengkuadratkan hasilnya, sedangkan MAE memberikan gambaran yang lebih sederhana mengenai rata-rata perbedaan (*error*) karena tidak mengkuadratkan hasilnya. Semakin kecil nilai MAE, MSE, dan RMSE yang diperoleh, maka semakin rendah pula tingkat kesalahan yang dihasilkan oleh model. Hal tersebut juga menunjukkan performa model yang semakin baik. Berikut ini merupakan *pseudocode* yang digunakan pada tahap evaluasi (*evaluation*):

```
import module mean squred_error_from sklearn.metrics
import module mean_absolute_error from sklearn.metrics
import numpy

Calculate Mean Squared Error using module mean_squared_error

Calculate Mean Absolute Error using module mean_absolute_error

Calculate Root Mean Squared Error using square_root of MSE

PRINT the evaluation metrics
```

3.4 Rancangan Antarmuka Sistem

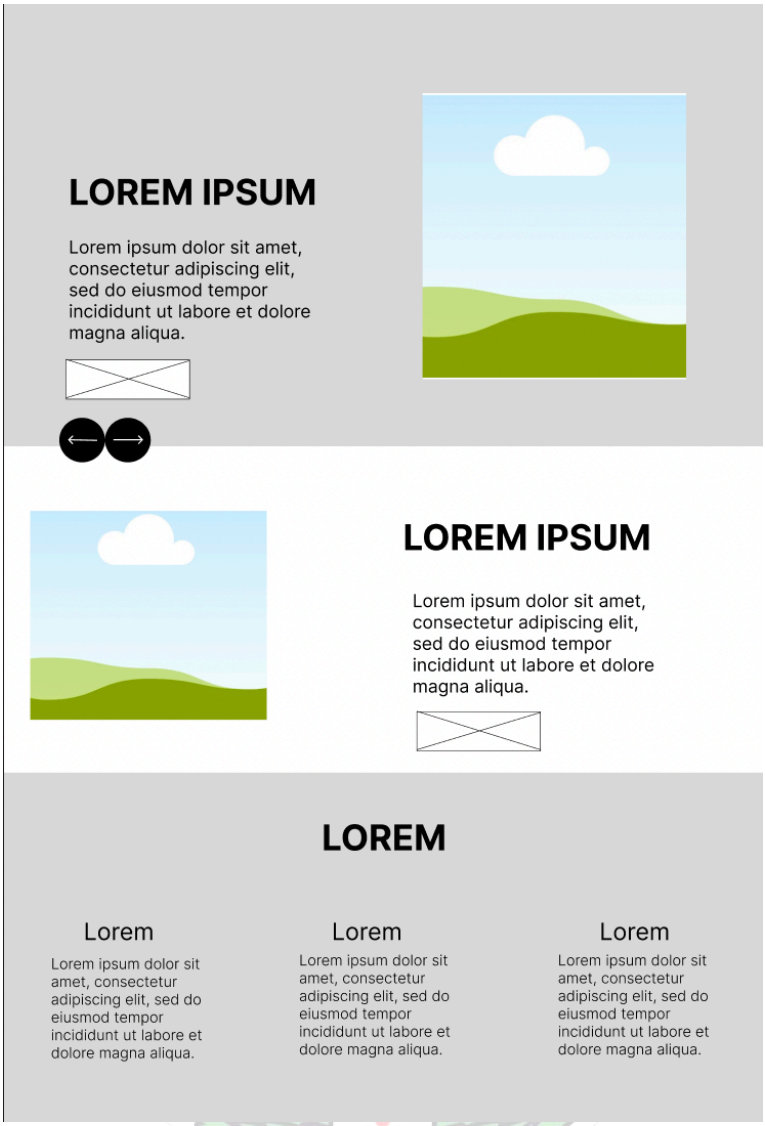
Setelah model berhasil dibangun dan menunjukkan kinerja yang baik, langkah selanjutnya adalah melakukan implementasi model tersebut dalam bentuk aplikasi web. Implementasi ini sangat penting untuk memudahkan pengguna dalam mengakses dan menggunakan model yang telah dikembangkan. Sebelum melanjutkan ke tahap implementasi, diperlukan sebuah rancangan antarmuka (*user interface*) dari sistem

yang akan dibangun. Rancangan antarmuka ini berfungsi untuk memberikan gambaran yang jelas mengenai tata letak, fitur, dan interaksi pengguna dengan sistem.

3.4.1 Rancangan Antarmuka *Landing Page*

Landing page adalah halaman yang pertama dilihat oleh pengguna ketika mengakses sebuah *website*. Pada *landing page*, pengguna akan menemukan beberapa informasi singkat terkait penelitian yang dilakukan. Terdapat penjelasan singkat terkait apa itu *inflated bills* dan mengapa deteksi *inflated bills* itu perlu dilakukan. Selain itu, terdapat juga penjelasan tekait pemilihan algoritma *Extreme Gradient Boosting* (XGBoost) sebagai metode utama dalam mendeteksi *inflated bills*.

Landing page juga menampilkan penjelasan singkat terkait fitur-fitur yang tersedia didalamnya seperti fitur *train, test,* dan *claim.* Fitur *train* memungkinkan pengguna untuk melatih dan memvalidasi model. Fitur *test* memungkinkan pengguna untuk mengevaluasi kinerja model. Sedangkan, fitur *claim* memungkinkan pengguna untuk memverifikasi klaim dan menerima hasil prediksi biaya verifikasi untuk mengidentifikasi adanya potensi *inflated bills.* Pada halaman ini juga tersedia tombol "Start Now" yang akan mengarahkan pengguna ke halaman *train* untuk membangun model. Rancangan antarmuka *landing page* dapat dilihat pada Gambar 3.11 dibawah ini:

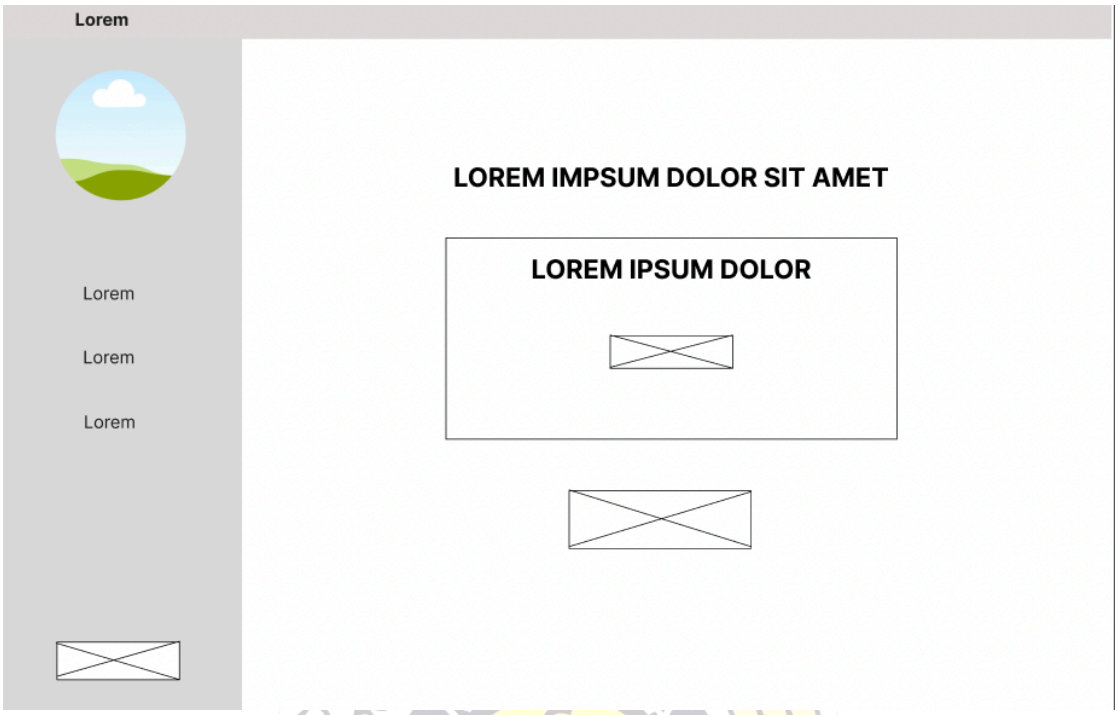


Gambar 3.11 Rancangan Antarmuka *Landing Page*

3.4.2 Rancangan Antarmuka *Training Page*

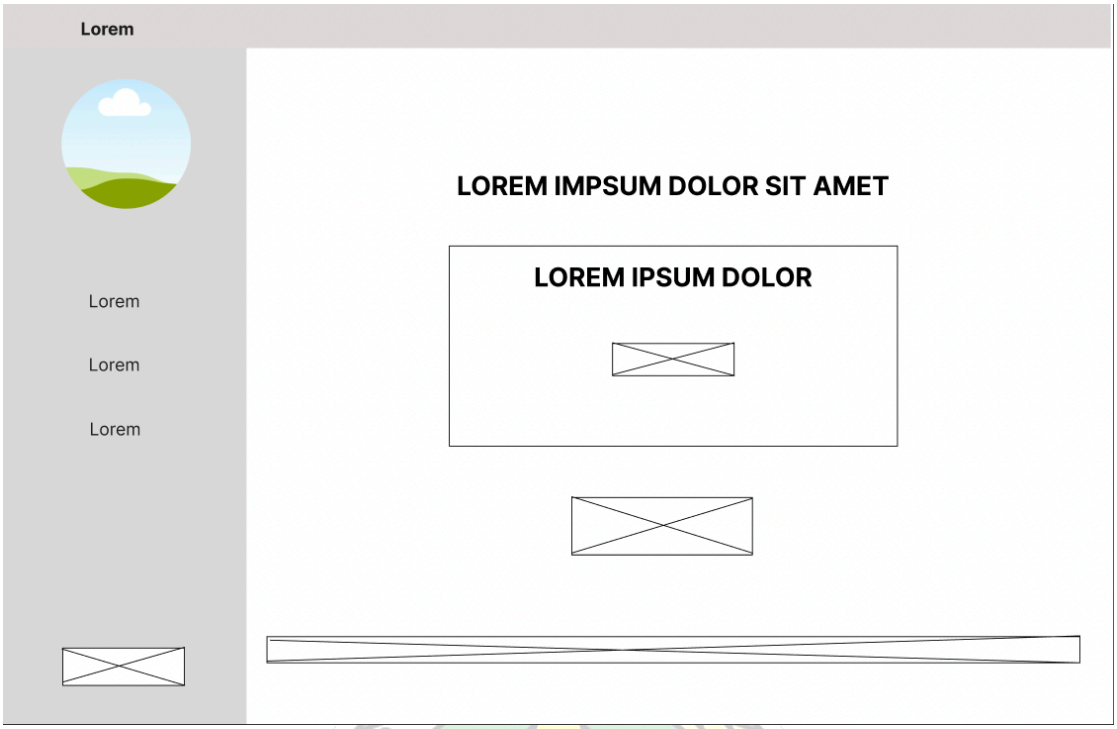
Halaman yang pertama muncul ketika pengguna menekan tombol "*Start*" adalah halaman pelatihan atau *training*. Pada halaman ini, terdapat beberapa elemen penting yang dapat memudahkan pengguna dalam proses pelatihan model. Di sisi kiri, terdapat *sidebar* yang berisi tiga opsi utama yakni halaman *training, testing*, dan *claim*. *Sidebar* tersebut dirancang untuk memudahkan navigasi antar halaman.Selain itu, halaman *training* dilengkapi dengan sebuah formulir *input* yang memungkinkan pengguna untuk mengunggah dataset *training* dalam format .csv. Setelah dataset *training* berhasil diunggah, pengguna dapat memulai proses pelatihan model dengan menekan tombol

"*TRAIN MODEL*". Rancangan antarmuka *training page* dapat dilihat pada Gambar 3.12 dibawah ini:



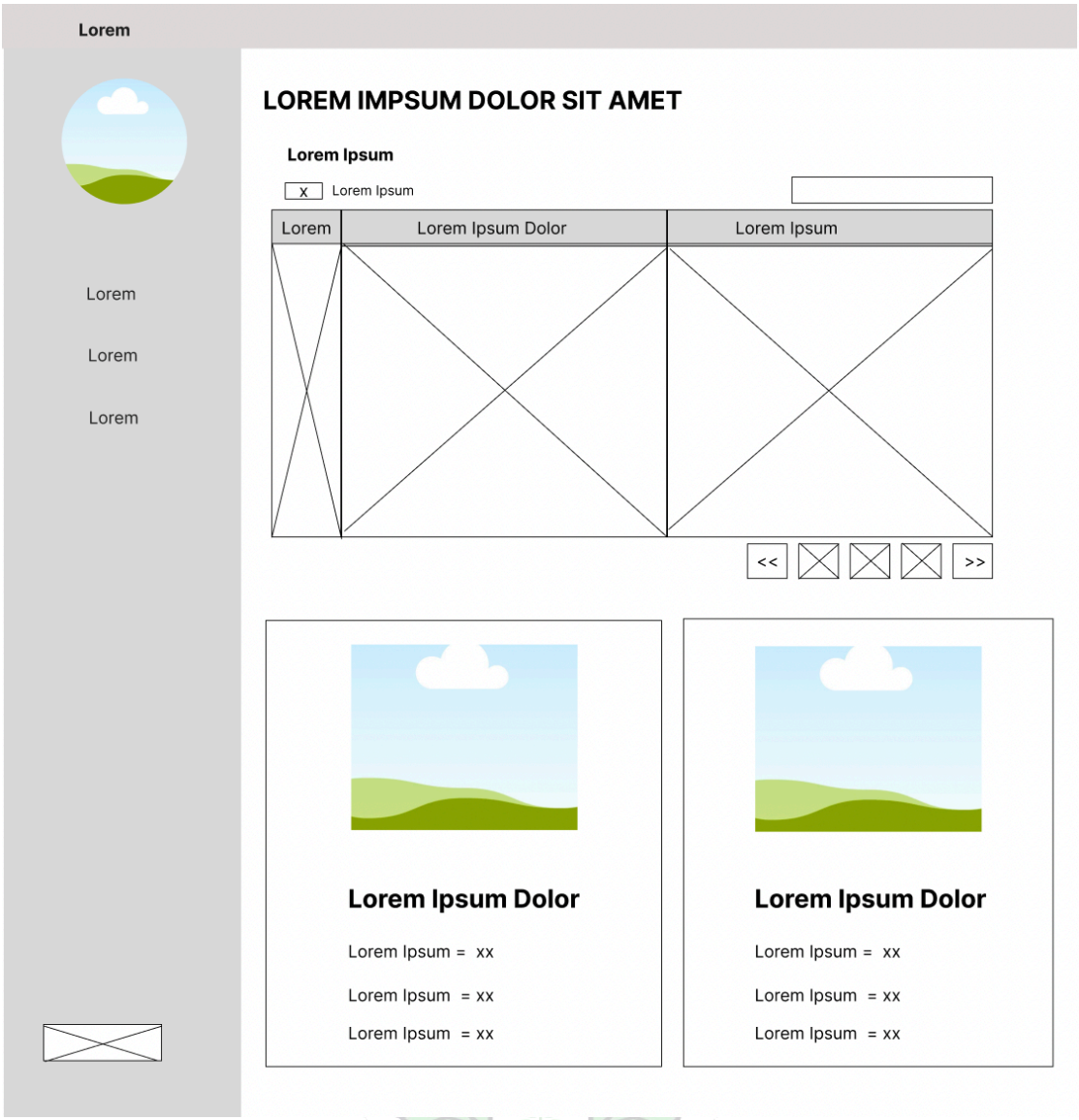
Gambar 3.12 Rancangan Antarmuka *Training Page*

Setelah pengguna menekan tombol "*TRAIN MODEL*", halaman akan menampilkan *progress bar* yang menunjukkan tahapan-tahapan dalam proses pelatihan model yang sedang dilalui. *Progress bar* tersebut memberikan informasi mengenai status dan pengembangan pelatihan model. Rancangan antarmuka halaman *training* setelah pengguna menekan tombol "*TRAIN MODEL"* terlihat seperti Gambar 3.13 dibawah ini:



Gambar 3.13 Rancangan Antarmuka *Training Page* Lanjutan

Setelah seluruh tahapan selesai, pengguna akan diarahkan ke halaman *output* yang menampilkan hasil performa model secara menyeluruh. Pada halaman ini, tersedia informasi mendetail terkait kinerja model seperti nilai *Mean Absolute Error* pada *data training* dan *data validation* pada setiap iterasi parameter *n_estimators.* Selain itu, ditampilkan pula grafik yang menggambarkan perkembangan nilai *Train MAE* dan *Validation MAE* sepanjang proses pelatihan serta nilai akhir keduanya. Selain itu, pengguna juga dapat melihat visualisasi dari *feature engineering* yang menunjukkan peran dari masing-masing fitur terhadap hasil prediksi. Rancangan antarmuka *output training process* terlihat seperti Gambar 3.14 dibawah ini:

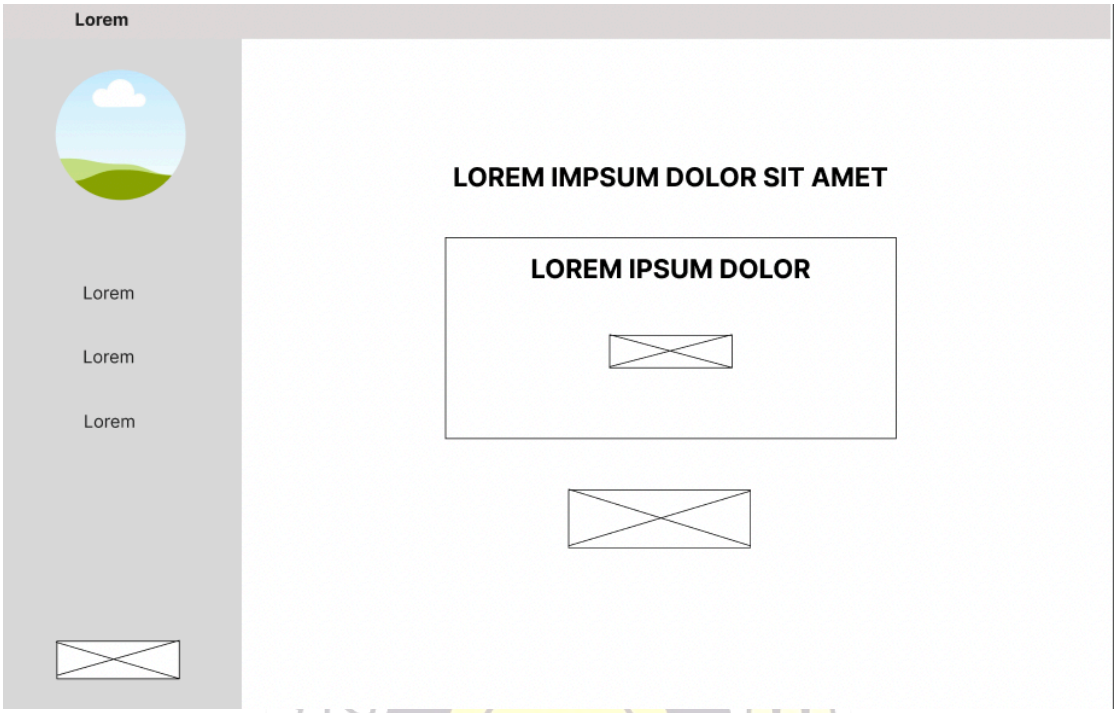


Gambar 3.14 Rancangan Antamuka Halaman Hasil *Training*

3.4.3 Rancangan Antarmuka *Testing Page*

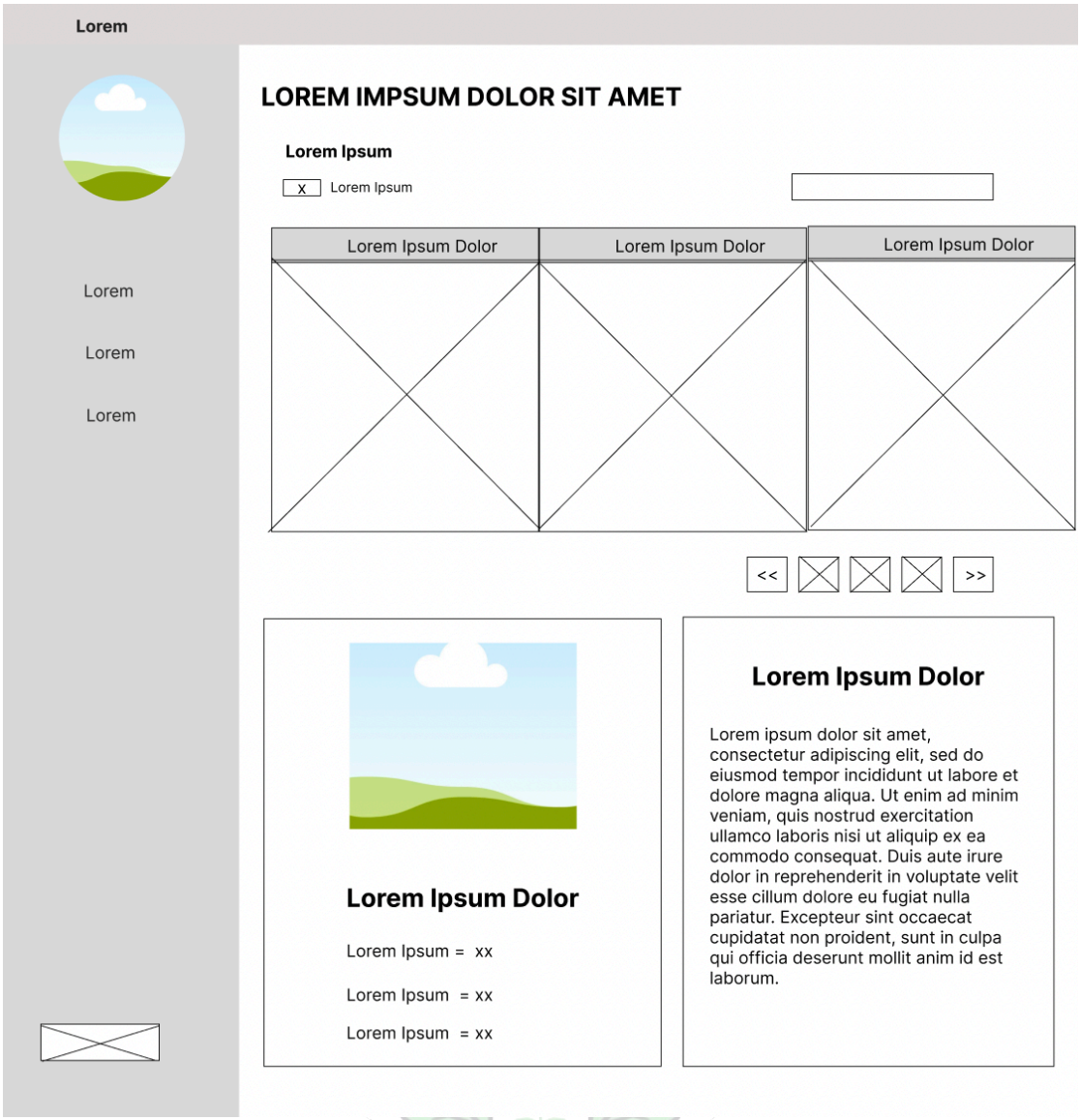
Halaman pengujian atau *testing page* memiliki desain yang serupa dengan halaman *training*. Namun, halaman ini berfungsi untuk menguji model yang telah dibangun sebelumnya pada halaman *training* menggunakan *data testing*. Pada halaman ini, terdapat sebuah formulir yang memungkinkan pengguna untuk mengunggah *data testing* dalam format *Comma Separated Value* (.csv). Setelah *data test* diunggah, pengguna dapat menekan tombol "*TEST MODEL*" untuk memulai proses pengujian model. Proses ini akan mengaktifkan model yang telah dibangun dan menguji kinerjanya terhadap *data testing*. Setelah tombol "*TEST MODEL*" ditekan, *progress bar*

akan status dari setiap tahapan proses pengujian. Rancangan antarmuka *testing page* dapat dilihat pada Gambar 3.15 dibawah ini:



Gambar 3.15 Rancangan Antarmuka *Testing Page*

Setelah proses pengujian atau *testing* model selesai, pengguna akan diarahkan ke halaman *output* yang menyajikan hasil prediksi secara mendetail. Pada halaman ini, pengguna dapat melihat perbandingan antara biaya tagih aktual, biaya verifikasi aktual, dan prediksi biaya verifikasi yang dihasilkan oleh model. Pada halaman ini juga tersedia visualisasi grafik yang membandingkan biaya verifikasi aktual dengan prediksi biaya verifikasi. Selain itu, halaman ini juga menampilkan nilai-nilai dari metrik evaluasi utama pada *data testing* yaitu, *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE), dan *Root Mean Squared Error* (RMSE). Metrik-metrik ini membantu memberikan informasi mengenai tingkat kesalahan atau *error* yang dibuat oleh model. Rancangan antarmukan halaman hasil *testing* terlihat seperti Gamabr 3.16 dibawah ini:



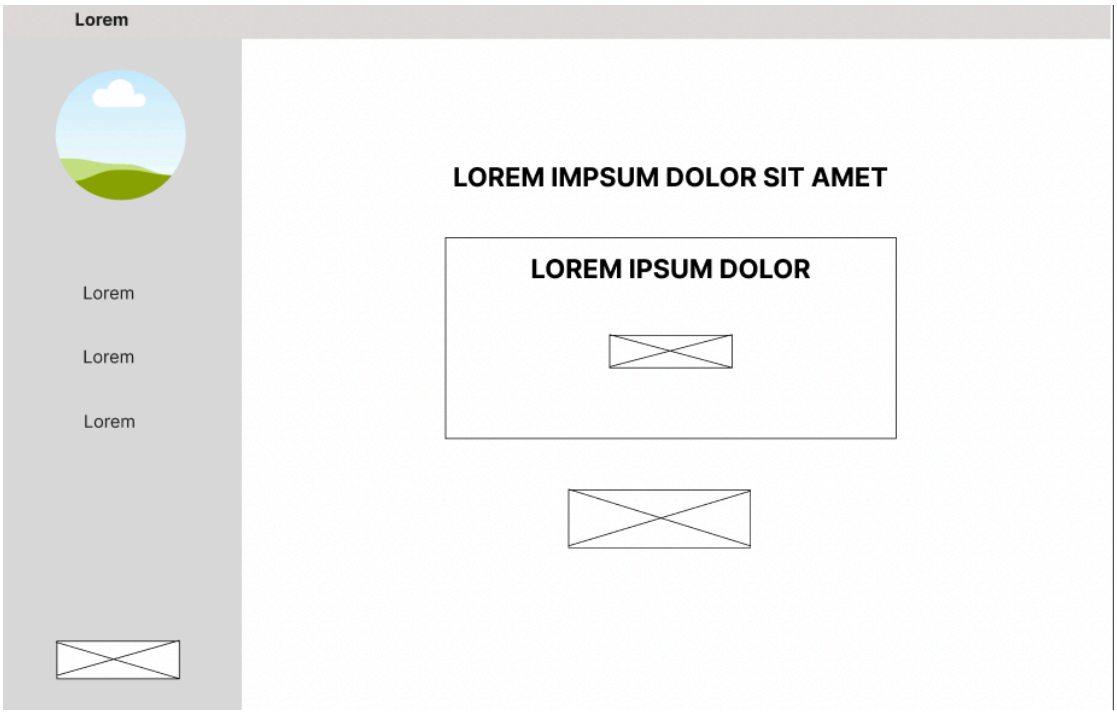
Gambar 3.16 Rancangan Antarmuka Hasil *Testing*

3.4.4 Rancangan Antarmuka Halaman *Claim*

Halaman *claim* merupakan halaman yang bertujuan untuk mempermudah pengguna dalam memverfikasi banyak klaim secara otomatis serta mendeteksi adanya potensi *inflated bills*. Proses tersebut dilakukan dengan memprediksi biaya verifikasi menggunakan model yang telah dibangun sebelumnya dan membandingkannya dnegan biaya tagihan yang diajukan oleh fasilitas kesehatan. Jika biaya tagihan yang diajukan oleh fasilitas kesehatan lebih besar dari biaya verifikasi yang diprediksi oleh sistem, maka sistem akan menandai klaim tersebut sebagai klaim dengan potensi *inflated bills*.

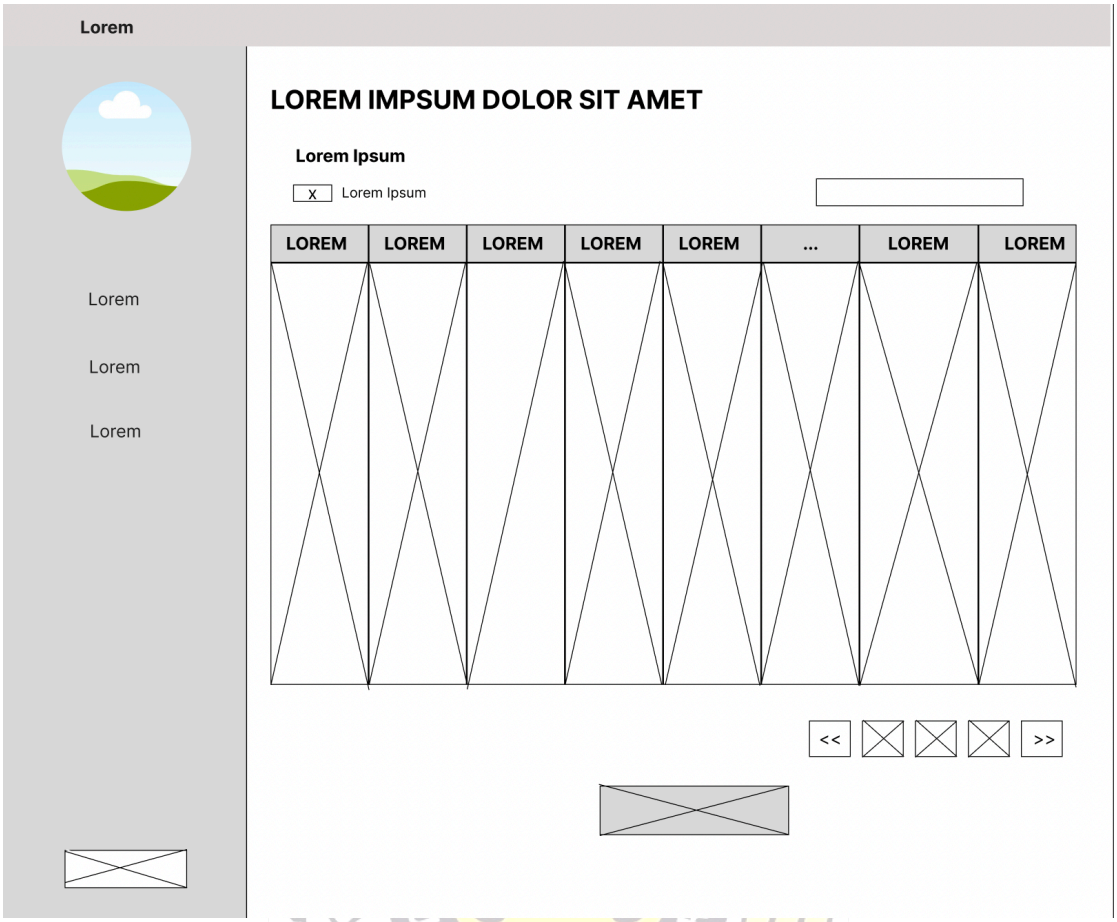
Dengan adanya sistem tersebut, pengguna dapat dengan mudah mengidentifikasi klaim yang mencurigakan dan memerlukan verifikasi lebih lanjut.

Halaman *claim* memiliki desain yang serupa dnegan halaman *training* dan *testing*. Pada halaman ini, terdapat sebuah *form* yang memungkinkan pengguna untuk mengunggah kumpulan klaim yang diajukan oleh fasilitas kesehatan dan perlu diverifikasi dalam format .csv. Setelah file diunggah, pengguna dapat menekan tombol "*Detect"* untuk memulai proses verifikasi. Selama proses berlangsung, terdapat sebuah *progress bar* yang menunjukkan status dari proses verifikasi. Rancangan antarmuka halaman *claim* terlihat seperti Gambar 3.17 dibawah ini:



Gambar 3.17 Rancangan Antarmuka Halaman *Claim*

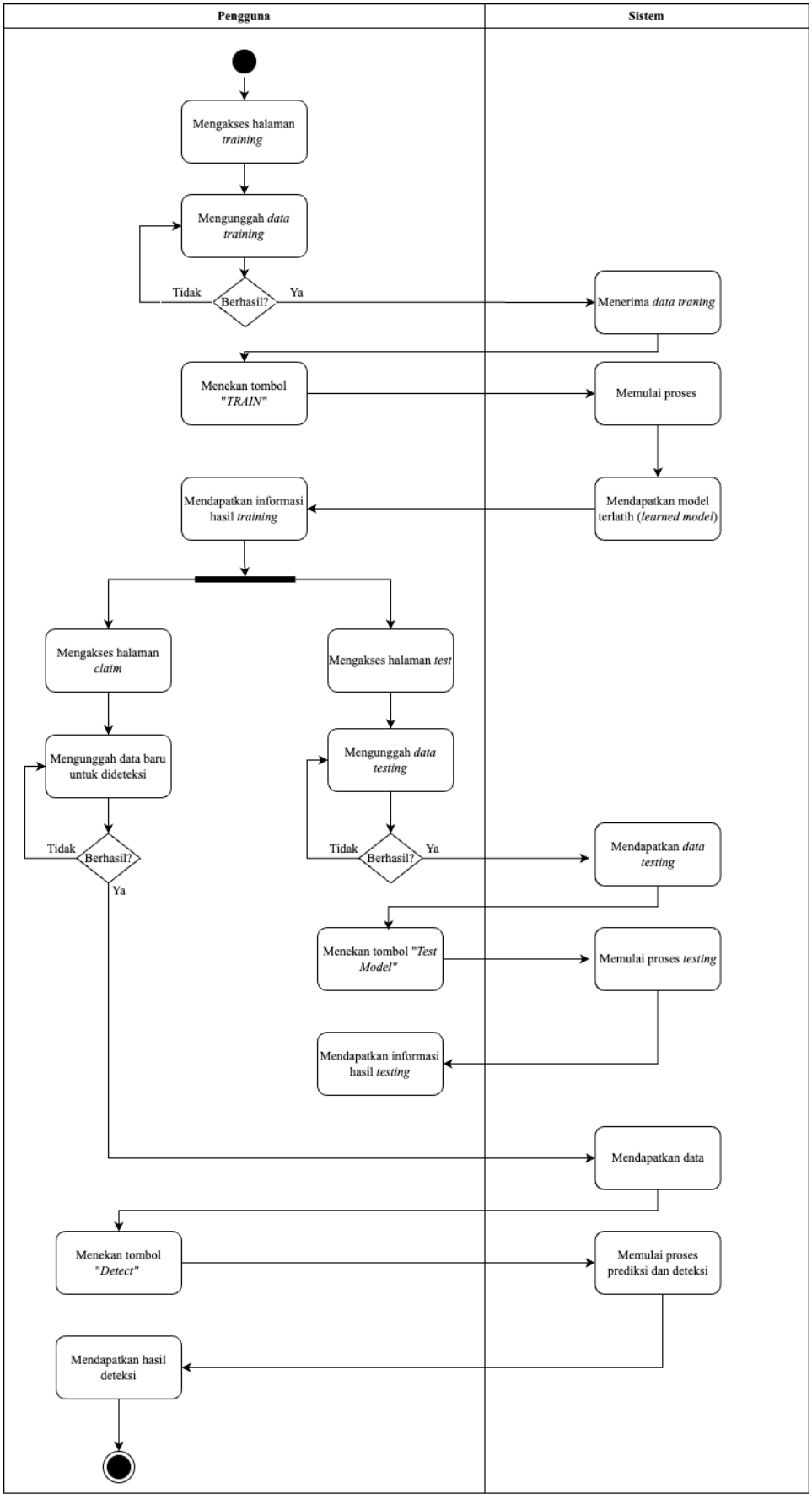
Setelah seluruh proses verifikasi dan deteksi potensi *inflated bills* selesai, halaman akan berganti ke halaman berikutnya yang menampilkan hasil deteksi potensi *inflated bills* dalam bentuk tabel. Tabel tersebut mencakup informasi-informasi dari klaim, prediksi biaya verifikasi, dan status yang menunjukkan apakah klaim tersebut normal atau berpotensi *inflated bills*. Selain itu, terdapat juga tombol "*Download Results*" yang memungkinkan pengguna untuk mengunduh hasil deteksi. Rancangan antarmuka halaman hasil deteksi terlihat seperti Gambar 3.18 dibawah ini:



Gambar 3.18 Rancangan Antarmuka Halaman Hasil Deteksi

3.5 *Activity Diagram*

Activity diagram atau diagram aktivitas merupakan representasi grafis terhadap setiap aktivitas yang dijalankan dalam sebuah sistem. Pada penelitian ini, penulis menyusun sebuah *activity diagram* untuk memvisualisasikan proses dalam sistem deteksi potensi *inflated bills* pada data klaim layanan BPJS Kesehatan menggunakan algoritma *Extreme Gradient Boosting (*XGBoost). *Diagram activity* pertama pada Gambar 3.19 dibawah ini menunjukkan alur kerja secara umum dari keseluruhan halaman (*page*) pada sistem.



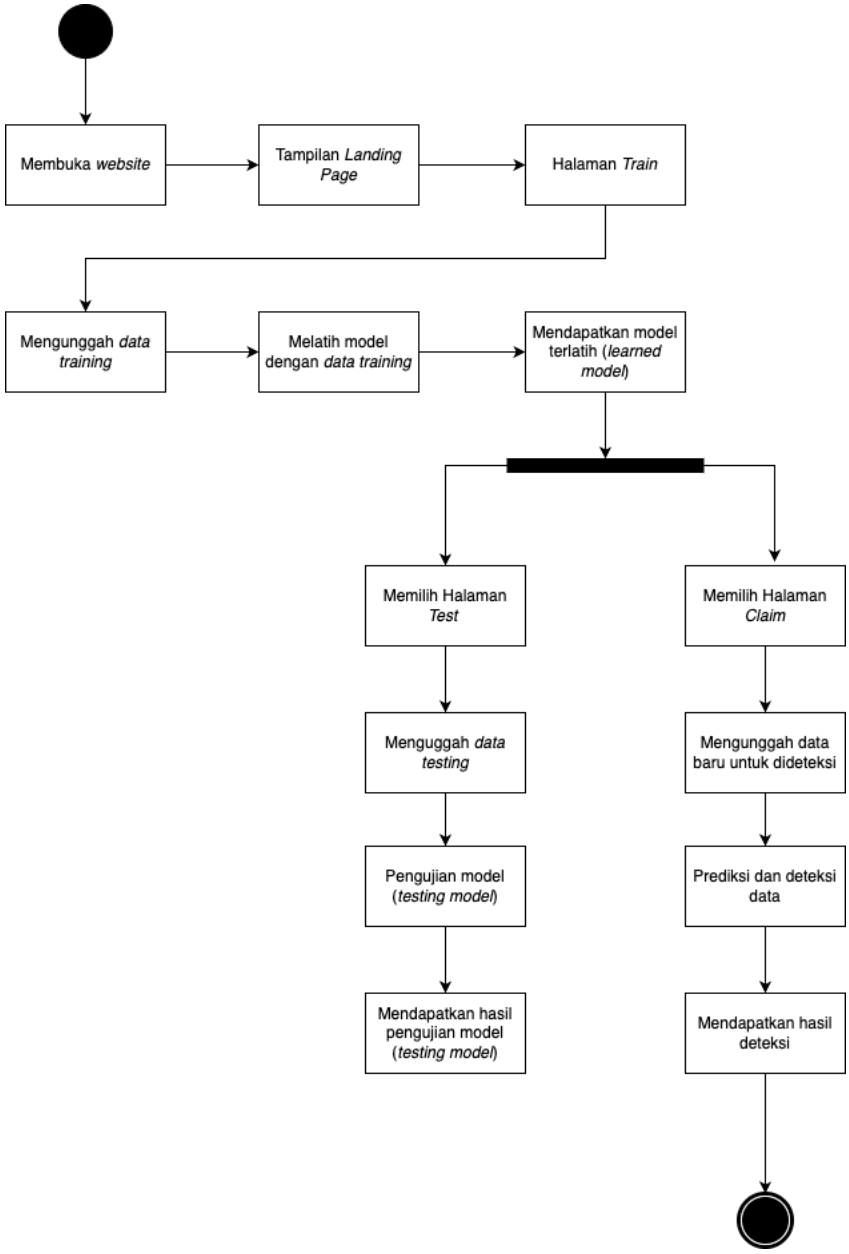
Gambar 3.19 *Activity Diagram* Sistem

Gambar 3.19 diatas merupakan *activity diagram* dari sistem yang menunjukkan alur interaksi antara pengguna dengan sistem. Proses dimulai dengan pengguna mengakses halaman *training* dan mengunggah *data training*. Jika proses unggah data tersebut berhasil, maka sistem akan menyimpan data tersebut. Namun, jika terjadi kegagalan, pengguna dapat mengunggah kembali data tersebut. Setelah data berhasil diunggah, pengguna dapat menekan tombol "*TRAIN*" sebagai tanda untuk sistem memulai proses pelatihan. Sistem menjalankan proses pelatihan menggunakan data yang telah diunggah dan menghasilkan sebuah model yang tersimpan dalam sistem. Model tersebut nantinya digunakan untuk menguji kinerja model (*testing model*) maupun untuk mendeteksi potensi *inflated bills* pada data baru.

Setelah proses *training* selesai, pengguna dapat memilih fitur mana untuk digunakan terlebih dahulu antara fitur *test* dan *claim*. Kedua fitur tersebut memiliki alur proses yang serupa dengan fitur *train* namun menghasilkan *output* yang berbeda. Fitur *test* dirancang untuk menguji kinerja model yang telah dibangun. Hasil dari fitur ini mencakup metrik-metrik evaluasi seperti *Mean Absolute Error, Mean Squared Error*, dan *Root Mean Squared Error* serta tabel perbandingan antara nilai aktual dengan nilai hasil prediksi model. Sementara itu, fitur *claim* digunakan untuk deteksi pada data baru. Hasil dari fitur ini berupa prediksi biaya verifikasi yang dapat digunakan sebagai alat untuk deteksi potensi *inflated bills*.

3.5.1 Alur Pengunaan Aplikasi

Alur pengunaan aplikasi merupakan tahapan-tahapan yang dilakukan pengguna dalam mengakses aplikasi maupun sistem. Gambar 3.20 dibawah ini menunjukkan alur penggunaan aplikasi:



Gambar 3.20 Alur Pengunaan Aplikasi

Pada Gambar 3.20 diatas terlihat sebuah Alur pengunaan aplikasi. Proses dimulai ketika pengguna membuka *website* dan secara otomatis menampilkan *landing page*. Setelah itu, pengguna diarahkan menuju halaman *train* sebagai langkah awal untuk menghasilkan model. Pada halaman *train,* pengguna diminta untuk mengunggah *data training*. Lalu, sistem akan mempelajari *data training* tersebut menggunakan algoritma *Extreme Gradient Boosting* (XGBoost) dengan kombinasi parameter yang telah ditetapkan sehingga menghasilkan model terlatih (*learned model*) yang siap digunakan. Setelah mendapatkan model, pengguna dapat memilih untuk melajutkan ke halaman *test* ataupun halaman *claim.*

Jika pengguna memilih melanjutkan ke halaman *tezt,* pengguna akan diminta untuk mengunggah *data testing*. Model yang telah dihaslkan sebelumnya akan diuji menggunakan *data testing* tersebut dan hasil pengujian ditampilkan kepada pengguna untuk mengevaluasi performa model.

Jika pengguna memilih untuk melanjutkan ke halaman *claim*, pengguna akan diminta untuk mengunggah mengunggah data baru yang ingin diprediksi biaya verifikasi dan dideteksi potensi *inflated bills.* Sistem akan memproses data tersebut dengan model yang dibangun dan dilatih sebelumnya, melakukan prediksi biaya verifikasi, dan mendeteksi potensi *inflated bills* pada klaim yang diunggah.

BAB 4

IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi Sistem

Sistem yang dibangun dalam penelitian ini memerlukan beberapa komponen untuk melakukan pelatihan model (*training model*) untuk algoritma *Extreme Gradient Boosting* (XGBoost) berupa perangkat keras dan perangkat lunak. Berikut adalah informasi perangkat keras dan perangkat lunak yang digunakan:

4.1.1 Spesifikasi Perangkat Keras

Perangkat keras yang digunakan mencakup komputer atau laptop dengan spesifikasi tertentu yang dirancang untuk memenuhi kebutuhan analisis dan pelatihan model. Spesifikasi tersebut mencakup merek, tipe, prosesor, penyimpanan dan RAM. Informasi lengkap mengenai spesifikasi perangkat keras yang digunakan pada penelitian ini terdapat pada Tabel 4.1 dibawah ini:

Tabel 4.1 Spesifikasi Perangkat Keras

Merek	Apple
Tipe	Macbook Air
Prosesor	M1 Chip
Penyimpanan	256 GB
RAM	8 GB

4.1.2 Spesifikasi Perangkat Lunak

Perangkat lunak yang digunakan dalam penelitian ini mencakup aplikasi atau *software* yang digunakan untuk proses pelatihan model (*training model*). Adapun spesifikasi perangkat lunak yang digunakan untuk pelatihan model (*training model*) algoritma *Extreme Gradient Boosting* (XGBoost) terdapat dalam Tabel 4.2 dibawah ini:

Tabel 4.2 Spesifikasi Perangkat Lunak

Operation System	macOS-12.5-arm64-arm-64bit
Tool	*Jupyter Notebook* dan *visual studio code*
Bahasa Pemrograman	*Python* versi 3.11.5
Library	*pandas, numpy, matplotlib, sklearn,* dan *XGBoost*

4.2 Implementasi Tahap *Preprocessing Data*

Pada tahap ini, data yang telah dikumpulkan dimuat ke dalam *Jupyter Notebook* untuk dilakukan proses preprocessing, yang bertujuan untuk mengolah data mentah (*raw data*) menjadi data yang lebih terstruktur dan berkualitas. Proses ini sangat penting untuk memastikan bahwa analisis dan model yang dibangun selanjutnya dapat menghasilkan *output* yang baik dan dapat diandalkan. Beberapa tahapan yang dilakukan dalam preprocessing data meliputi *data cleaning* yang bertujuan untuk mengidentifikasi dan menghapus data yang tidak relevan atau berkualitas rendah, *feature engineerin* yang berfokus pada pengembangan fitur baru dari data yang ada untuk meningkatkan performa model, *label encoding* yang digunakan untuk mengubah kategori data menjadi format numerik agar dapat diproses oleh algoritma *machine learning,* dan *normalization data* yang bertujuan untuk menyamakan skala nilai antar fitur agar model

dapat bekerja dengan lebih efektif. Dengan mengikuti tahapan ini, diharapkan data yang digunakan akan lebih siap dan optimal untuk tahapan selanjutnya.

4.2.1 Implementasi *Data Cleaning*

Pada tahap *data cleaning*, penulis menerapkan beberapa tahap seperti penghapusan kolom yang tidak relevan, menghapus nilai-nilai yang hilang (*missing values*), dan menghapus data yang duplikat. Beberapa kolom yang dihapus termasuk identitas pasien seperti nomor peserta (PSTV01) dan nomor keluarga (PSTV02) yang bersifat personal dan tidak relevan untuk digunakan. Kolom lain seperti bobot (PSTV15), nomor asal rujukan (FKP02), dan ID kunjungan (FKL02) juga dihapus karena tidak mempengaruhi besar biaya tagih.

Informasi geografis seperti provinsi dan kabupaten/kota FKRTL (FKL05 dan FKL06) serta detail lain seperti kepemilikan FKRTL (FKL07), jenis FKRTL (FKL08), dan jenis poli (FKL11). Kolom-kolom tersebut dihapus juga karena tidak berkaitan langsung dengan estimasi biaya. Segmen peserta (FKL12), status pulang (FKL14), data diagnosis saat masuk (FKL15, FKL15A, FKL16, dan FKL16A), kode ICD-10 yang redundan (FKL17A, FKL18A, FKL19, dan FKL19A), data tentang fasilitas kesehatan perujuk (FKL15, FKL26, FKL27, FKL28,dan FKL29), dan deskripsi pelayanan tambahan seperti (FKL36, FKL29, FKL42, dan FKL45). Kolom-kolom tersebut dihapus karena sudah diwakilkan oleh kolom yang lain dan tidak menjadi komponen penentuan biaya yang terdapat dalam Peranturan Menteri Kesehatan No.3 Tahun 2023 tentang Standar Tarif BPJS Kesehatan. Setelah dilakukan penghapusan kolom yang tidak diperlukan tersebut, langkah selanjutnya adalah menghapus data yang hilang (*missing values*) dan menghapus data yang duplikat. Hasil dari tahap *data cleaning* dapat dilihat pada Gambar 4.1 dibawah ini:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 723476 entries, 0 to 723475
Data columns (total 26 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   FKL03   723476 non-null  datetime64[ns]
 1   FKL04   723476 non-null  datetime64[ns]
 2   FKL09   723476 non-null  category
 3   FKL10   723476 non-null  category
 4   FKL13   723476 non-null  category
 5   FKL17   723476 non-null  category
 6   FKL18   723476 non-null  object
 7   FKL20   723476 non-null  category
 8   FKL21   723476 non-null  category
 9   FKL22   723476 non-null  int8
 10  FKL23   723476 non-null  category
 11  FKL30   723476 non-null  object
 12  FKL31   723476 non-null  category
 13  FKL32   723476 non-null  int32
 14  FKL33   723476 non-null  object
 15  FKL34   723476 non-null  int32
 16  FKL35   723476 non-null  object
 17  FKL37   723476 non-null  int32
 18  FKL38   723476 non-null  object
 19  FKL40   723476 non-null  int32
 20  FKL41   723476 non-null  object
 21  FKL43   723476 non-null  int32
 22  FKL44   723476 non-null  object
 23  FKL46   723476 non-null  int32
 24  FKL47   723476 non-null  int32
 25  FKL48   723476 non-null  float64
dtypes: category(8), datetime64[ns](2), float64(1), int32(7), int8(1), object(7)
memory usage: 81.5+ MB
```

Gambar 4.1 Hasil Implementasi *Data Cleaning*

Pada Gambar 4.1 di atas dapat dilihat adanya perubahan yang signifikan pada dataset yang digunakan. Pada awalnya, dataset tersebut terdiri dari 1.176.437 baris dan 55 kolom, mencerminkan informasi yang sangat luas dan beragam. Namun, setelah proses *data cleaning* dilakukan, ukuran *dataset* berkurang menjadi 723.476 baris dan 26 kolom. Perubahan ini terjadi akibat penghapusan 29 kolom yang dianggap tidak relevan untuk analisis yang akan dilakukan, sehingga tidak memberikan kontribusi yang berarti terhadap hasil penelitian. Selain itu, beberapa baris yang memiliki *missing values* dan *duplicated values* juga dihapus untuk memastikan kualitas dan integritas data.

4.2.2 Implementasi Tahap *Feature Engineering*

Setelah data dibersihkan, langkah selanjutnya adalah *feature engineering* yang merupakan tahap pengembangan fitur-fitur dalam dataset. Pada proses ini, terdapat tiga langkah yang dilakukan oleh penulis, yaitu *feature construction, feature selection,* dan *feature improvement*. Pada tahap *feature construction*, penulis akan menghasilkan fitur baru dari fitur-fitur yang sudah ada seperti fitur *Length of Stay* (LOS) yang mengindikasi durasi perawatan pasien. Pada tahap *feature selection,* penulis akan memilih kembali kolom yang sesuai dengan kebutuhan pembuatan model. Sedangkan pada tahap *feature improvement*, penulis melakukan upaya dalam meningkatkan kualitas data dengan membersihkan data kembali.

Dalam pembuatan fitur *Length of Stay* (LOS), peneliti menggunakan operasi pengurangan antara fitur tanggal pulang kunjungan (FKL04) dan tanggal datang kunjungan (FKL03). Setelah tahap *feature construction* selesai, langkah selanjutnya adalah *feature selection*. Pada tahap ini penulis memutuskan untuk menghilangkan kolom tanggal datang kunjungan (FKL03) dan tanggal pulang kunjungan (FKL04). Hal tersebut dilakukan karena informasi yang terdapat dalam kolom tersebut sudah diwakilkan oleh kolom *length of stay* yang telah dibuat sebelumnya.

Langkah berikutnya dalam *feature engineering* adalah *feature improvement* yang merupakan upaya peningkatan kualitas dari fitur-fitur yang ada. Pada tahap ini, peneliti melakukan proses pembersihan data kembali dengan menghapus data yang hilang dan data yang duplikat. Hasil dari seluruh proses *feature engineering* terlihat pada Gambar 4.2 berikut ini:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 251341 entries, 0 to 251340
Data columns (total 25 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   FKL09   251341 non-null  category
 1   FKL10   251341 non-null  category
 2   FKL13   251341 non-null  category
 3   FKL17   251341 non-null  category
 4   FKL18   251341 non-null  object
 5   FKL20   251341 non-null  category
 6   FKL21   251341 non-null  category
 7   FKL22   251341 non-null  int8
 8   FKL23   251341 non-null  category
 9   FKL30   251341 non-null  object
 10  FKL31   251341 non-null  category
 11  FKL32   251341 non-null  int32
 12  FKL33   251341 non-null  object
 13  FKL34   251341 non-null  int32
 14  FKL35   251341 non-null  object
 15  FKL37   251341 non-null  int32
 16  FKL38   251341 non-null  object
 17  FKL40   251341 non-null  int32
 18  FKL41   251341 non-null  object
 19  FKL43   251341 non-null  int32
 20  FKL44   251341 non-null  object
 21  FKL46   251341 non-null  int32
 22  FKL47   251341 non-null  int32
 23  FKL48   251341 non-null  float64
 24  LOS     251341 non-null  int64
dtypes: category(8), float64(1), int32(7), int64(1), int8(1), object(7)
memory usage: 26.4+ MB
```

Gambar 4.2 Hasil Implementasi *Feature Engineering*

Pada Gambar 4.2 merupakan hasil implementasi dari *feature engineering* yang mencakup beberapa perubahan penting pada dataset. Salah satu perubahan yang terjadi adalah adanya kolom LOS atau *Length of Stay* yang dihasilkan dari pengolahan data kunjungan pasien. Selain itu, kolom-kolom yang sebelumnya berisi informasi tanggal, seperti tanggal datang kunjungan (FKL03) dan tanggal pulang kunjungan (FKL04) kini telah dihapus.

4.2.3 Implementasi Tahap *Label Encoding*

Setelah menyelesaikan tahap *feature engineering*, langkah selanjutnya adalah melakukan tahap *label encoding* untuk setiap tipe data kategorik dan objek. Proses ini dilakukan dengan bantuan pustaka `*sklearn.preprocessing*` dan fungsi `*LabelEncoder*`.

Proses ini bertujuan untuk mengubah tipe data kategorik dan objek tersebut menjadi numerik. Kini seluruh variabel dalam dataset memiliki tipe data numerik atau *int64* seperti terlihat pada Gambar 4.3 dibawah ini:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 251341 entries, 0 to 251340
Data columns (total 25 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   FKL09   251341 non-null  int64
 1   FKL10   251341 non-null  int64
 2   FKL13   251341 non-null  int64
 3   FKL17   251341 non-null  int64
 4   FKL18   251341 non-null  int64
 5   FKL20   251341 non-null  int64
 6   FKL21   251341 non-null  int64
 7   FKL22   251341 non-null  int64
 8   FKL23   251341 non-null  int64
 9   FKL30   251341 non-null  int64
 10  FKL31   251341 non-null  int64
 11  FKL32   251341 non-null  int64
 12  FKL33   251341 non-null  int64
 13  FKL34   251341 non-null  int64
 14  FKL35   251341 non-null  int64
 15  FKL37   251341 non-null  int64
 16  FKL38   251341 non-null  int64
 17  FKL40   251341 non-null  int64
 18  FKL41   251341 non-null  int64
 19  FKL43   251341 non-null  int64
 20  FKL44   251341 non-null  int64
 21  FKL46   251341 non-null  int64
 22  FKL47   251341 non-null  int64
 23  FKL48   251341 non-null  float64
 24  LOS     251341 non-null  int64
dtypes: float64(1), int64(24)
memory usage: 47.9 MB
```

Gambar 4.3 Hasil Implementasi Tahap *Label Encoding*

Gambar 4.3 diatas menunjukkan hasil implementasi tahap *label encoding* bahwa kolom-kolom yang sebelumnya bersifat object maupun category telah berubah menjadi numerik. Perubahan tersebut dibutuhkan untuk mempermudah proses pelatihan model.

4.2.4 Implementasi Tahap *Normalization Data*

Setelah menyelesaikan tahap *label encoding,* langkah selanjutnya adalah melakukan normalisasi atau standarisasi pada dataset. Hal tersebut dilakukan karena nilai yang terdapat dalam dataset memiliki rentang yang beragam dan dapat mempengaruhi kinerja model. Pada penelitian ini, penulis menerapkan `*StandardScaler*`. Langkah pertama dalam proses ini adalah mendefinisikan fitur-fitur dan target yang akan digunakan secara terpisah. Kemudian membuat *scaler* untuk fitur-fitur dan target secara terpisah. Setelah itu, dilakukan normalisasi data menggunakan *scaler* yang telah dibuat sebelumnya. Terakhir, seluruh data yang sudah dinormalisasi digabungkan kembali ke dalam sebuah variabel. Hasil dari proses normalisasi data terlihat seperti Gambar 4.4 dibawah ini:

	FKL09	FKL10	FKL13	FKL17	FKL18	FKL20	FKL21	FKL22	FKL23	FKL30	...	FKL37	FKL38	FKL40	FKL4
0	-0.386324	0.954541	0.503167	-0.422490	-0.569863	-0.707336	-0.821529	-0.762688	-0.726252	1.012752	...	-0.069403	-0.026993	-0.022741	0.0432
1	-0.386324	-1.047623	0.503167	-1.622126	-1.483101	-2.270504	-0.259478	-0.688479	0.698026	0.785222	...	-0.069403	-0.026993	-0.022741	0.0432
2	-0.386324	-1.047623	0.503167	-1.622126	-1.483650	-2.270504	-0.259478	-0.688479	0.698026	0.785222	...	-0.069403	-0.026993	-0.022741	0.0432
3	-0.386324	-1.047623	0.503167	-1.843371	-1.606659	-0.533650	-0.259478	-0.465850	0.698026	0.785222	...	-0.069403	-0.026993	-0.022741	0.0432
4	-0.386324	0.954541	0.503167	1.396632	1.381817	1.550574	-0.821529	0.276246	-0.726252	1.067376	...	-0.069403	-0.026993	-0.022741	0.0432
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
251336	-0.156001	-1.047623	0.503167	-0.424948	-0.572060	-0.707336	-0.259478	-0.465850	2.122305	0.816228	...	-0.069403	-0.026993	-0.022741	0.0432
251337	-0.156001	-1.047623	0.503167	-0.424948	-0.573158	-0.707336	-0.259478	-0.465850	2.122305	1.013478	...	-0.069403	-0.026993	-0.022741	0.0432
251338	-0.156001	0.954541	0.503167	0.307617	0.451007	1.550574	-0.821529	0.127827	-0.726252	0.448511	...	-0.069403	-0.026993	-0.022741	0.0432
251339	-0.156001	0.954541	0.503167	-0.641276	-0.747788	0.508462	1.426674	0.498875	-0.726252	0.711599	...	-0.069403	-0.026993	-0.022741	0.0432
251340	-0.156001	0.954541	0.503167	1.367133	1.352712	0.508462	1.426674	1.537809	-0.726252	-0.050948	...	-0.069403	-0.026993	-0.022741	0.0432

251341 rows x 25 columns

Gambar 4.4 Hasil Implementasi Tahap *Normalization Data*

Setelah normalisasi metode *StandardScaler* (*Z-Score Normalization*) selesai, data sekarang berada dalam rentang yang lebih baik seperti yang ditunjukkan pada Gambar 4.4 di atas. Dibandingkan dengan kondisi sebelumnya, normalisasi ini berhasil mengubah skala nilai dataset dan menghasilkan rentang yang lebih konsisten.

4.3 Implementasi Tahap *Training* Model

Tahap pelatihan model atau *training model* diawali dengan proses pencarian kombinasi parameter terbaik atau *parameter tuning*. Pada tahap ini, penulis melakukan pengujian terhadap beberapa kombinasi untuk tiga parameter utama yaitu *n_estimators, learning_rate*, dan *max_depth*. Parameter *n_estimators* diuji pada empat nilai berbeda yaitu 50, 100, 200, dan 300. Sedangkan, parameter *learning_rate* diuji pada tiga nilai berbeda yaitu 0.01, 0.05, dan 0.1. Sementara itu, parameter *max_depth* diuji pada empat nilai berbeda yaitu 2, 4, 6, dan 8.

Setiap kombinasi nilai parameter diuji secara bergantian sehingga mendapatkan 48 total kombinasi (4 nilai pada *n_estimators* x 3 nilai pada *learning_rate* x 4 nilai pada *max_depth*). Tujuan dari *parameter tuning* ini adalah untuk mencari kombinasi terbaik yang dapat menghasilkan performa terbaik pada model. Tabel 4.3 dibawah ini menunjukkan rincial dari setiap kombinasi yang diuji beserta nilai *Mean Absolute Error* (MAE) yang diperoleh pada *data training* (*Train MAE*) dan *data validation* (*Validation MAE*).

Tabel 4.3 Hasil Implementasi *Parameter Tuning*

n_estimators	learning_rate	max_depth	Train MAE	Validation MAE
50	0.01	2	0.31227	0.3093
50	0.01	4	0.29135	0.28822
50	0.01	6	0.29017	0.28737
50	0.01	8	0.29031	0.28771
50	0.05	2	0.09189	0.09049
50	0.05	4	0.04477	0.04361
50	0.05	6	0.04172	0.04126
50	0.05	8	0.04013	0.04054
50	0.1	2	0.04193	0.04063
50	0.1	4	0.01939	0.01862
50	0.1	6	0.01116	0.0117

n_estimators	learning_rate	max_depth	Train MAE	Validation MAE
50	0.1	8	0.00845	0.01
100	0.01	2	0.21156	0.20922
100	0.01	4	0.17849	0.1762
100	0.01	6	0.17663	0.17468
100	0.01	8	0.17667	0.17507
100	0.05	2	0.04206	0.04073
100	0.05	4	0.01954	0.01894
100	0.05	6	0.01149	0.01201
100	0.05	8	0.00855	0.01023
100	0.1	2	0.02851	0.02739
100	0.1	4	0.01397	0.01357
100	0.1	6	0.00872	0.01007
100	0.1	8	0.00597	0.00867
200	0.01	2	0.11062	0.10907
200	0.01	4	0.07076	0.06939
200	0.01	6	0.06768	0.06688
200	0.01	8	0.06667	0.06655
200	0.05	2	0.02551	0.02434
200	0.05	4	0.01335	0.01311
200	0.05	6	0.00884	0.0101
200	0.05	8	0.00585	0.00868
200	0.1	2	0.0241	0.02312
200	0.1	4	0.01148	0.01194
200	0.1	6	0.00673	0.0093
200	0.1	8	0.0041	0.00828

n_estimators	learning_rate	max_depth	Train MAE	Validation MAE
300	0.01	2	0.07865	0.07731
300	0.01	4	0.03236	0.03124
300	0.01	6	0.02924	0.02901
300	0.01	8	0.02737	0.02812
300	0.05	2	0.02274	0.0217
300	0.05	4	0.01186	0.01196
300	0.05	6	0.00768	0.00959
300	0.05	8	0.00468	0.0083
300	0.1	2	0.0222	0.02134
300	0.1	4	0.01016	0.01147
300	0.1	6	0.00577	0.00892
300	0.1	8	0.00332	0.0082

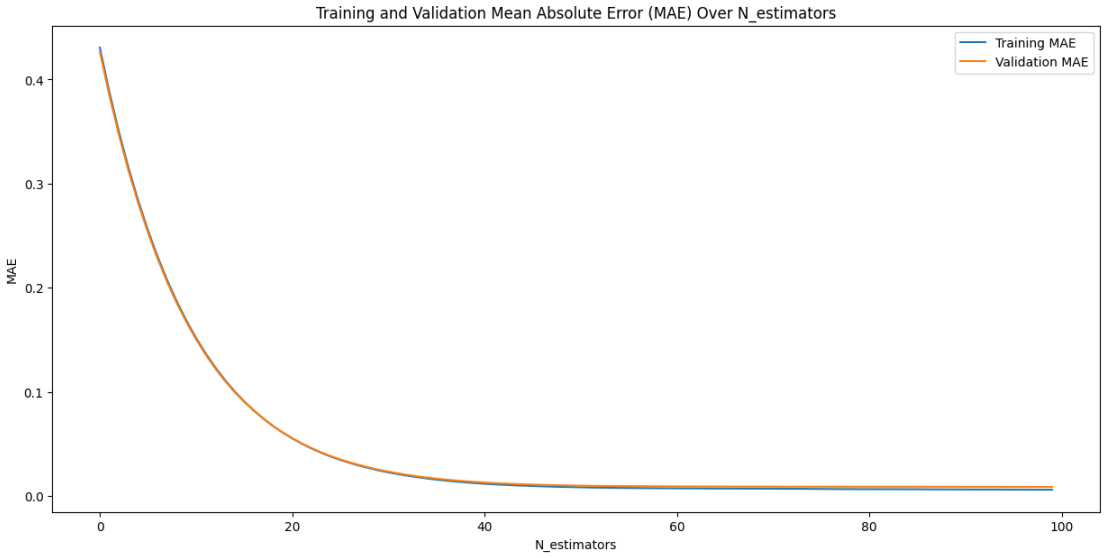
4.4 Implementasi Tahap *Model Selection*

Pada tahap pelatihan model (*training model*) sebelumnya, proses *parameter tuning* dilakukan untuk menentukan kombinasi parameter yang paling baik dan optimal. Dari proses ini, ditemukan bahwa kombinasi parameter *n_estimator* sebesar 100, *learning_rate* sebesar 0.1, dan *max_depth* sebesar 8. Kombinasi tersebut dipilih berdasarkan hasil evaluasi menggunakan nilai *Mean Absolute Error* (MAE). Pada *data validation*, nilai MAE tercatat sebesar 0.00867, sementara pada *data training* sebesar 0.00597. Selain nilai MAE yang rendah di kedua subset data, selisih antara keduanya juga kecil sehingga tidak *overfit* maupun *underfit* Tabel 4.4 dibawah ini merangkum kombinasi parameter

Tabel 4.4 Parameter Akhir

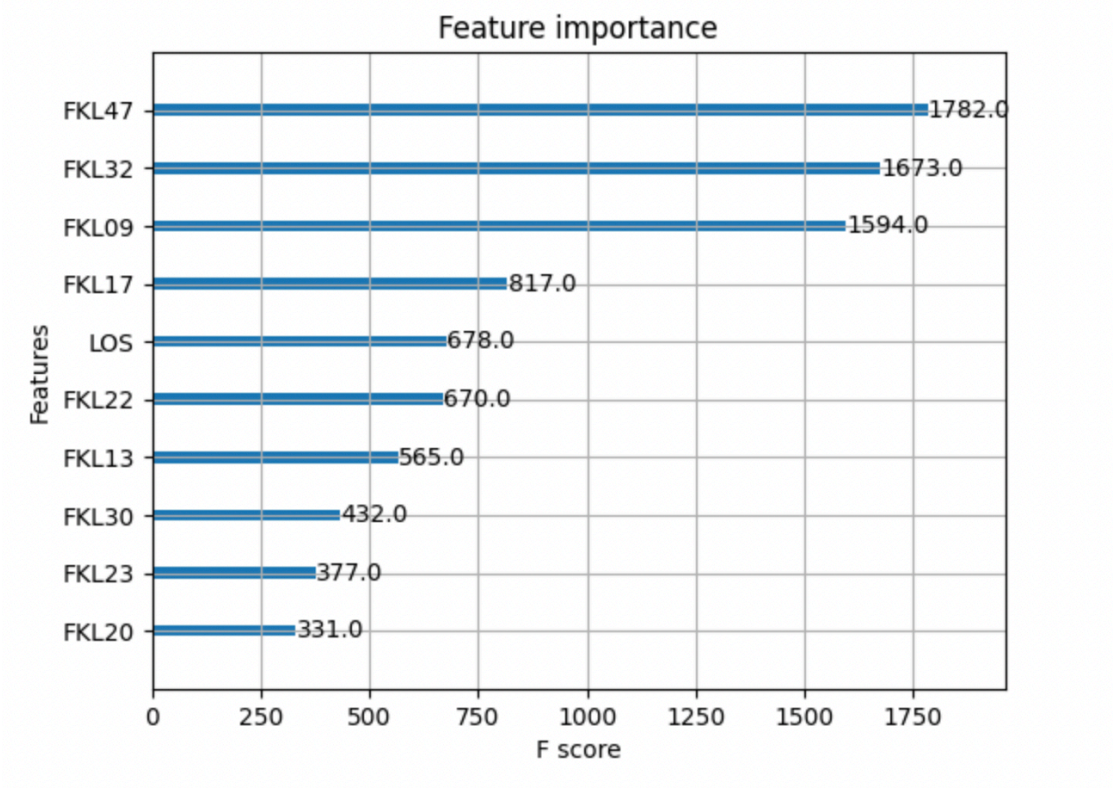
Parameter	Nilai
n estimators	100
Learning rate	0.1
Max depth	8

Kombinasi parameter pada Tabel 4.4 di atas menghasilkan nilai *Mean Abolute Error* (MAE) yang rendah, baik untuk *data training* maupun *data validation*. Hal tersebut menunjukkan bahwa model yang dihasilkan mampu meminimalisir kesalahan (*error*) pada kedua *subset* tersebut. Untuk memberikan gambaran lebih rinci terkait kinerja model, Gambar 4.5 dibawah ini menunjukkan *plot* atau visualisasi yang menunjukkan nilai MAE pada *data training* dan *data validation* untuk setiap *n_estimators*.



Gambar 4.5 *Plot Training dan Validation* MAE

Gambar 4.5 diatas menunjukkan *plot* garis untuk nilai MAE pada *data training* dan *data validation* seiring bertambahnya *n_estimators*. Terlihat bawah nilai MAE terus menurun seiring bertambahnya *n_estimators* dan nilainya rendah (dibawah 0.01) pada kedua *subset*. Selain itu, terlihat juga bahwa garis nilai MAE pada kedua *subset* data tersebut hampir sejajar. Hal tersebut menunjukkan bahwa model yang dihasilkan tidak *overfitting* maupun *underfitting*. Selain itu, dari proses pelatihan model didapatkan juga *feature importance*. *Feature importance* menunjukkan pengaruh masing-masing fitur terhadap nilai prediksi yang dihasilkan oleh model. Gambar 4.6 dibawah ini menunjukkan 10 fitur terbaik dalam *feature importance*



Gambar 4.6 *Feature Importance*

Pada Gambar 4.6 diatas menampilkan hasil visualisasi dari sepuluh fitur teratas dengan nilai *feature importance* tertinggi. *Feature importance* menunjukkan seberapa besar pengaruh suatu fitur terhadap nilai prediksi target dalam hal ini adalah FKL48 atau biaya verifikasi. Berdasarkan hasil visualisasi tersebut, fitur FKL47 memiliki pengaruh terbesar dengan nilai *feature importance* mencapai 1782.0, diikuti oleh FKL32 dengan nilai 1673.0, dan FKL09 sebesar 1594.0. Urutan berikutnya meliputi FKL17 sebesar 817.0, fitur LOS sebesar 678.0, FKL22 sebesar 670.0, FKL13 sebesar 565.0, FKL30 sebesar 432.0, FKL23 sebesar 377.0, dan FKL20 sebesar 331.0. Urutan tersebut menunjukkan bahwa fitur-fitur tersebut adalah yang paling signifikan dalam mempengaruhi prediksi biaya verifikasi.

4.5 *Testing Model*

Setelah mendapatkan model terbaik, langkah selanjutnya adalah pengujian model (*testing model*) menggunakan data uji (*data testing*) yang tidak pernah terlihat oleh model sebelumnya dalam tahap pelatihan dan validasi. Proses ini akan menjadi evaluasi akhir yang menilai kemampuan model dalam generalisasi data baru. Sama seperti *data*

training dan *data validation*, *data testing* juga melalui tahapan *preprocessing data* yang sama. Model terbaik tersebut kemudian diaplikasikan terhadap *data testing* dan menghasilkan prediksi biaya verifikasi. Perbandingan nilai aktual dan nilai hasil prediksi pada *data testing* terlihat seperti Tabel 4.5 dibawah ini:

Tabel 4.5 Hasil *Testing* Sebelum *Inverse Tranform*

Biaya Verifikasi Aktual	**Prediksi Biaya Verifikasi**
-0.191883	-0.19225574
-0.437602	-0.4372736
-0.427518	-0.42719582
-0.362185	-0.36095664
-0.42235	-0.42257154
0.028733	0.02928215
-0.0687	-0.06885646
0.244648	0.24832201
0.634921	0.6423973
-0.453518	-0.4528882
...	...
0.625357	0.62023234
-0.086913	-0.08570632
3.578114	3.6488037
-0.086913	-0.08570632
-0.409126	0.40892756
-0.443237	-0.44291544
-0.396745	-0.39420792
-0.444223	-0.44390136
0.608849	0.6064794

Tabel 4.5 diatas menampilkan hasil pengujian model (*testing model*) terhadap *data testing* yang telah mengalami proses normalisasi data sebelumnya menggunakan *StandardScaler*. Proses tersebut menyebabkan seluruh data berada pada rentang yang sama sehingga hasil prediksi yang dihasilkan juga berada pada skala normalisasi. Untuk

memberikan interpretasi yang lebih mudah untuk dipahami, maka penulis juga melakukan *inverse transform*.

Proses *inverse transform* ini bertujuan untuk mengembalikan nilai prediksi pada skala awal atau skala yang asli sebelum dilakukan normalisasi, sehingga nilai prediksi dapat dibandingkan secara langsung dengan nilai aktual dalam konteks skala nyata, seperti besaran biaya tagih dan biaya verifikasi dalam rupiah. Hasil prediksi pada *data testing* setelah dilakukan *inverse transform* terlihat seperti Tabel 4.6 dibawah ini:

Tabel 4.6 Hasil *Testing* Setelah *Inverse Transform*

Biaya Verifikasi Aktual	**Prediksi Biaya Verifikasi**
1591200	1589122.9
221800	223631.5
278000	279795.25
642100	648948.1
306800	305566.5
2820700	2823759.8
2277700	2276830.5
4024000	4044475
6199000	6240667.5
133100	136611
...	...
2176200	2182925.8
22601500	22995.454
2176200	2182.925.8
380500	381604.75
190400	192189.5
449500	463637.75
184900	18669475
6053700	6.040496
786200	777956.9

4.6 Evaluasi

Dalam melakukan evaluasi terhadap kinerja model yang telah dibangun, penulis menggunakan beberapa metrik evaluasi seperti *Mean Squared Error* (MSE), *Root Mean Squared Error* (RMSE), dan *Mean Absolute Error* (MAE). Metrik-metrik tersebut dipilih karena mampu memberikan gambaran yang baik untuk menilai perbedaan antara nilai aktual dan nilai prediksi yang dihasilkan oleh model. Untuk menghitung tingkat kesalahan (*error*) antara data aktual dan prediksi, penulis menggunakan *data testing* yang telah dinormalisasi tanpa penerapan proses *inverse transform*. Nilai *error* pada *data testing* terlihat seperti Tabel 4.7 dibawah ini:

Tabel 4.7 Nilai *Error Data Testing*

Biaya Verifikasi Aktual	**Prediksi Biaya Verifikasi**	$\|y_i - \hat{y}_i\|$
-0.191883	-0.19225574	0.00037274
-0.437602	-0.4372736	-0.0003284
-0.427518	-0.42719582	-0.00032218
-0.362185	-0.36095664	-0.00122836
-0.42235	-0.42257154	0.00022154
0.028733	0.02928215	-0.00054915
-0.0687	-0.06885646	0.00015646
0.244648	0.24832201	-0.00367401
0.634921	0.6423973	-0.0074763
-0.453518	-0.4528882	-0.0006298
...	...	...
0.625357	0.62023234	0.00512466
-0.086913	-0.08570632	-0.00120668
3.578114	3.6488037	-0.0706897
-0.086913	-0.08570632	-0.00120668
-0.409126	0.40892756	-0.81805356
-0.443237	-0.44291544	-0.00032156
-0.396745	-0.39420792	-0.00253708
-0.444223	-0.44390136	-0.00032164
0.608849	0.6064794	0.0023696

Berdasarkan hasil perhitungan *error* atau evaluasi pada Tabel 4.7, maka dapat dilakukan perhitungan *Mean Squared Error* (MSE), *Root Mean Squared Error* (RMSE), dan *Mean Absolute Error* (MAE). *Mean Squared Error* (MSE) dihitung menggunakan Persamaan 2.10 sebagai dibawah ini:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$MSE = \frac{\left|\begin{array}{c}(0.0003)^2 + (-0.0003)^2 + (-0.0003)^2 + (-0.001)^2 + \\ (0.0002)^2 + (-0.0005)^2 + (0.0001)^2 + (-0.0036)^2 + \\ (-0.007)^2 + (-0.0006)^2 + \ldots + \\ (0.005)^2 + (-0.001)^2 + (-0.07)^2 + (-0.001)^2 \\ + (-0.818)^2 + (-0.0003)^2 + (-0.002)^2 \\ + (-0.0003)^2 + (0.002)^2 + (0.0014)^2\end{array}\right|}{25135}$$

MSE = 0.01547383221952624= 0.015

Setelah mendapatkan nilai *Mean Squared Error* (MSE) sebesar 0.015, langkah selanjutnya adalah menghitung *Root Mean Squared Error* (RMSE). RMSE memberikan ukuran seberapa jauh prediksi model dari nilai aktual dalam satuan yang sama dengan data aslinya. Untuk menghitung RMSE, digunakan persamaan 2.11 sebagai berikut:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

$$RMSE = \sqrt{MSE}$$

$$RMSE = \sqrt{0.01547383221952624}$$

RMSE = 0.12439385925167785 = 0.12

Setelah mendapatkan nilai *Root Mean Squared Error (*RMSE) sebesar 0,12. Langkah berikutnya adalah menghitung *Mean Absolute Error* (MAE) yang akan

digunakan untuk mengukur rata-rata kesalahan absolut antara nilai prediksi dan nilai aktual. Berikut ini adalah Persamaan 2.12 untuk menghitung MAE:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]$$

$$MAE = \frac{\left|\begin{array}{c}(0.0003)^2 + (-0.0003)^2 + (-0.0003)^2 + (-0.001)^2 + \\ (0.0002)^2 + (-0.0005)^2 + (0.0001)^2 + (-0.0036)^2 \\ + (-0.007)^2 + (-0.0006)^2 + \ldots + \\ (0.005)^2 + (-0.001)^2 + (-0.07)^2 + (-0.001)^2 + \\ (-0.818)^2 + (-0.0003)^2 + (-0.002)^2 \\ + (-0.0003)^2 + (0.002)^2 + (0.0014)^2\end{array}\right|}{25135}$$

$$MAE = 0.008864111109520307 = 0.008$$

Dari hasil perhitungan diatas, didapatkan nilai MSE senilai 0.015, nilai RMSE senilai 0.12 dan nilai MAE senilai 0.008. Nilai-nilai tersebut menunjukkan kesalahan prediksi yang relatif kecil. *Mean Squared Error* (MSE) mengukur rata-rata selisih dari kuadrat antara nilai prediksi dan nilai aktual. Semakin kecil nilainya, maka semakin baik kinerja model dalam memprediksi data. Dalam hal ini, nilai MSE yang rendah menunjukkan bahwa prediksi yang dihasilkan model cukup akurat dengan selisih yang tidak terlalu besar dengan nilai sebenarnya (nilai aktual).

Selanjutnya adalah *Root Mean Squared Error* (RMSE) sebesar 0.12 yang memberikan gambaran lebih jelas mengenai tingkat kesalahan (*error*) prediksi dalam satuan asli data. RMSE digunakan karena memberikan interpretasi yang lebih mudah untuk dipahami. Nilai RMSE yang lebih kecil menunjukkan bahwa model dapat memprediksi biaya verifikasi dengan lebih akurat.

Terakhir, nilai *Mean Absolute Error* (MAE) sebesar 0,008 menunjuukan bahwa rata-rata kesalahan *absolute* pada prediksi sekitar 0,8%. Nilai MAE yang rendah tersebut menunjukkan bahwa rata-rata antara prediksi dan nilai aktual cukup kecil. Hal tersebut menandakan bahwa model dapat memprediksi biaya verifiksi dengan tingkat akurasi atau ketepatan yang baik. Nilai MAE memberikan gambaran tentang seberapa jauh nilai prediksi yang dihasilkan oleh model dengan nilai sebenarnya tanpa memperhitungkan arah kesalahan.

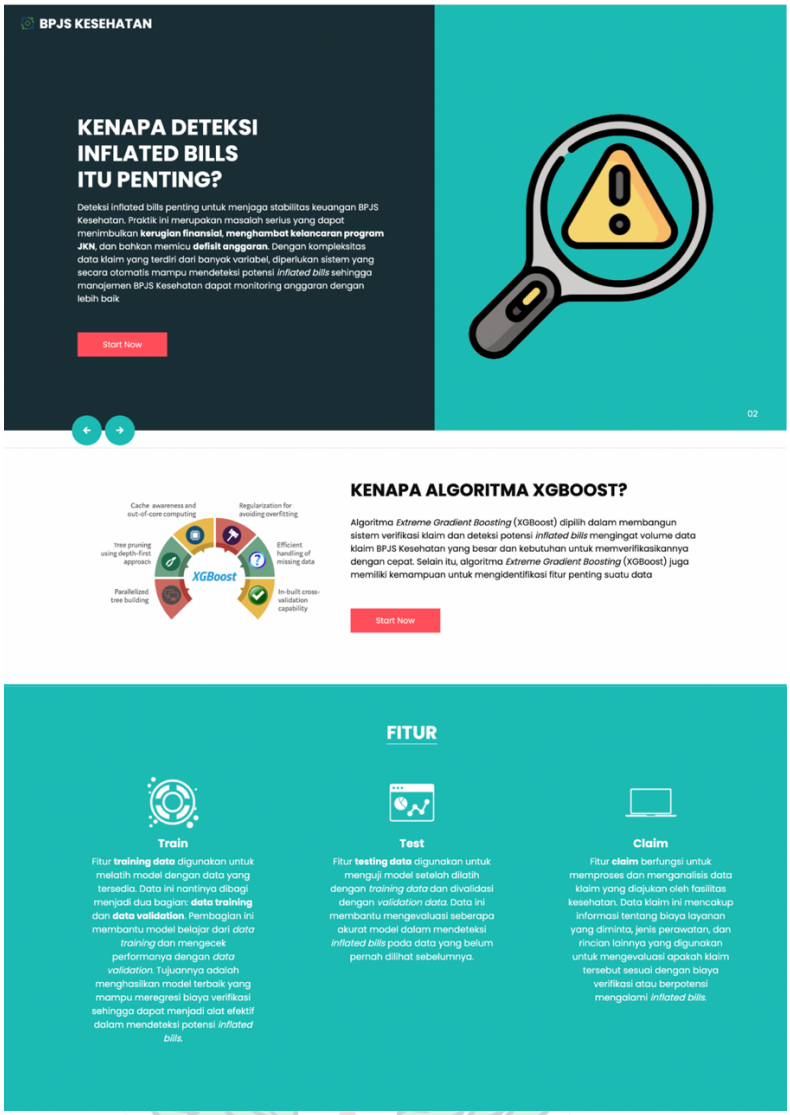
Secara keseluruhan, nilai MSE, RMSE, dan MAE yang relatif kecil menunjukkan model yang dibangun memiliki kinerja yang baik dalam memprediksi biaya verifikasi klaim layanan BPJS Kesehatan menggunakan algoritma *Extreme Gradient Boosting* (XGBoost). Dengan adanya kemampuan ini, maka model berpotensi digunakan sebagai alat dalam mendeteksi potensi *inflated bills* sehingga menghasilkan proses verifikasi yang cepat dan efisien.

4.7 Implementasi Perancangan Antarmuka

Adapun proses implementasi perancangan antarmuka yang telah dijelaskan pada Bab 2 sebelumnya menjadi sebuah *website* menggunakan Flask, HTML, dan CSS. Berikut ini merupakan hasil dari implementasi rancangan antarmuka antara lain:

1. Tampilan *Landing Page*

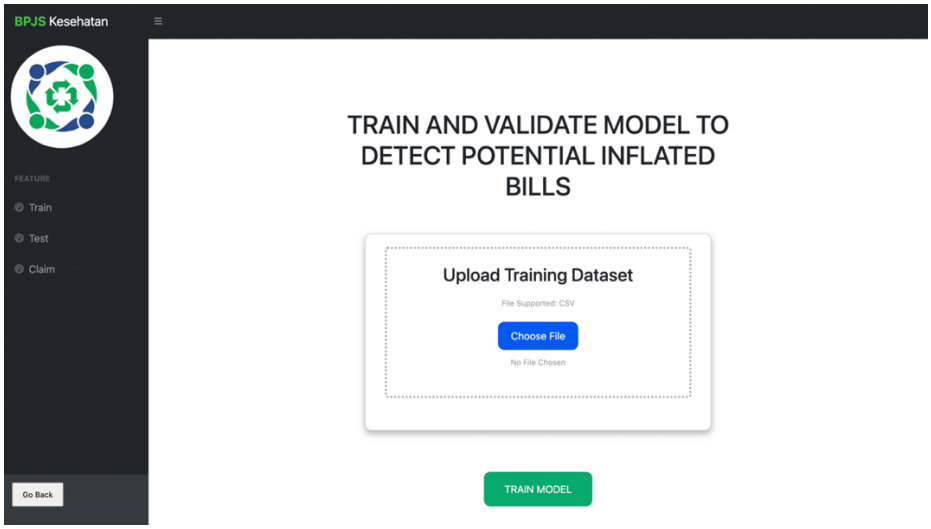
   Landing page merupakan halaman yang pertama kali dilihat oleh ketika mengakses sebuah *website.* Tampilan *landing page* dapat dilihat pada Gambar 4.7 dibawah ini:



Gambar 4.7 Tampilan *Landing Page*

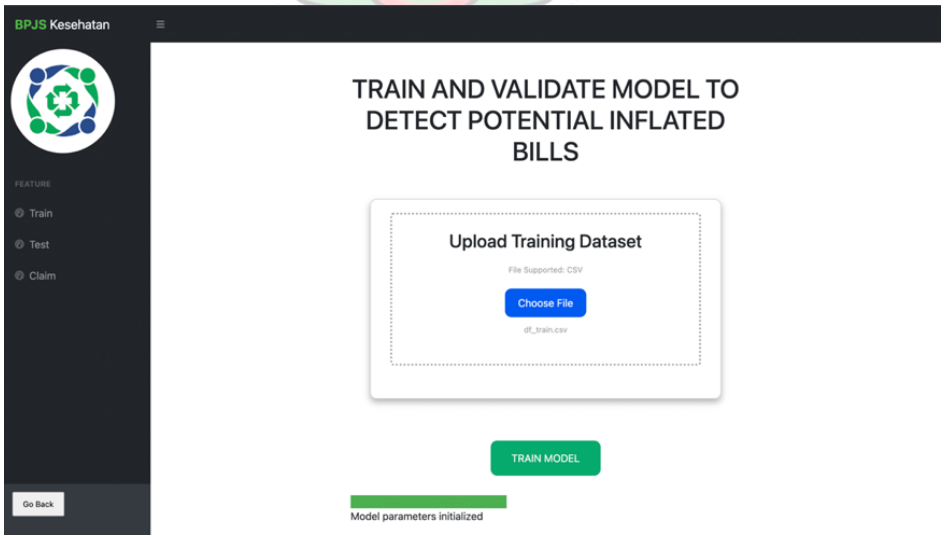
2. Tampilan Halaman *Training*

   Halaman *training* merupakan halaman yang digunakan oleh pengguna untuk melatih model dalam mendeteksi potensi inflated bills pada data klaim layanan BPJS Kesehatan menggunakan algoritma *Extreme Gradient Boosting* (XGBoost). Pengguna dapat mengakses halaman ini setelah menekan tombol "*Start Now*" pada *landing page*. Tampilan halaman *training* dapat dilihat seperti pada Gambar 4.8 dibawah ini:



Gambar 4.8 Tampilan Halaman *Training*

Pada halaman *training*, terdapat sebuah *form input* yang dapat digunakan oleh pengguna untuk mengunggah *dataset* yang digunakan untuk melatih model. Setelah *dataset* diunggah, terdapat tombol "*TRAIN MODEL*" yang berfungsi untuk memulai proses pelatihan model tersebut. Untuk mempermudah pengguna dalam berpindah halaman, sistem ini juga dilengkapi dengan *sidebar* disebelah kiri dengan opsi seperti *training, testing, claim,* dan *back*. Setelah pengguna berhasil mengunggah dan menekan tombol "*TRAIN MODEL*", sistem akan menampilkan *progress bar* yang menunjukkan status dari proses pelatihan model. Tampilan *progress bar* pada halaman *training* ditunjukkan pada Gambar 4.9 dibawah ini:



Gambar 4.9 Tampilan *Progress Bar*

Setelah proses pelatihan atau *training* selesai, sistem beralih ke halaman selanjutnya yang menampilkan hasil dari proses *training* tersebut. Tampilan halaman hasil *training* terlihat seperti pada Gambar 4.10 dibawah ini:

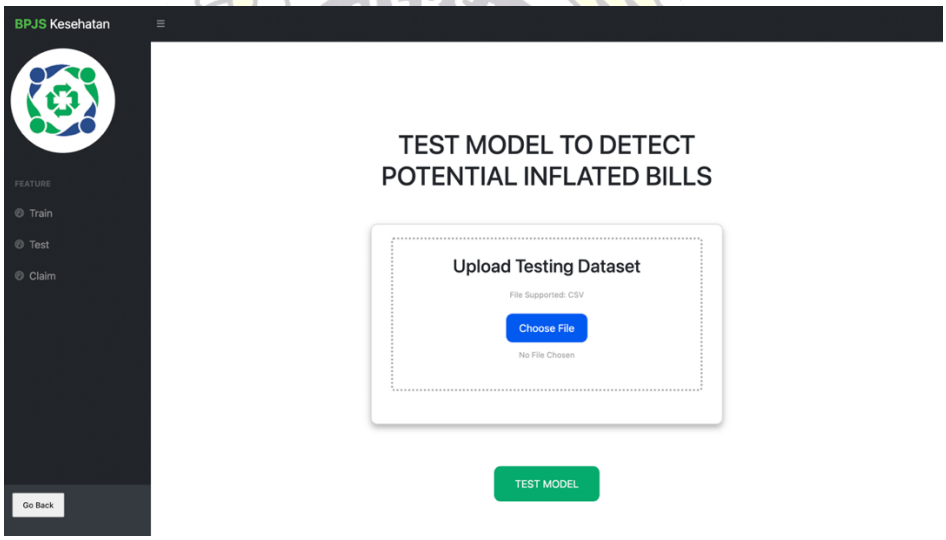
Gambar 4.10 Tampilan Halaman Hasil *Training*

Gambar 4.10 diatas menunjukkan sebuah tabel berisi *training* MAE dan *validation MAE* yang pada setiap iterasi parameter *n_estimators*. Dibawah tabel, terdapat grafil yang memvisualisasikan perkembangan *Train MAE* dan

Validation MAE beserta nilai akhir MAE pada *data training* dan *data validation*. Sementara itu, disebelah kanan terdapat visualisasi *feature importance* yang menunjukkan kontribusi setiap fitur dalam model.

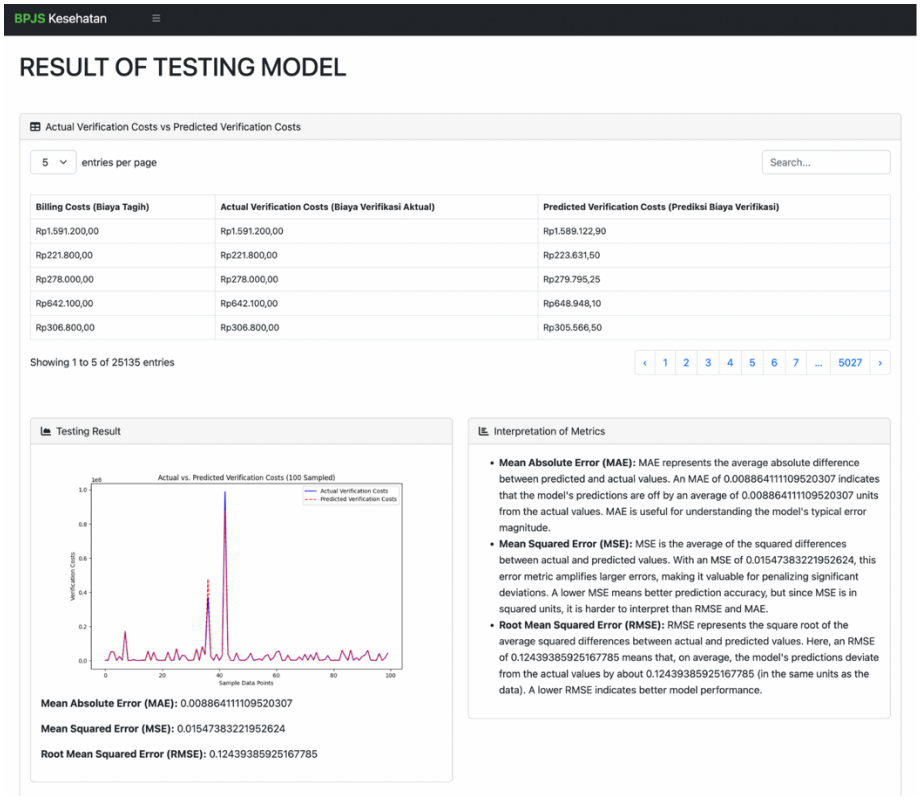
3. Tampilan Halaman *Testing*

Halaman *testing* merupakan hlaman yang dapat digunakan oleh pengguna dalam menilai kinerja model dengan menggunakan data yang tidak pernah dilihat oleh *data training* sebelumnya. Pada halaman ini, pengguna dapat menggunggah *file* dengan format .csv untuk data testing. Setelah *file* tersebut berhasil diunggah, pengguna dapat menekan tombol "*Test Model*" untuk memulai proses pengujian. Selama proses ini, terdapat *progress bar* yang menampilkan status dari pengujian model. Tampilan halaman *testing* terlihat seperti Gambar 4.12 dibawah ini:



Gambar 4.11 Tampilan Halaman *Testing*

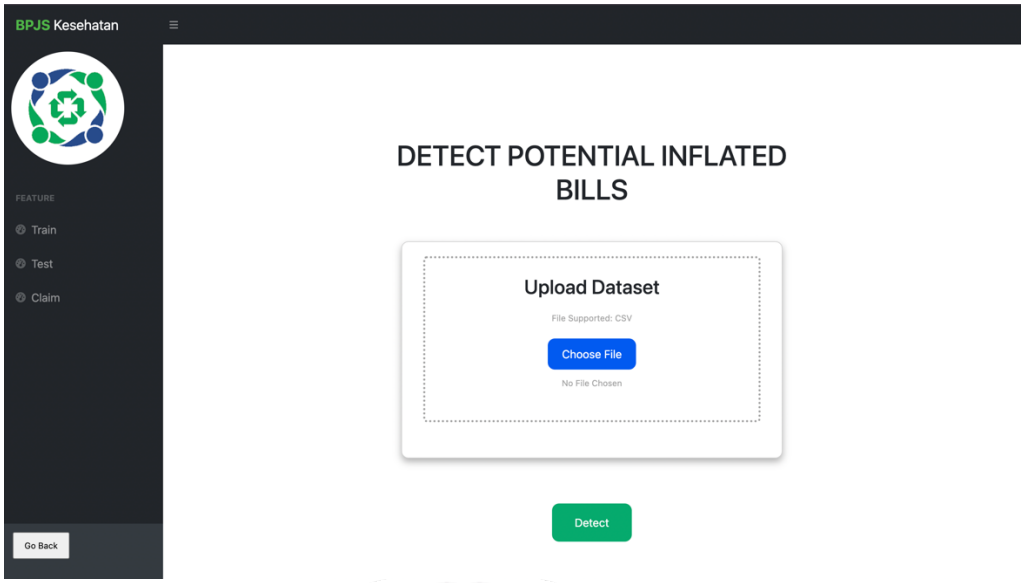
Setelah seluruh proses pengujian selesai, sistem beralih ke halaman hasil *testing* yang berisi tabel dengan nilai aktual dari biaya tagih dan biaya verifikasi serta nilai prediksi dari biaya verifikasi. Dengan adanya tabel tersebut, pengguna dapat melihat dan membandingkan hasil prediksi oleh model dengan nilai sebenarnya yang terdapat dalam data. Selain itu, terdapat juga informasi mengenai metrik evaluasi yang digunakan seperti MSE, RMSE, dan MAE. Tampilan halaman hasil *testing* dapat dilihat pada Gambar 4.13 dibawah ini:



Gambar 4.12 Tampilan Halaman Hasil *Testing*

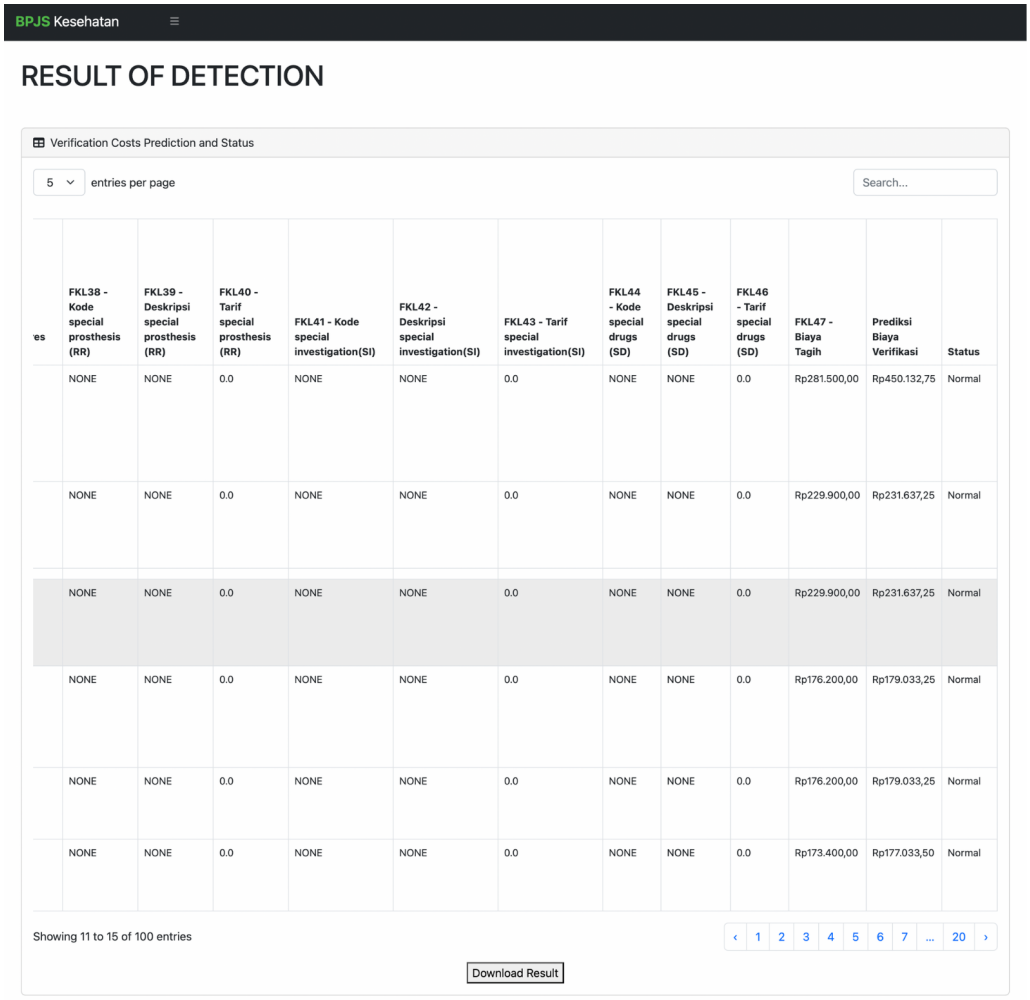
4. Tampilan Halaman *Claim*

   Halaman *claim* merupakan halaman yang dapat digunakan oleh pengguna dalam memverifikasi klaim dan mendeteksi potensi *inflated bills*. Pada halaman ini, pengguna dapat mengunggah *file* dengan format CSV yang berisi klaim-klaim yang diajukan oleh fasilitas kesehatan dan perlu untuk diverifikasi. Setelah kumpulan klaim tersebut diunggah, pengguna dapat menekan tombol " *Detect*" untuk memulai proses verifikasi dan deteksi. Selama proses tersebut, terdapat *progress bar* yang menunjukkan kemajuan dan status dari proses verifikasi dan deteksi *inflated bills*. Di bawah ini adalah Gambar 4.14 yang menampilkan tampilan halaman *claim*:



Gambar 4.13 Tampilan Halaman *Claim*

Setelah proses verifikasi dan deteksi selesai, sistem akan secara otomatis beralih ke halaman hasil yang menyajikan informasi lengkap mengenai klaim yang diajukan oleh fasilitas kesehatan. Halaman ini mencakup detail penting seperti biaya tagih yang diajukan, prediksi biaya verifikasi yang dihasilkan oleh model, serta status klaim tersebut. Jika biaya tagih yang diajukan oleh fasilitas kesehatan melebihi prediksi biaya verifikasi, sistem akan menandai klaim tersebut sebagai berpotensi mengalami *inflated bills*. Sebaliknya, jika biaya tagih tersebut sesuai dengan prediksi atau lebih kecil, klaim akan dianggap normal. Hal ini memberikan gambaran yang jelas kepada verifikator mengenai status setiap klaim yang diajukan. Selain itu, pengguna juga memiliki opsi untuk mengunduh hasil deteksi dalam bentuk file melalui tombol "*Download Results*", yang memudahkan dalam menyimpan dan merujuk kembali ke hasil analisis kapan saja diperlukan. Gambar 4.15 berikut menunjukkan tampilan halaman hasil deteksi:



Gambar 4.14 Tampilan Halaman Hasil Deteksi

.

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian terkait deteksi potensi *inflated bills* pada data klaim layanan BPJS Kesehatan yang telah dilakukan sebelumnya menghasilkan kesimpulan sebagai berikut:

1. Proses *parameter tuning* dilakukan sebanyak 48 kali dengan menguji kombinasi nilai pada tiga parameter utama yaitu *n_estimators* (50, 100, 200, dan 300) *learning_rate* (0.01, 0.05, dan 0.1), dan *max_depth* (2, 4, 6, dan 8).
2. Hasil dari parameter tuning menunjukkan bahwa model dengan kombinasi terbaik dimiliki oleh *n estimator* sebesar 100, *learning rate* sebesar 0.1, dan *max depth* sebesar 8. Model tersebut menghasilkan nilai *Mean Absolute Error* (MAE) sebesar 0.005 pada *data training* dan 0.008 pada *data validation*.
3. Hasil pengujian pada penelitian ini menunjukkan bahwa model memiliki nilai MAE sebesar 0.008, MSE sebesar 0.015, dan RMSE sebesar 0.12 yang diuji pada *data testing*. Nilai-nilai tersebut menunjukkan bahwa model memiliki tingkat kesalahan yang kecil dalam memprediksi biaya verifikasi dan dapat digunakan sebagai alat untuk mendeteksi potensi *inflated bills.*

5.2 Saran

Berikut saran-saran yang dapat penulis berikan untuk pengembangan penelitian di masa mendatang, antara lain:

1. Memperluas cakupan data dengan menggunakan lebih banyak sampel data terbaru untuk meningkatkan representasi yang sesuai dengan kebutuhan sistem.
2. Melibatkan lebih banyak variabel dalam penelitian selanjutnya untuk meningkatkan wawasan sistem yang dibangun dalam mendeteksi potensi *inflated bills*.

3. Melakukan *parameter tuning* pada parameter-parameter lain untuk mengoptimalisasi kinerja model.

DAFTAR PUSTAKA

Adams, F., Agsar, R., Anggoro, D., Satria, M. B., Oktavia, A. W., & Chamidah, N. (2021). Perbandingan Normalisasi Data Untuk Klasifikasi Wine Menggunakan Algoritma Naïve Bayes, Decision Tree, Dan Support Vector Machine. In *Seminar Nasional Mahasiswa Ilmu Komputer Dan Aplikasinya (Senamika) Jakarta-Indonesia*.

Adiwiguna, Y. P., Purbaseno, H., Syuhara, H., Pribadi, F. J., Sarbani, A., & Kusuma, A. D. (2022). Membangun Tool Prediksi Yang Komprehensif Guna Meningkatkan Akurasi Produk Akhir Pencampuran Kualitas Batubara Menerapkan Perbandingan Kolaboratif Pada Algoritma Regresi Hingga Neural Net. *Indonesian Mining Professionals Journal*, *4*(2), 111–122.

Afrina, E., Cut, D., Aidha, N., Ramdlaningrum, H., Wahyudi, D., Silvia, K., Fanggidae, J., Dwi, H., Ningrum, R., Thaariq, R. M., Kartika, W., & Chrisnahutama, A. (2020). *Defisit Jaminan Kesehatan Nasional (Jkn): Mengapa Dan Bagaimana Mengatasinya?*

Alghifari, F., & Juardi, D. (2021). Penerapan Data Mining Pada Penerapan Data Mining Pada Penjualan Makanan Dan Minuman Menggunakan Metode Algoritma Naïve Bayes. *Jurnal Ilmiah Informatika*, *9*(02), 75–81. Https://Doi.Org/Https://Doi.Org/10.33884/Jif.V9i02.3755

Almaliki, M. F., Ningrum, I. P., & Saputra, R. A. (2023). Implementasi Metode Mesin Rekomendasi User Based Filtering Pada Sistem Penyewaan Alat Pertambangan. *Jurnal Manajemen Informatika(Jamika)*, *3*(1), 40–51.

Annisa, R., Winda, S., Dwisaputro, E., & Isnaini, K. N. (2021). Mengatasi Defisit Dana Jaminan Sosial Kesehatan Melalui Perbaikan Tata Kelola. *Integritas: Jurnal Antikorupsi*, *6*(2), 209–224. Https://Doi.Org/Https://Doi.Org/10.32697/Integritas.V6i2.664

Browniee, J. (2020). *Data Preparation For Machine Learning: Data Cleaning, Feature Selection, And Data Transforms In Python*. Machine Learning Mastery.

Chen, T., & Guestrin, C. (2016). Xgboost: A Scalable Tree Boosting System. *Proceedings Of The Acm Sigkdd International Conference On Knowledge*

Discovery And Data Mining, *13-17-August-2016*, 785–794. Https://Doi.Org/10.1145/2939672.2939785

Dharma, A. S., & Tambunan, V. (2021). Penerapan Model Pembelajaran Dengan Metode Reinforcement Learning Menggunakan Simulator Carla. *Jurnal Media Informatika Budidarma*, *5*(4), 1405. Https://Doi.Org/10.30865/Mib.V5i4.3169

Diba, A. F., & Oka Warmana, G. (2022). Sosialisasi Bpjs Kesehatan Pada Pegawai Security Untuk Mengetahui Pentingnya Jaminan Kesehatan Di Ciputra Surabaya. *Jurnal Pengabdian Kepada Masyarakat*, *2*(3), 198–202. Https://Jurnalfkip.Samawa-University.Ac.Id/Karya_Jpm/Index

Herdian, C., Kamila, A., & Agung Musa Budidarma, I. G. (2024). Studi Kasus Feature Engineering Untuk Data Teks: Perbandingan Label Encoding Dan One-Hot Encoding Pada Metode Linear Regresi. *Technologia : Jurnal Ilmiah*, *15*(1), 93. Https://Doi.Org/10.31602/Tji.V15i1.13457

Indrawan, G., Lemes, I. N., & Surata, I. N. (2024). Peranan Dinas Kesehatan Dalam Pencegahan Dan Penanganan Kecurangan (Fraud) Dalam Pelaksanaan Program Jaminan Kesehatan Berdasarkan Peraturan Menteri Kesehatan Republik Indonesia Nomor 16 Tahun 2019 Di Kabupaten Buleleng. *Kertha Widya*, *11*(2), 59–84.

Joel, L. O., Doorsamy, W., & Sena Paul, B. (2022). A Review Of Missing Data Handling Techniques For Machine Learning. *International Journal Of Innovative Technology And Interdisciplinary Sciences Www.Ijitis.Org*, *5*(3), 971–1005. Https://Doi.Org/10.15157/Ijitis.2022.5.3.971-1005

Kurnia, D., Mazdadi, M. I., Kartini, D., Nugroho, R. A., Abadi, F., & Korespondensi, P. (2023). Seleksi Fitur Dengan Particle Swarm Optimization Pada Klasifikasi Penyakit Parkinson Menggunakan Xgboost. *Jurnal Teknologi Informasi Dan Ilmu Komputer (Jtiik)*, *10*(5), 1083–1094. Https://Doi.Org/10.25126/Jtiik.2023107252

Kurniawan, D. (2022). *Pengenalan Machine Learning Dengan Python*. Elex Media Komputindo.

Kusnaidi, M. R., Gulo, T., & Aripin, S. (2022). Penerapan Normalisasi Data Dalam Mengelompokkan Data Mahasiswa Dengan Menggunakan Metode K-Means Untuk Menentukan Prioritas Bantuan Uang Kuliah Tunggal. *Journal Of Computer System And Informatics (Josyc)*, *3*(4), 330–338. Https://Doi.Org/10.47065/Josyc.V3i4.2112

Maretva Cendani, L., & Wibowo, A. (2022). *Perbandingan Metode Ensemble Learning Pada Klasifikasi Penyakit Diabetes* (Vol. 13, Issue 1).

Mauren Michaela, S., Nurmalasari, M., & Hosizah, H. (2021). Fraud In Healthcare Facilities: A Narrative Review. *Public Health Of Indonesia*, *7*(4), 166–171. Https://Doi.Org/10.36685/Phi.V7i4.465

Mienye, I. D., & Sun, Y. (2022). A Survey Of Ensemble Learning: Concepts, Algorithms, Applications, And Prospects. In *Ieee Access* (Vol. 10, Pp. 99129–99149). Institute Of Electrical And Electronics Engineers Inc. Https://Doi.Org/10.1109/Access.2022.3207287

Mitigasi Defisit Jkn. (2024). Kompas. Https://Www.Kompas.Id/Baca/Opini/2024/01/12/Mitigasi-Defisit-Jkn

Nabrawi, E., & Alanazi, A. (2023). Fraud Detection In Healthcare Insurance Claims Using Machine Learning. *Risks*, *11*(9). Https://Doi.Org/10.3390/Risks11090160

Ninia, K. V., Sholihah, B., & Rochman, A. (2023). Prediction Of Inefficiency In Health Insurance Administration Institutions In Indonesia Using Light Gradient Boosting Machine. *Intelmatics*, *3*(1), 17–22.

Nugraha, A. C., & Irawan, M. I. (2023). Komparasi Deteksi Kecurangan Pada Data Klaim Asuransi Pelayanan Kesehatan Menggunakan Metode Support Vector Machine (Svm) Dan Extreme Gradient Boosting (Xgboost). *Jurnal Sains Dan Seni Its*, *12*(1), A40–A46. Https://Doi.Org/Http://Dx.Doi.Org/10.12962/J23373520.V12i1.107032

Ozdemir, S., & Susarla, D. (2018). *Feature Engineering Made Easy: Identify Unique Features From Your Dataset In Order To Build Powerful Machine Learning Systems*. Packt Publishing Ltd.

Parthasarathy, S., Lakshminarayanan, A. R., Khan, A. A. A., Sathick, K. J., & Jayaraman, V. (2023). Detection Of Health Insurance Fraud Using Bayesian Optimized Xgboost. *International Journal Of Safety And Security Engineering*, *13*(5), 853–861. Https://Doi.Org/10.18280/Ijsse.130509

Peraturan Menteri Kesehatan Nomor 16 Tahun 2019 Tentang Pencegahan Dan Penanganan Kecurangan (Fraud) Serta Pengenaan Sanksi Administrasi Terhadap Kecurangan (Fraud) Dalam Pelaksanaan Program Jaminan Kesehatan, Pub. L. No. 16 (2019).

Permana, I., & Salisah, F. N. (2022). The Effect Of Data Normalization On The Performance Of The Classification Results Of The Backpropagation Algorithm. *Indonesian Journal Of Informatic Research And Software Engineering*, *2*(1), 67–72.

Prasetya, M. R. A., Priyatno, A. M., & Nurhaeni. (2023). Penanganan Imputasi Missing Values Pada Data Time Series Dengan Menggunakan Metode Data Mining. *Jurnal Informasi Dan Teknologi*, *5*(2), 52–62.

Retnoningsih, E., & Pramudita, R. (2020). Mengenal Machine Learning Dengan Teknik Supervised Dan Unsupervised Learning Menggunakan Python. *Bina Insani Ict Journal*, *7*(2), 156–165.

Rofiq, H. N. (2023). Deteksi Inefisiensi Pada Klaim Bpjs Kesehatan Dengan Menggunakan Machine Learning. *Jurnal Jaminan Kesehatan Nasional*, *3*(1), 83–98. Https://Doi.Org/10.53756/Jjkn.V3i1.134

Rukhsar, L., Haider Bangyal, W., Nisar, K., & Nisar, S. (2022). Prediction Of Insurance Fraud Detection Using Machine Learning Algorithms. *Mehran University Research Journal Of Engineering And Technology*, *41*(1), 33–40. Https://Doi.Org/10.22581/Muet1982.2201.04

Ruswanti, D. (2020). Pengukuran Performa Support Vector Machine Dan Neural Netwok Dalam Meramalkan Tingkat Curah Hujan. *Jurnal Gaung Informatika*, *3*(1).

Santoso, P., Abijono, H., & Anggreini, N. L. (2021). Algoritma Supervised Learning Dan Unsupervised Learning Dalam Pengolahan Data. *Unira Malang* |, *4*(2).

Septiana Rizky, P., Haiban Hirzi, R., Hidayaturrohman, U., Hamzanwadi Selong Jl Tgkh Muhammad Zainuddin Abdul Madjid Pancor, U., & Timur, L. (2022). Perbandingan Metode Lightgbm Dan Xgboost Dalam Menangani Data Dengan Kelas Tidak Seimbang. In *J Statistika* (Vol. 15, Issue 2). Www.Unipasby.Ac.Id

Shahani, N. M., Zheng, X., Liu, C., Hasan, F. U., & Li, P. (2021). Developing An Xgboost Regression Model For Predicting Young's Modulus Of Intact Sedimentary Rocks For The Stability Of Surface And Subsurface Structures. *Frontiers In Earth Science*, *9*.

Simanjuntak, H. T. A., Sigiro, M., & Simanungkalit, H. M. (2021). Deteksi Fraud Pada Klaim Layanan Rumah Sakit Menggunakan Model Neural Network. *Journal Of Applied Technology And Informatics Indonesia*, *1*(1).

Singh, D., & Singh, B. (2020). Investigating The Impact Of Data Normalization On Classification Performance. *Applied Soft Computing*.

Singh, K. (2024). Healthcare Fraudulence: Leveraging Advanced Artificial Intelligence Techniques For Detection. *International Research Journal Of Modernization In Engineering Technology And Science*. Https://Doi.Org/10.56726/Irjmets49394

Siringoringo, R., Perangin Angin, R., & Rumahorbo, B. (2022). Model Klasifikasi Genetic-Xgboost Dengan T-Distributed Stochastic Neighbor Embedding Pada Peramalan Pasar. *Jurnal Times*, *Xi*(1), 30–36. Https://Archive.Ics.Uci.Edu/Ml/Datasets/Online+Retail

Sitorus, C. M., Rizal, A., & Jajuli, M. (2020). Prediksi Risiko Perjalanan Transportasi Online Dari Data Telematik Menggunakan Algoritma Support Vector Machine. *Jurnal Teknik Informatika Dan Sistem Informasi*, *6*(2). Https://Doi.Org/10.28932/Jutisi.V6i2.2672

Soebroto, A. A. (2019). *Buku Ajar Ai, Machine Learning & Deep Learning*. Https://Www.Researchgate.Net/Publication/348003841

Syukron, M., Santoso, R., & Widiharih, T. (2020). Perbandingan Metode Smote Random Forest Dan Smote Xgboost Untuk Klasifikasi Tingkat Penyakit Hepatitis C Pada Imbalance Class Data. *Jurnal Gaussian*, *9*(3), 227–236. Https://Doi.Org/Https://Doi.Org/10.14710/J.Gauss.9.3.227-236

Tanha, J., Abdi, Y., Samadi, N., Razzaghi, N., & Asadpour, M. (2020). Boosting Methods For Multi-Class Imbalanced Data Classification: An Experimental Review. *Journal Of Big Data*, *7*(1). Https://Doi.Org/10.1186/S40537-020-00349-Y

Vermaysha, A., & Nurmalitasari, N. (2023). Prediksi Harga Rumah Di Kabupaten Karanganyar Menggunakan Metode Regresi Linear. *In Prosiding Seminar Nasional Teknologi Informasi Dan Bisnis*, 6–11.

Wang, W., Chakraborty, G., & Chakraborty, B. (2021). Predicting The Risk Of Chronic Kidney Disease (Ckd) Using Machine Learning Algorithm. *Applied Sciences (Switzerland)*, *11*(1), 1–17. Https://Doi.Org/10.3390/App11010202

Yulianti, S. E. H., Soesanto, O., & Sukmawaty, Y. (2022). Penerapan Metode Extreme Gradient Boosting (Xgboost) Pada Klasifikasi Nasabah Kartu Kredit. *Journal Of Mathematics: Theory And Applications*, 21–26.

Zhao, Y. F., Xie, J., & Sun, L. (2024). On The Data Quality And Imbalance In Machine Learning-Based Design And Manufacturing—A Systematic Review. *Engineering*. Https://Doi.Org/10.1016/J.Eng.2024.04.024