DOI: 10.22146/jnteti.Nomor DOI

BPJS Visit Type Classification for Prediction Healthy-Sick using XGBoost Approach

Naomi Elena Lumbanraja[1], Karina Checilia Situmorang[2], Rebecca Yulyartha Bulawan Sihombing[3], Tio Manalu[4]

Github : NaomiLumbanraja/BPPJS-Hackaton-Visit-Type-Prediction-Healthy-Sick

INTISARI — Penelitian ini bertujuan mengimplementasikan algoritma XGBoost untuk mengklasifikasikan jenis kunjungan peserta BPJS Kesehatan di Fasilitas Kesehatan Tingkat Pertama (FKTP) menjadi kategori sehat atau sakit. Dataset yang digunakan mencakup data kepesertaan, diagnosis, kunjungan, dan klaim BPJS periode 2015–2021. Proses dilakukan dengan pendekatan CRISP-DM, mencakup pemahaman bisnis, eksplorasi data, persiapan data, pembangunan model, evaluasi, hingga deployment. Hasil evaluasi menunjukkan akurasi sebesar 64,98%, dengan performa baik pada kelas mayoritas namun terbatas pada kelas minoritas. Model ini diharapkan mampu mendukung perencanaan sumber daya FKTP, meningkatkan efisiensi, dan memastikan layanan kesehatan yang optimal.

KATA KUNCI — *BPJS Kesehatan, FKTP, XGBoost, Prediksi Sehat-Sakit, Data Mining, Klasifikasi Kunjungan, CRISP-DM, Layanan Kesehatan.*

I. PENDAHULUAN

1.1 Latar Belakang

Kesehatan merupakan salah satu kebutuhan dasar bagi setiap individu yang berperan sangat dalam menunjang kualitas hidup masyarakyat. Setiap individu pastinya akan berusaha mencari pengobatan apabila merasa tidak sehat dengan cara menghubungi rumah sakit atau sarana Kesehatan lainnya. Namun, tidak semua orang dapat merasakan pelayanan kesehatan dengan baik karena tidak mampu dalam membayar biaya perobatan yang mahal [1]. Oleh karena itu, akses terhadap pelayanan kesehatan yang merata dan terjangkau menjadi suatu tujuan utama dalam pembangunan sosial-ekonomi di setiap negara.

Di Indonesia, prinsip dasar pembangunan kesehatan dirumuskan dalam Undang-Undang Dasar Tahun 1945, yaitu pasal 28, yang menyatakan bahwa kesehatan adalah hak fundamental setiap warga negara. Maka, untuk memastikan setiap warga negara mendapatkan pelayanan kesehatan yang layak dan terjangkau, dibentuk Badan Penyelenggaraan Jaminan Sosial Kesehatan (BPJS Kesehatan). BPJS Kesehatan merupakan lembaga yang didirikan pemerintah sebagai implementasi dari Jaminan Kesehatan Nasional (JKN) [2]. Program BPJS Kesehatan diatur dalam Undang-Undang No. 40 Tahun 2004 tentang Sistem Jaminan Sosial Nasional dan Undang-Undang No 24. Tahun 2011 tentang BPJS untuk memberikan jaminan sosial menyeluruh bagi setiap warga negara bertujuan untuk memenuhi kebutuhan dasar hidup yang layak menuju terwujudnya masyarakat Indonesia yang sejahtera, adil dan makmur [3][4].

Namun, BPJS Kesehatan masih memerlukan peningkatan dalam hal efisiensi dan keakuratan data untuk mengelola pelayanan, memonitor peserta, serta untuk mengidentifikasi dan memitigasi resiko kesehatan yang dapat mempengaruhi biaya layanan kesehatan. Selain itu, kompleksitas sistem dan tingginya jumlah peserta menuntut adanya solusi berbasis data untuk meningkatkan efisiensi dan efektivitas layanan kesehatan, khususnya di Fasilitas Kesehatan Tingkat Pertama (FKTP) [5]. FKTP, sebagai garda terdepan dalam pelayanan kesehatan, menangani berbagai jenis kunjungan peserta, baik untuk keperluan preventif (sehat) maupun untuk penangan penyakit (sakit). Namun, tanpa prediksi yang akurat mengenai jenis kunjungan, sering kali terjadi kelebihan atau kekurangan sumber daya, seperti tenaga medis, obat-obatan, dan fasilitas pendukung lainnya. Hal ini tidak hanya menurunkan efisiensi layanan, tetapi juga dapat mempengaruhi kualitas pelayanan kesehatan yang diterima oleh peserta [5].

Melalui pendekatan berbasis data, BPJS Kesehatan dapat mengatasi tantangan ini dengan memanfaatkan model klasifikasi untuk mengidentifikasi pola kunjungan peserta. Data

p-ISSN xxxx–xxxx | e-ISSN xxxx–xxxx

BPJS Kesehatan antara lain data kepesertaan dan data pelayanan kesehatan yang merekam aktivitas pemanfaatan JKN-KIS oleh peserta JKN di fasilitas Kesehatan yang telah bekerjasama dengan BPJS Kesehatan dan diklaim oleh fasilitas kesehatan kepada BPJS Kesehatan [6]. Dalam proyek ini, klasifikasi kunjungan tidak hanya dibagi menjadi kategori "Sehat" dan "Sakit" tetapi juga didasarkan pada status kunjungan peserta, seperti "Berobat Jalan", "Rujuk Lanjut", "Sembuh", "Pulang Paksa", "Kunjungan Sehat", dan lainnya. Informasi ini memungkinkan analisis lebih mendalam terhadap pola kunjungan peserta, membantu dalam perencanaan sumber daya, dan meningkatkan efisiensi operasional di FKTP.

Proyek ini bertujuan untuk mengimplementasikan pendekatan XGBoost dalam rangka meningkatkan sistem analisis dan prediksi pada BPJS Kesehatan. XGBoost (*Extreme Gradient Boosting*) telah menjadi salah satu algoritma yang paling efektif dalam memecahkan berbagai masalah prediksi dan klasifikasi [7]. XGBoost, yang merupakan teknik *ensemble learning* berbasis pohon keputusan, memiliki kemampuan untuk menangani data dalam jumlah besar dengan kompleksitas tinggi serta mampu memberikan prediksi yang sangat akurat.

Dengan menggunakan pendekatan ini, BPJS Kesehatan dapat mengoptimalkan alokasi sumber daya, seperti tenaga medis, obat-obatan, serta peralatan sesuai dengan prediksi kebutuhan nyata di lapangan. Hal ini akan mendukung peningkatan efisiensi dan efektivitas pelayanan kesehatan, sekaligus memastikan setiap peserta mendpaatkan layanan yang layak sesuai kebutuhan.

1.1 Tujuan

Adapun tujuan dari pelaksanaan proyek ini:

1. Memenuhi salah satu syarat dalam penyelesaian mata kuliah Data Mining T.A. 2024/2025
2. Memahami konsep, metode, dan aplikasi algoritma XGBoost secara mendalam untuk menyelesaikan masalah nyata, khususnya prediksi status jenis kunjungan peserta FKTP dalam konteks BPJS Kesehatan.
3. Mengembangkan dan mengimplementasikan model prediktif berbasis data yang dapat membantu dalam meningkatkan efisiensi dan efektivitas layanan kesehatan, dengan fokus pada analisis status kunjungan peserta, termasuk kategori kunjungan sakit dan kunjungan sehat, dengan tujuan untuk mendukung perencanaan sumber daya yang lebih optimal di FKTP.

1.2 Manfaat

Manfaat dari pengerjaan proyek ini yaitu:

1. Mahasiswa dapat memperoleh pengalaman langsung dalam membangun model prediktif menggunakan algoritma seperti XGBoost, yang merupakan keterampilan penting dalam pengembangan karir di bidang data science dan machine learning.
2. Memberikan gambaran nyata bagaimana data mining dapat memberikan solusi terhadap tantangan di dunia nyata, khususnya dalam sektor kesehatan yang kompleks.
3. Membantu BPJS Ksehatan dalam mengoptimalkan perencanaan seumber daya di FKTP melalui pengembangan model prediktif yang dapat meningkatkan efisiensi dan kualitas layanan kesehatan.

1.3 Ruang Lingkup

Ruang lingkup dalam proyek yang dikerjakan yaitu:

1. Dataset yang digunakan yaitu data sampe BPJS Kesehatan tahun 2015 sampai 2021 yang terdiri dari 4 jenis data yaitu:
   a. Data Kepesertaan: data yang berisikan informasi mengenai peserta BPJS Kesehatan.
   b. Data Nonkapitasi: data klaim yang tidak masuk dalam sistem kapitasi.
   c. Data Kunjungan: data yang berisikan catatan kunjungan peserta ke fasilitas kesehatan tingakt pertama (FKTP).
   d. Data Diagnosis Sekunder: data yang mencakup informasi tambahan terkait diagnosis peserta.
2. Proyek ini dilaksanakan dengan mengikuti CRISP-DM yang terdiri atas tahap business understanding, data understanding, data preparation, modeling, evaluation, dan deployment.
3. Pendekatan algoritma yang digunakan dalam CRISP-DM yaitu XGBoost yang merupakan salah satu algoritma dari machine learning yang unggul ddalam melakukan klasifikasi jenis kunjungan peserta di FKTP,

termasuk kunjungan sehat atau sakit, dengan mempertimbangkan status kunjungan peserta.

1.4 Istilah dan Singkatan

Singkatan	Istilah
BPJS	Badan Penyelenggaraan Jaminan Sosial Kesehatan
JKN	Jaminan Kesehatan Nasional
FKTP	Fasilitas Kesehatan Tingkat Pertama
XGBoost	*Extreme Gradient Boosting*
CRISP-DM	*Cross Industry Standard Process for Data Mining*

II. STUDI LITERATURE

2.1 Prediction

Prediction dalam *machine learning* mengacu pada kemampuan model untuk memperkirakan atau memprediksi nilai atau label berdasarkan pola yang dipelajari dari data sebelumnya dan kemudian menerapkan pola tersebut untuk membuat keputusan atau estimasi tentang data baru [8]. Memprediksi adalah alat yang sangat berguna dalam pengambilan keputusan berbasis data, membantu individu dan organisasi untuk merencanakan dan mengantisipasi hasil di masa yang akan datang. Proses prediksi dilakukan setelah *data training* dilatih untuk menemukan pola atau hubungan antara *input* (fitur) dan *output* (label/target). Beberapa algoritma yang populer untuk prediksi dalam *machine learning* untuk kombinasi regresi dan klasifikasi yaitu *Gradient Boosting Machines* yang telah dikembangkan dalam beberapa algoritma lainnya yaitu XGBoost, LightGBM, dan CatBoost, kemudian *Neural Network*, dan *Support Vector Machine* (SVM) yang memiliki varian untuk regresi dan klasifikasi.

2.1.1 XGBoost

XGBoost (*Extreme Gradient Boosting*) adalah algoritma *machine learning* berbasis *ensemble learning*, yang menggunakan pendekatan *gradient boosting* untuk menghasilkan model prediktif yang kuat dan efisien. XGBoost menggunakan pendekatan yang lebih efisien dengan memanfaatkan *multi-threading* dan pengorginasasian data untuk mengurangi waktu pelatihan model. Hal ini memungkinkan XGBoost untuk menghasilkan model yang lebih cepat dan dengan akurasi yang tinggi dalam tugas klasifikasi dan prediksi. Algortima ini dirancang untuk menangani masalah *overfitting* dan dapat digunakan pada berbagai jenis *dataset*. XGBoost sering digunakan dalam tugas *prediction* karena kemampuannya dalam menghasilkan model prediktif yang akurat, baik untuk klasifikasi (kategori diskrit) maupun regresi (nilai kontinu) [7].

2.2 BPJS Kesehatan

BPJS Kesehatan merupakan lembaga yang dibentuk untuk menyelenggarakan program jaminana sosial di bidang Kesehatan di Indonesia. Tujuan utama didirikannya BPJS Kesehatan adalah untuk menjamin pemenuhan kebutuhan dasar hidup yang layak, khususnya dalam Kesehatan, bagi seluruh rakyat Indonesia. BPJS Kesehatan dibentuk berdasarkan Undang-Undang Nomor 24 Tahun 2011 tentang Badan Penyelenggara Jaminan Sosial, yang mengatur penyelenggaraan jaminan sosial di Indonesia. BPJS Kesehatan beroperasi dengan prinsip kemanusiaan, manfaat, dan keadilan sosial, yang bertujuan untuk memberikan perlindungan kesehatan kepada seluruh masyarakyat Indonesia sebagai hak dasar manusia. Hal ini menunjukkan bahwa BPJS Kesehatan memainkan peran penting dalam memastikan bahwa semua warga negara memiliki akses ke layanan kesehatan. Namun, BPJS Kesehatan dalam memberikan pelayanan publik yang efektif, memiliki tantangan termasuk perlunya evaluasi dan perbaikan yang berkelanjutan untuk memastikan keadilan dan kualitas dalam penyampaian layanan Kesehatan [2].

2.3 CRISP-DM

Cross Industry Standard Process for Data Mining (CRISP-DM) adalah model proses yang digunakan saat menjalankan proyek *data mining*, yang dikembangkan oleh perusahaan terkemuka seperti DaimlerChrysler, SPSS, NCR, dan OHRA. Model ini dibuat dengan tujuan menciptakan kerangka kerja yang dapat diandalkan dan diadaptasi untuk berbagai kebutuhan industry tanpa bergantung pada sektor tertentu atau teknologi yang digunakan. CRISP-DM dirancang untuk membuat proyek *data mining* lebih efisien, dapat diulang, serta mendorong kolaborasi yang lebih baik antar tim dengan enam fase utama yang terstruktur namun fleksibel [9].

p-ISSN xxxx–xxxx | e-ISSN xxxx–xxxx

Fungsi dari penggunaan CRISP-DM:

1. Memberikan panduan terstruktur untuk merancang langkah-langkah proyek *data mining* secara menyeluruh.
2. Memfasilitasi komunikasi antara anggota tim proyek dan dengan pihak eksternal melalui terminologi dan proses yang seragam.
3. Membantu dalam mendokumentasikan keputusan, hasil, dan pengalaman proyek untuk evaluasi dan pembelajaran di masa depan.
4. Memberikan fleksibilitas untuk menyesuaikan proses dengan kebutuhan spesifik proyek atau industri.
5. Mengurangi waktu dan biaya melalui pendekatan proses yang standar dan teruji.

Tujuan CRISP-DM:

1. Menstandarisasi proses *data mining* agar lebih dapat diandalkan.
2. Meningkatkan pemahaman dan pengelolaan proyek *data mining* bagi berbagai pemangku kepentingan.
3. Memberikan panduan langkah demi langkah yang dapat digunakan baik oleh analisis data pemula maupun berpengalaman.
4. Meningkatkan kepercayaan pelanggan terhadap proyek *data mining* sebagai praktik yang dapat diprediksi hasilnya.
5. Mendukung pengembangan solusi *data mining* yang berkelanjutan dan lebih baik di masa mendatang.

III. METODE PENELITIAN

3.1 CRISP-DM

Proyek ini menggunakan model *CRoss Industry Standard Process for Data Mining* (CRISP-DM) dalam pelaksanaan yang dilakukan untuk mencapai tujuan proyek. CRISP-DM adalah model yang dirancang untuk membantu dalam pelaksanaan proyek *data mining*. Model ini terdiri beberapa fase, tugas generic, dan tugas khusus yang memungkinkan pengguna untuk merencanakan dan mendokumentasikan proyek dengan efektif. Adapun fase utama dalam CRISP-DM yaitu *business understanding, data understanding, data preparation, modelling, evaluation,* dan *deployment*.

3.1.1 Business Understanding

Tujuan utama dari analisis data ini adalah untuk membantu BPJS Kesehatan dalam meningkatkan efisiensi dan efektivitas pelayanan kesehatan melalui prediksi jenis kunjungan peserta di fasilitas kesehatan tingkat pertama (FKTP). Dengan prediksi yang tepat, BPJS Kesehatan dapat mengidentifikasi apakah kunjungan peserta bersifat preventif atau untuk penanganan penyakit. Informasi ini memungkinkan BPJS dan FKTP untuk merencanakan sumber daya yang lebih optimal dan mengalokasikan tenaga medis, obat-obatan, serta peralatan sesuai kebutuhan nyata di lapangan. Segmentasi peserta berdasarkan jenis kunjungan ini diharapkan dapat memberikan manfaat bisnis yang signifikan, termasuk:

a. Mengurangi potensi kelebihan atau kekurangan sumber daya dengan mengetahui jenis kunjungan peserta yang diprediksi akan datang.
b. Prediksi jenis kunjungan membantu FKTP memberikan pelayanan yang lebih cepat dan tepat, baik untuk kunjungan sehat maupun kunjungan sakit.
c. BPJS dapat mengalokasikan anggaran yang lebih efektif, misalnya dengan mengurangi pengeluaran untuk fasilitas preventif saat terjadi peningkatan kunjungan sakit, atau sebaliknya.

3.1.2 Data Understanding

Data Understanding membantu dalam mengidentifikasi karakteristik, pola, dan kualitas data yang akan diolah. Tanpa pemahaman yang baik, proses data mining bisa menghasilkan interpretasi yang salah dan keputusan yang tidak tepat. Ada beberapa hal yang akan dilakukan pada tahapan ini:

a. Pengumpulan data

   Mengumpulkan data yang diperlukan untuk analisis, termasuk informasi tentang diagnosis (ICD-10), jenis fasilitas kesehatan, pola rujukan, riwayat kunjungan, dan variabel-variabel lainnya yang relevan.
b. Analisis Data

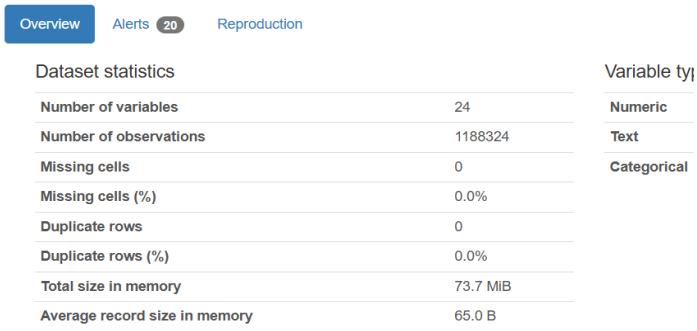
   Melakukan eksplorasi awal untuk memahami distribusi data, pola umum, serta variabilitas pada fitur diagnosis, jenis fasilitas, pola rujukan, dan riwayat kunjungan.
c. Validasi Data

Memastikan data yang dikumpulkan valid dan berkualitas, misalnya dengan memeriksa data yang hilang, mengidentifikasi nilai outlier, atau inkonsistensi dalam fitur.

3.1.2.1 Data Overview

Dataset FKTP ini mencakup informasi seperti nomor kunjungan, diagnosis ICD-10, jenis layanan, serta pola rujukan. Data ini berguna untuk menganalisis efektivitas pelayanan kesehatan yang diberikan melalui model pembayaran kapitasi. Untuk penjelasan deskriptif dari setiap kolom dalam data ini, *dataset* FKTP dikombinasikan dengan metadata yang juga memberikan deskriptif dari setiap kode di *dataset*. *Dataset* FKTP yang teah dikombinasikan ini, dapat dilihat dalam Lampiran 1.

Analisis awal dilakukan dengan menampilkan ringkasan *dataset* yang dihasilkan dari proses analisis awal menggunakan Ydata Profiling.



Gambar 3.1 Dataset Overview

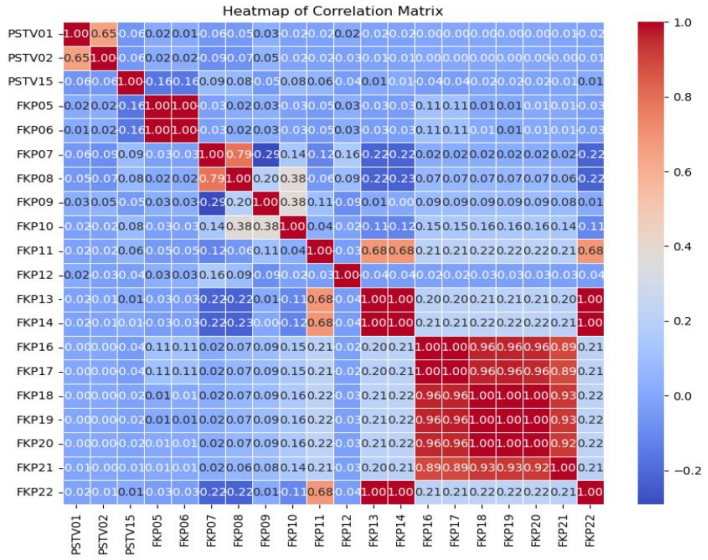
Informasi yang dapat diperoleh yaitu:

a. *Dataset* memiliki 24 kolom berupa data nuemrik, teks, dan kategori. Dengan terdapat 16 kolom dengan tipe data numerik, 4 kolom dengan tipe data teks, dan 4 kolom dengan tipe data kategori.
b. *Dataset* ini memiliki 1.188.324 baris, yang menunjukkan bahwa ukuran *dataset* cukup besar.
c. Tidak ada nilai yang hilang pada *dataset* sehingga dapat dikatakan bahwa data bersih karena sudah dilakukan proses *data cleaning*.
d. Tidak ada baris yang terduplikasi, sehingga setiap entri data dianggap unik.

Dataset Overview pada Gambar 3.1 memberikan visualisasi lengkap mengenai struktur awal dan distribusi data untuk mendukung validasi dan pemahaman awal.

3.1.2.2 Correlations

Correlation adalah ukuran statistic yang menggambarkan sejauh mana dua variabel memili hubungan yang saling linier. Heatmap pada Gambar 3.2 menunjukkan hubungan korelasi antar variabel dalam *dataset*. Skala warna digunakan untuk merepresentasikan kekuatan hubungan, di mana warna merah gelap menunjukkan korelasi positif yang kuat, warna putih menunjukkan tidak adanya korelasi, dan warna merah gelap menunjukkan korelasi negatif yang kuat. Variabel yang memiliki korelasi positif tinggi cenderung saling meningkat bersama, sedangkan variabel dengan korelasi negatif tinggi memiliki hubungan berlawanan.



Gambar 3.2 Heatmap Correlation

Heatmap ini menunjukkan hubungan korelasi antar kolom pada dataset FKTP Kapitasi. Korelasi dihitung dengan menggunakan nilai koefisien Pearson, yang berkisar antara -1 hingga 1:

a. Nilai positif (mendekati 1) menunjukkan hubungan positif yang kuat antara dua variabel, artinya saat nilai satu variabel meningkat, variabel lainnya juga cenderung meningkat.
b. Nilai negatif (mendekati -1) menunjukkan hubungan negatif yang kuat, artinya saat satu variabel meningkat, yang lain cenderung menurun.
c. Nilai mendekati 0 mengidentifikasikan bahwa tidak adanya hubungan linear yang signifikan.

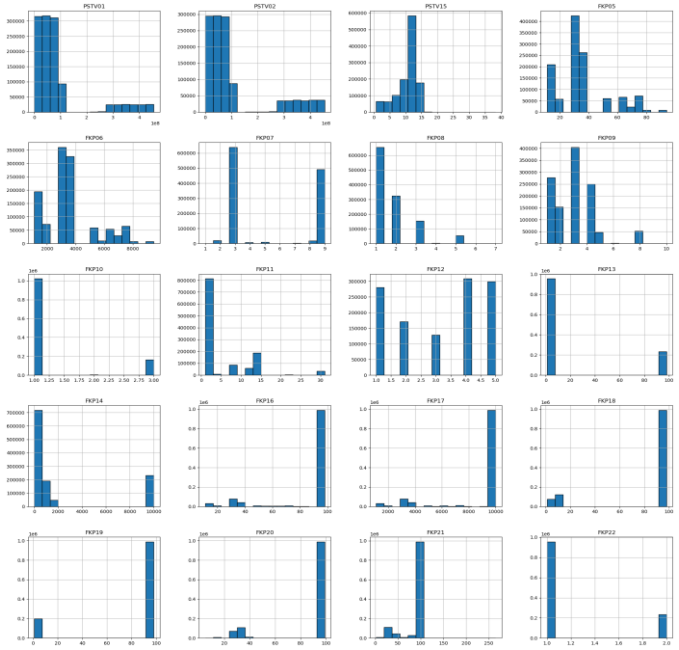
p-ISSN xxxx–xxxx | e-ISSN xxxx–xxxx

Fitur utama yang paling berhubungan dengan FKP22 (Jenis Kunjungan FKTP) adalah FKP13 (Status Kunjungan Peserta), dengan korelasi bernilai 1.00. Hal ini menunjukkan bahwa status pulang, seperti sembuh, rujukan, atau rawat inap, berkaitan dengan jenis kunjungan, apakah kunjungan tersebut bersifat sehat atau sakit. Selain itu, FKP14 (Kode Diagnosis ICD-10) memiliki korelasi sebesar 1.00, yang mengindikasikan bahwa jenis diagnosis juga memengaruhi jenis kunjungan, terutama untuk penyakit kronis atau kasus tertentu. FKP11 (Jenis Poli FKTP) juga memiliki hubungan rendah yaitu sekitar 0.15 sampai 0.21, yang menggambarkan bahwa jenis poli, seperti poli umum atau spesialis, dapat memberikan indikasi jenis kunjungan peserta. Sementara itu, fitur pendukung, seperti FKP16 dan FKP17 yang menjelaskan lokasi fasilitas tujuan rujukan, menunjukkan korelasi sangat rendah dengan jenis kunjungan, meskipun tetap relevan dalam kasus tertentu, seperti rujukan ke fasilitas kesehatan spesifik. Fitur lain, yang memiliki hubungan lemah dengan FKP22, dapat memberikan konteks tambahan dalam analisis.

Karena proyek ini berfokus untuk melakukan klasifikasi kunjungan sehat atau sakit, sehingga atribut yang memiliki hubungan signifikan dengan atribut target FKP13 (Status Kunjungan Peserta) menjadi prioritas dalam analisis. Atribut-atribut yang dipilih antara lain FKP14 (Kode Diagnosis ICD-10) dan FKP22 (Jenis Kunjungan FKTP), yang menunjukkan hubungan yang relevan dengan target klasifikasi. Selain itu, meskipun atribut FKP11 (Jenis Poli FKTP) memiliki korelasi sebesar 0,68, atribut ini tetap digunakan karena informasi yang terkandung di dalamnya dianggap dapat memperkaya model dalam melakukan pengklasifikasian secara lebih akurat. Pemilihan atribut ini dilakukan dengan tujuan untuk memastikan bahwa model yang dihasilkan memiliki performa yang optimal dengan mempertimbangkan relevansi dan kontribusi informasi dari setiap fitur.

3.1.2.3 Data Integer Histogram Visualization

Dengan memperhatikan Gambar 3.3, dapat dilihat tampilan dari histogram distribusi data dari setiap data integer dalam *dataset*. Setiap histogram menggambarkan distribusi data dari masing-masing kolom numerik. Visualisasi ini membantu dalam memahami pola data dan potensi outlier atau nilai dominan dalam setiap data yang integer.



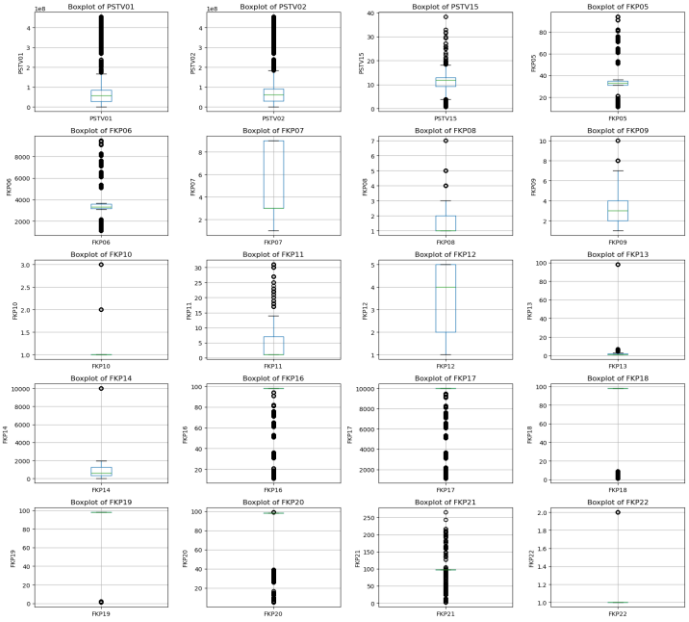
Gambar 3.3 Data Numerical Histogram Visualization

Sebagian besar kolom dalam data menunjukkan distribusi yang tidak merata, dengan data yang cenderung terkonsentrasi pada nilai tertentu, mengindikasikan adanya distribusi yang skewed, yaitu distribusi data yang tidak simetris, sehingga nilai-nilai datanya cenderung terkumpul lebih banyak di salah satu sisi rata-rata. Sebagai contoh, kolom seperti FKP16, FKP17, dan FKP18 menunjukkan data yang sangat terpusat pada satu nilai tertentu, yang mencerminkan kurangnya variasi dalam distribusi data tersebut. Visualisasi ini memberikan gambaran awal mengenai pola data, keberadaan potensi outlier, serta nilai-nilai dominan dalam atribut numerik, yang penting untuk tahapan analisis selanjutnya, termasuk pre-processing dan pemilihan fitur.

3.1.2.4 Data Integer Boxplot Visualization

Gambar 3.4 menunjukkan diagram *boxplot* dari setiap data di atribut numerik dengan tujuan menganalisis pola distribusi data dan memahami kemungkinan outlier pada data. Selain membantu mengidentifikasi potensi outlier, boxplot juga digunakan untuk menentukan atribut integer yang bersifat kategorikal. Atribut yang memiliki banyak nilai di luar rentang distribusi normal menunjukkan adanya keberagaman data, di mana data mayoritas mungkin didominasi oleh beberapa kategori tertentu, sedangkan data minoritas tersebar dalam berbagai nilai lainnya. Hal ini penting untuk dianalisis agar

dapat menangani atribut-atribut tersebut dengan tepat, seperti melalui teknik *encoding* atau normalisasi data.



Gambar 3.**Error! No text of specified style in document.**.4 Data Numerical Boxplot Visualization

Hasil analisis visualisasi boxplot memberikan gambaran mengenai distribusi data numerik ataupun kategorikal dan potensi outlier pada atribut dalam dataset. Berdasarkan hasil tersebut, atribut numerik dapat dikelompokkan sebagai berikut:

1. Atribut yang tidak memiliki outlier yaitu FKP07 dan FKP12. Atribut-atribut ini menunjukkan distribusi data yang stabil dan berada sepenuhnya dalam rentang distribusi normal tanpa adanya nilai yang ekstrem.
2. Atribut yang memiliki sedikit outlier dengan jarak dekat yaitu FKP08, FKP09, dan FKP10. Outlier pada atribut-atribut ini relatif sedikit dan berada dekat dengan batas distribusi utama, sehingga tidak memberikan pengaruh besar pada analisis data.
3. Atribut yang memiliki sedikit outlier dengan jarak jauh yaitu FKP13, FKP14, FKP18, FKP19, dan FKP22. Outlier pada atribut-atribut ini lebih signifikan karena jaraknya yang jauh dari distribusi utama, yang berpotensi memengaruhi analisis jika tidak ditangani dengan benar.
4. Atribut yang memiliki banyak outlier adalah PSTV01, PSTV02, PSTV15, FKP05, FKP06, FKP11, FKP16, FKP17, FKP20, dan FKP21. Atribut-atribut ini memiliki banyak data yang berada di luar batas distribusi normal, menunjukkan adanya variasi yang sangat besar antara data mayoritas dan minoritas.

3.1.2.5 Data Object Unique Count Analysis

Gambar 3.5 untuk menampilkan nilai unik dari setiap kolom objek dalam *dataset*.

```
Kolom: FKP02
Jumlah nilai unik: 1188324
Nilai unik: ['1687160334' '1457720536' '1732989007' ... '342530321P000054'
 '342530221P000002' '243990921P000026']
----------------------------------------
Kolom: FKP14A
Jumlah nilai unik: 1411
Nilai unik: ['' 'A00' 'A01' ... 'Z97' 'Z98' 'Z99']
----------------------------------------
Kolom: FKP15
Jumlah nilai unik: 5170
Nilai unik: ['9999' 'A00' 'A000' ... 'Z992' 'Z999' 'Z998']
----------------------------------------
Kolom: FKP15A
Jumlah nilai unik: 5169
Nilai unik: ['9999' 'Cholera' 'Cholera due to Vibrio cholerae 01, biovar cholerae' ...
 'Dependence on renal dialysis'
 'Dependence on unspecified enabling machine and device'
 'Dependence on other enabling machines and devices']
----------------------------------------
Kolom: FKP05_Detail
Jumlah nilai unik: 34
Nilai unik: ['DKI Jakarta' 'Bengkulu' 'Jawa Barat' 'Nusa Tenggara Barat'
 'DI Yogyakarta' 'Kalimantan Timur' 'Jawa Timur' 'Riau' 'Kepulauan Riau'
 'Jawa Tengah' 'Sumatera Utara' 'Aceh' 'Sulawesi Utara' 'Bali'
...
Kolom: FKP22_Detail
Jumlah nilai unik: 2
Nilai unik: ['Kunjungan sehat\xa0' 'Kunjungan sakit']
```

Gambar 3.5 Data Obejct Unique Count Analysis

Penjelasan dari output yang menggambarkan distribusi dan variasi nilai unik pada kolom-kolom dalam *dataset*:

1. Kolom FKP02 memiliki jumlah nilai unik yang sangat besar, yaitu 1.188.324. Nilai uniknya berupa kode angka besar, menunjukkan bahwa kolom ini mungkin merepresentasikan suatu identifikasi unik, yaitu nomor peserta.
2. Pada kolom FKP14A terdapat 1.411 nilai unik dengan pola berupa kode seperti A00, A01, hingga Z99. Tampaknya kolom ini merepresentasikan kode diagnosis kesehatan berdasarkan ICD-10.
3. Kolom FKP15 memiliki 5.170 nilai unik, menunjukkan variasi yang lebih luas dibandingkan FKP14A. Nilai dalam kolom ini dapat berkaitan dengan rincian spesifik diagnosis atau tindakan medis.
4. Kolom FKP15A memiliki jumlah nilai unik hampir sama dengan FKP15, yaitu 5.169, namun berisi deskripsi yang lebih mendetail, seperti ketergantungan pada alat medis tertentu. Kolom ini melengkapi informasi yang diberikan oleh FKP15 dengan memberikan deskripsi tambahan.
5. Kolom FKP05_Detail memiliki 34 nilai unik, memuat nama-nama provinsi di Indonesia. Kolom ini menunjukkan distribusi geografis atau lokasi fasilitas kesehatan terkait.
6. Kolom FKP11_Detail memiliki 22 nilai unik, berisi nama-nama jenis poliklinik yang tersedia di fasilitas kesehatan. Kolom ini mencerminkan jenis layanan kesehatan yang diberikan.
7. Dalam kolom FKP13_Detail terdapat 7 nilai unik yang menjelaskan status kunjungan pasien, seperti kunjungan

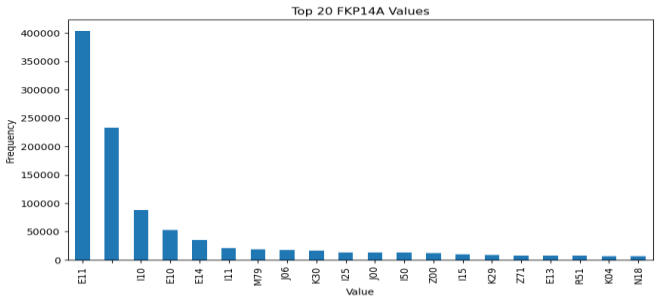
p-ISSN xxxx–xxxx | e-ISSN xxxx–xxxx

sehat, rujuk, atau status lainnya. Kolom ini relevan untuk menganalisis pola dan jenis kunjungan pasien.

8. Kolom ICD10_Text memiliki 1.411 nilai unik, sama dengan FKP14A. Kolom ini memberikan deskripsi lebih rinci mengenai kode diagnosis yang ada di FKP14A, membantu interpretasi data medis.
9. Kolom FKP22_Detail hanya memiliki 2 nilai unik, yaitu kunjungan sehat dan kunjungan sakit. Kolom ini merupakan fitur penting yang digunakan dalam klasifikasi, karena menentukan jenis kunjungan yang dilakukan pasien.

3.1.2.6 Data Object FKP14A (Kode Diagnosis umum) TOP-20 Values Frequency Visualization

Gambar 3.6 membantu memahami distribusi frekuensi nilai-nilai teratas dalam kolom FKP14A.

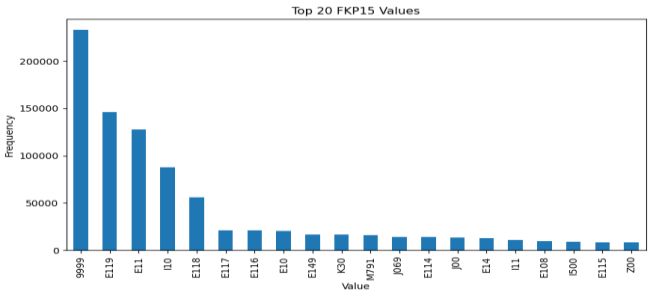


Gambar 3.6 Data Cateogrical FKP15 TOP-20 Values Frequency Visualization

Grafik menunjukkan distribusi frekuensi 20 nilai teratas dari kolom FKP14A. Nilai E11 memiliki frekuensi tertinggi dengan lebih dari 400.000 kemunculan, diikuti oleh I10 dan E10 yang masing-masing memiliki frekuensi lebih rendah secara bertahap. Sementara nilai lainnya, seperti I15, K29, dan N18, memiliki frekuensi yang jauh lebih rendah. Visualisasi ini membantu memahami pola dominasi nilai tertentu dalam data kategorikal dan mengidentifikasi potensi nilai-nilai penting atau outlier.

3.1.2.7 Data Object FKP15 (Kode Diagnosis lanjut) TOP-20 Values Frequency Visualization

Gambar 3.7 merupakan grafik menunjukkan distribusi frekuensi nilai-nilai teratas dalam kolom FKP15.

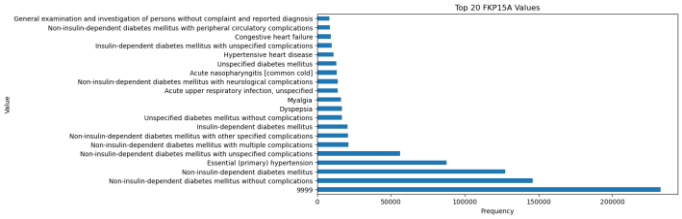


Gambar 3.7 Data Object FKP15 (Kode Diagnosis lanjut) TOP-20 Values Frequency Visualization

Nilai 9999 memiliki frekuensi tertinggi, diikuti oleh E119, E11, dan I10. Nilai-nilai ini memiliki frekuensi yang menurun secara bertahap. Nilai-nilai lain seperti E116, E149, dan M79.1 memiliki frekuensi yang jauh lebih rendah. Grafik ini memberikan wawasan tentang nilai yang paling dominan dalam kolom FKP15, yang dapat membantu dalam analisis lebih lanjut, seperti mendeteksi dominasi nilai tertentu atau keberadaan nilai outlier.

3.1.2.8 Data Object FKP15A (Deskripsi Diagnosis lanjut) TOP-20 Values Frequency Visualization

Gambar 3.8 memmberikan gambar grafik menunjukkan distribusi frekuensi dari 20 nilai teratas di kolom FKP15A.



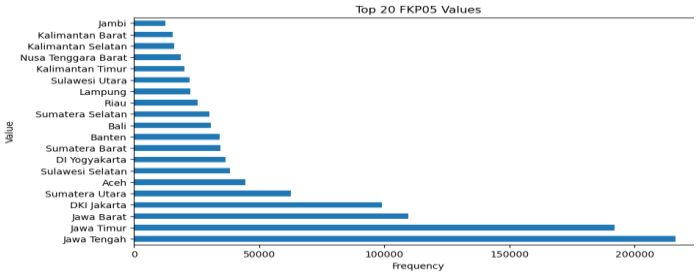
Gambar 3.8 Data Object FKP15A (Deskripsi Diagnosis Lanjut)

Nilai 9999 memiliki frekuensi tertinggi, diikuti oleh deskripsi seperti "Non-insulin-dependent diabetes mellitus without complications" dan "Essential (primary) hypertension". Frekuensi ini menurun pada nilai-nilai berikutnya. Grafik ini memberikan wawasan tentang dominasi beberapa kategori tertentu dalam kolom tersebut, yang dapat membantu dalam analisis lebih lanjut seperti identifikasi nilai-nilai umum atau mendeteksi data yang tidak seimbang.

3.1.2.9 Data Object FKP05 (Provinsi) TOP 20 Values Frequency Visualization

Gambar 3.9 menunjukkan grafik distribusi frekuensi 20 nilai teratas dari kolom FKP05 (Provinsi), yang merepresentasikan

nama-nama provinsi di Indonesia berdasarkan jumlah kemunculan dalam *dataset*.

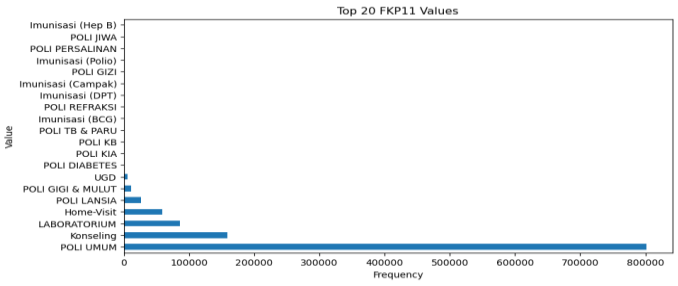


Gambar 3.9 Data Object FKP05 (Provinsi) TOP-20

Hasil grafik menunjukkan bahwa provinsi dengan jumlah frekuensi tertinggi yaitu Provinsi Jawa Tengah, setelah diikuti Provinsi Jawa Timur dan Jawa Barat. Provinsi Jambi, Kalimantan barat, dan Kalimantan Selatan memiliki jumlah frekuensi yang jauh lebih kecil dibandingkan provinsi lainnya yang mengindikasikan distribusi yang tidak merata. Pola ini menunjukkan bahwa provinsi dengan populasi besar atau layanan kesehatan yang lebih aktif dilakukan cenderung memiliki banyak data dalam kolom ini.

3.1.2.10 Data Object FKP11 (Jenis Poliklinik) TOP-20 Values Frequency Visualization

Gambar 3.10 menunjukkan distribusi frekuensi 20 nilai teratas dari kolom FKP11, yang mewakili jenis poliklinik atau layanan Kesehatan dalam fasilitas Kesehatan. Visualisasi ini membantu dalam mengidentifikasi prioritas layanan dalam fasilitas kesehatan dan menganalisis pola penggunaan layanan kesehatan di *dataset*.

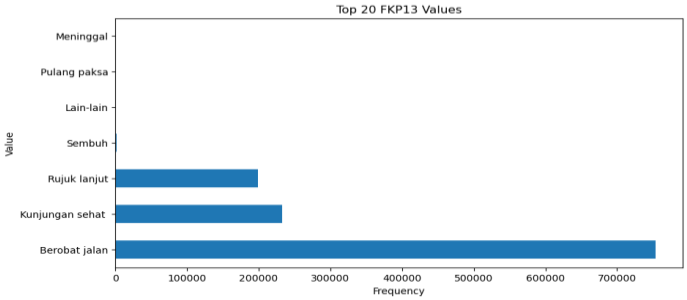


Gambar 3.10 Data Object FKP11 (Jenis Poliklinik)

Poli Umum merupakan jenis layanan yang memiliki frekuensi tertinggi dan signifikan dibandingkan dengan jenis layanan lainnya. Dengan nilai sebesar 800.000 kali, menunjukkan bahwa layanan ini paling sering digunakan oleh pasien. Jenis layanan Konseling dan Laboratorium juga memiliki frekuensi yang cukup tinggi, namun jauh lebih kecil dibandingkan dengan jenis Poli Umum. Untuk layanan spesifik seperti Poli Lansia, Poli Gigi dan Mulut, dan beberapa jenis imunisasi memiliki frekuensi yang lebih kecil. Distribusi ini mencerminkan bahwa kebutuhan layanan kesehatan umum mendominasi data, sementara layanan spesifik lebih jarang digunakan.

3.1.2.11 Data Object FKP13 (Status Kunjungan pasien) TOP-20 Values Frequency Visualization

Gambar 3.11 menunjukkan distribusi frekuensi 20 nilai teratas dari kolom FKP13, yang merepresentasikan status kunjungan pasien fasilitas Kesehatan.



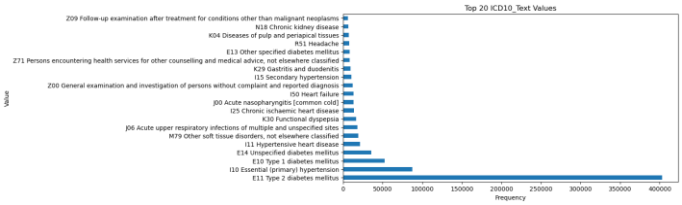
Gambar 3.11 Data Object FKP13 (Status Kunjungan Pasien)

Status “Berobat Jalan” memiliki frekuensi tertinggi, yaitu mendekati 750.00, yang menunjukkan bahwa mayoritas kunjungan pasien adalah keperluan pengobatan rawat jalan. Kemudian, status “Kunjungan Sehat” memiliki frekuensi signifikan meskipun jauh lebih kecil dibandingkan “Berobat Jalan”. Hal ini mengindifikasikan kunjungan tanpa keluhan Kesehatan yang lebih jarang terjadi. Lalu, status ”Rujuk Lanjut” memiliki jumlah yang cukup kecil jika dibandingkan dengan dua status sebelumnya, yang mengindifikasikan bahwa sebagaian kecil pasien membutuhkan layanan lanjutan di fasilitas kesehatan yang lebih besar. Status lainnya, yang memiliki frekuensi sangat rendah, menunjukkan kasus yang lebih jarang terjadi.

3.1.2.12 Data Object ICD10_Text (Deskripsi Diagnosis umum) TOP-20 Values Frequency Visualization

Gambar 3.12 menunjukkan distribusi frekuensi 20 nilai teratas dari kolom ICD10_Text yang mengindifikasikan deskripsi lengkap untuk diagnosis umum terhadap peserta.

p-ISSN xxxx–xxxx | e-ISSN xxxx–xxxx



Gambar 3.12 Data Object ICD10_Text (Deskripsi Diagnosis umum)

Grafik ini merupakan informasi pelengkap dari Gambar 3.6 yang mengandung informasi dari setiap kode. Grafik menunjukkan distribusi frekuensi 20 nilai teratas dari kolom FKP14A. Nilai E11 memiliki frekuensi tertinggi dengan lebih dari 400.000 kemunculan, diikuti oleh I10 dan E10 yang masing-masing memiliki frekuensi lebih rendah secara bertahap. Sementara nilai lainnya, seperti I15, K29, dan N18, memiliki frekuensi yang jauh lebih rendah.

3.1.2.13 Conclusion

Setelah melakukan analisis melalui langkah *data understanding* terhadap data utama yang digunakan, maka untuk pelaksanaan proyek atribut-atribut yang digunakan berikut sebagai *feature* dan target untuk melatih model dalam mengklasifikasikan prediksi kunjungan sehat atau sakit. Atribut yang dipilih terdiri atas data integer dam data objek, yaitu:

a. FKP05: merepresentasikan provinsi asal pasien. Atribut ini membantu memahami distribusi kunjungan berdasarkan wilayah geografis, yang dapat berkontribusi pada pola kunjungan Kesehatan.
b. FKP11: atribut ini menunjukkan jenis poliklinik yang dikunjungi. Atribut ini memberikan informasi penting mengenai jenis layanan kesehatan yang diperoleh pasien.
c. FKP13: status kunjungan pasien, seperti "Berobat Jalan" atau "Kunjungan Sehat". Atribut ini menjadi target utama dalam klasifikasi.
d. FKP14: kode diagnosis kesehatan pasien. Atribut ini relevan dalam mengidentifikasi pola penyakit atau alasan kunjungan pasien.
e. FKP14A: kode tambahan untuk diagnosis pasien, memberikan rincian lebih lanjut tentang kondisi kesehatan pasien.
f. FKP15: deskripsi tindakan atau terapi yang diberikan. Atribut ini berguna untuk memahami intervensi yang dilakukan selama kunjungan.
g. FKP15A: detail tambahan terkait tindakan medis, seperti ketergantungan alat medis. Atribut ini menambahkan informasi spesifik yang penting untuk klasifikasi.
h. FKP22: mengindikasikan jenis kunjungan, yaitu "Kunjungan Sehat" atau "Kunjungan Sakit". Atribut ini mendukung klasifikasi yang lebih langsung terkait target.
i. FKP05_Detail: nama provinsi asal pasien. Atribut ini memberikan representasi tekstual dari FKP05 untuk memperkaya konteks geografis.
j. FKP11_Detail: nama poliklinik yang dikunjungi. Atribut ini melengkapi informasi dari FKP11 dengan deskripsi yang lebih mudah dipahami.
k. FKP13_Detail: deskripsi status kunjungan pasien. Atribut ini merupakan pelengkap dari FKP13 dan membantu klasifikasi dengan konteks lebih spesifik.
l. ICD10_Text: deskripsi diagnosis yang menjelaskan kode di FKP14A, memberikan wawasan yang lebih kaya mengenai kondisi pasien.
m. FKP22_Detail: deskripsi jenis kunjungan pasien. Atribut ini memperkuat informasi yang ada di FKP22.

Pemilihan atribut-atribut ini berlatarkan berbagai alasan yang diantaranya adalah korelasi tinggi antara atribut dengan atribut target, manfaat atribut dalam memberikan informasi klasifikasi yang lebih detail pada hasil klasifikasi yang dirancang, dan manfaat dari informasi diagnosis yang disimpan oleh atribut. Walaupun demikian, atribut-atribut ini akan diperbaharui dengan menerapkan tahapan *pre-processing* untuk memastikan data yang *clean*, tidak *noise*, tidak *outlier*, dan terdistribusi dengan baik. Disamping itu akan dilakukan beberapa pengaturan pada isi atribut-atribut tertentu yang disesuaikan dengan informasi dari *dataset* sekunder. Hal ini bertujuan untuk memaksimalkan informasi yang kaya kedalam atribut sehingga mampu memberikan klasifikasi kunjungan sakit dan sehat yang lebih detail.

3.1.3 Data Preparation

Fase ini menyiapkan data untuk *modelling* yang termasuk didalamnya yaitu pembersihan data, transformasi, dan

pemilihan fitur yang relevan. Ada beberapa hal yang akan dilakukan pada tahapan ini:

a. Memilah Data

   Memilih fitur-fitur yang relevan untuk analisis prediksi, yaitu diagnosis (ICD-10), jenis fasilitas kesehatan, pola rujukan, dan riwayat kunjungan.

b. Membersikan Data

   Menghapus data yang duplikat, mengisi data yang hilang, dan mengatasi nilai outlier untuk memastikan kualitas data tetap terjaga.

c. Mengkonstruksi Data

   Jika diperlukan, menciptakan fitur-fitur baru yang mungkin memiliki nilai prediktif lebih tinggi, seperti lama kunjungan rata-rata atau jumlah rujukan dalam periode tertentu.

d. Menentukan Label Data

   Menetapkan label target untuk klasifikasi, yaitu kategori seperti “Berobat Jalan”, “Rujuk Lanjut”, “Sembuh”, “Pulang Paksa”, “Kunjungan Sehat”, dan lainnya, berdasarkan data riwayat kunjungan atau deskripsi diagnosis.

e. Mengintegrasikan Data

   Menggabungkan data dari berbagai sumber jika diperlukan agar data yang digunakan lebih lengkap dan komprehensif.

3.1.3.1 Dataset Attribute Selection

Pada bagian ini, atribut-atribut akan digunakan dalam pemodelan sesuai hasil *data understanding*,dipilih untuk dianalisis. Atribut-atribut ini dianggap penting untuk mendukung klasifikasi status kunjungan sehat atau sakit.

```
# view dataset before selection
df.info()
# Chosing the relevan columns
selected_columns = ['FKP05', 'FKP11', 'FKP13', 'FKP14', 'FKP15', 'FKP15A', 'FKP22', 'FKP05_Detail', 'FKP11_Detail', 'FKP13_Detail', 'ICD10_Text', 'FKP22_Detail']
selected_data = df[selected_columns]
print("Informasi Kolom yang Dipilih:")
print(selected_data.info())
```

Atribut-atribut yang dipilih dibuat dalam variabel ‘selected_columns’, yang dimasukkan dalam *dataset* baru yang hanya berisi atribut yang terpilih, yaitu ‘selected_data’. Lalu, isi informasi *dataset* ditampilkan untuk memastikan hanya atribut yang relevan yang disertakan.

3.1.3.2 Dataset Attribute Reconstruction

Pada bagian ini, nama atribut yang terpilih, diubah untuk membuatkan lebih deskriptif dan mudah dipahami, terutama jika data akan digunakan untuk model atau laporan.

```
# Changing the name of the chosen columns
column_mapping = {
    'FKP05': 'Kode_Provinsi',
    'FKP11': 'Kode_Jenis_Poliklinik',
    'FKP13': 'Kode_Status_Kunjungan_Peserta',
    'FKP14': 'Kode_Diagnosis',
    'FKP15': 'Kode_Diagnosis_Detail',
    'FKP15A': 'Nama_Diagnosis_Detail',
    'FKP22': 'Kode_Jenis_Kunjungan',
    'FKP05_Detail': 'Nama_Provinsi',
    'FKP11_Detail': 'Jenis_Poliklinik',
    'FKP13_Detail': 'Status_Kunjungan_Peserta',
    'ICD10_Text': 'Nama_Diagnosis',
    'FKP22_Detail': 'Jenis_Kunjungan',
}
selected_data.rename(columns=column_mapping, inplace=True)
```

Nama atribut diubah dengan mendefinisikan perubahan nama atribut lama menjadi nama baru dalam sebuah kamus ‘column_mapping’. Kemudian, nama atribut dalam *dataset* diubah dengan ‘selected_data.rename()’.

3.1.3.3 Dataset Attribute Cleaning

Pada bagian ini, data yang duplikat dihapus untuk memastikan *dataset* bersih dan mengurangi potensi bias dalam analisis.

```
# Delete data duplicates
selected_data = selected_data.drop_duplicates()
print("Informasi Dataset Setelah Mengganti Nama Kolom:")
print(selected_data.info())
```

Data yang duplikasi dihapus dengan menggunakan fungsi ‘drop_duplicates()’ pada semua atribut. Setelag itu, ditampilkan informasi *dataset* untuk memastikan data bersih.

```
# Mengecek jumlah nilai null pada setiap kolom
missing_values = selected_data.isnull().sum()

# Menampilkan kolom yang memiliki nilai null
print("Kolom dengan nilai null:")
print(missing_values[missing_values > 0])
```

Tujuan bagian kode ini adalah untuk mengetahui kolom mana saja yang memiliki data kosong dan perlu

p-ISSN xxxx–xxxx | e-ISSN xxxx–xxxx

diperbaiki dengan menggunakan ‘isnull().sum()’. Lalu, ditampilkan jumlahnya dengan menggunakan ‘print()’. Langkah ini penting untuk memastikan *dataset* bersih sebelum digunakan dalam analisis serta pelatihan model.

```
# Menghapus baris yang memiliki NaN pada kolom Jenis_Poliklinik
selected_data = selected_data.dropna(subset=['Jenis_Poliklinik'])

# Mengisi NaN pada kolom Nama_Diagnosis dengan nilai 'Tidak Ada'
selected_data['Nama_Diagnosis'] = selected_data['Nama_Diagnosis'].fillna('Tidak Ada')

# Outputkan hasil untuk memastikan perubahan
print(selected_data.isnull().sum())
```

Kemudian, setiap baris yang kosong pada kolom ‘Jenis_Poliklinik’ dihapus, pada kolom ‘Nama_Diagnosis’ yang memiliki nilai kosong diganti menjadi ‘Tidak Ada’. Dipastikan kembali apakah masih ada kolom yang memiliki baris kosong atau tidak di bagian akhir.

3.1.3.4 Defining Labels for the Dataset

Proses label dimulai dengan membuat kategori baru pada kolom ‘Label’ dengan menyederhanakan beberapa nilai yang ada di kolom ‘Kode_Status_Kunjungan_Peserta’, berikut langkah-langkahnya:

a. Mengambil nilai unik

```
# Ambil nilai unik dari kolom 'Status_Kunjungan_Peserta'
unique_values = selected_data['Kode_Status_Kunjungan_Peserta'].unique()
```

Tujuannya yaitu untuk memahami kategori asli sebelum memetakan ulang ke label yang baru. Daftar nilai unik diambil dari kolom ‘Kode_Status_Kunjungan_Peserta’.

b. Menentukan label baru

```
# Fungsi untuk menentukan label numerik
def determine_label(label):
    if label == 5 or label == 6 or label == 7:  # Gabungkan Pulang Paksa (5), Lain-lain (6), dan Meninggal (7)
        return 4  # Label baru untuk kategori "Lainnya"
    elif label == 98:  # Kunjungan Sehat tetap dengan label 3
        return 3
    return label - 1  # Mengubah label 1 menjadi 0, 2 menjadi 1, 3 menjadi 2, dst.
```

Untuk menyederhanakan analisis, nilai 5 (Pulang Paksa), 6 (Lain-lain), dan 7 (Meninggal) digabung menjadi kategori baru dengan label 4. Nilai 98 (Kunjungan Sehat) tetap dipetakan sebagai 3, dan nilai lainnya seperti 1, 2, dan 3 dikurangi 1 sehingga nilai baru dimulai dari 0.

c. Menerapkan fungsi ke kolom ‘Kode_Status_Kunjungan_Peserta’

```
# Terapkan fungsi ke kolom 'Label'
selected_data['Label']=selected_data['Kode_Status_Kunjungan_Peserta'].apply(determine_label)
```

Fungsi ‘determine_label’ diterapkan pada setiap baris di kolom ‘Kode_Status_Kunjungan_Peserta’ untuk menghasilkan kolom baru ‘Label’ dengan kategori yang telah disederhanakan.

d. Menampilkan distribusi baru

```
# Tampilkan distribusi baru dari kolom 'Label' dan 'Status_Kunjungan_Peserta'
print(selected_data[['Label','Status_Kunjungan_Peserta', 'Kode_Status_Kunjungan_Peserta']].value_counts())
```

Fungsi menampilkan distribusi baru ini untuk memahami bagaimana pengelompokan ulang memengaruhi data.

e. Mengidentifikasi kelas mayoritas dan minoritas

```
# Mengidentifikasi kelas mayoritas dan minoritas
majority_class_0 = selected_data[selected_data['Label'] == 0]  # Kelas mayoritas 0
majority_class_1 = selected_data[selected_data['Label'] == 1]  # Kelas mayoritas 1
minority_classes = selected_data[(selected_data['Label'] != 0) & (selected_data['Label'] != 1)]  # Semua kelas selain 0 dan 1 adalah minoritas
```

Label 0 yaitu ‘Berobat Jalan’, merupakan kelas mayoritas dengan jumlah data paling banyak. Label 1 juga dianggap mayoritas namun jumlahnya lebih sedikit dibandingkan dengan Label 0, yaitu ‘Rujuk Lanjut’. Sedangkan Label 2, 3, dan 4 adalah kelas data minoritas dengan jumlah data jauh lebih kecil.

f. Menentukan ukuran sampling

```
# Menentukan ukuran yang ingin diambil untuk mayoritas, sesuai dengan jumlah kelas minoritas
minority_count = len(minority_classes)
```

g. Undersampling kelas mayoritas

```
# Undersampling kelas mayoritas 0
majority_undersampled_0 = resample(majority_class_0,
```

```
                                replace=False,     # Tidak ada
        duplikasi
                                n_samples=22000,   #Sesuaikan
        jumlah sampel dengan kelas minoritas
                                random_state=42)


        # Undersampling kelas mayoritas 1
        majority_undersampled_1                    =
        resample(majority_class_1,
                                    replace=False,  # Tidak
        ada duplikasi
                                    n_samples=20000,    #
        Sesuaikan jumlah sampel dengan kelas minoritas
                                    random_state=42)
```

Fungsi 'resample' digunakan untuk melakukan sampling ulang untuk mengurangi jumlah data dari kelas mayoritas tanpa pengulangan (replace = False). 'n_samples' merupakan jumlah data yang akan diambil setelah undersampling, dan 'random_state' digunakan untuk memastikan hasil sampling dapat diproduksi.

h. Menggabungkan data

```
# Gabungkan kembali kelas mayoritas yang sudah diundersample
dengan kelas minoritas
selected_data = pd.concat([majority_undersampled_0,
majority_undersampled_1, minority_classes])
```

Kemudian, data dari kelas mayoritas (0 dan 1) digabung kembali dengan kelas minoritas untuk membentuk *dataset* baru dengan distribusi yang lebih seimbang.

i. Memeriksa distribusi baru

```
# Memeriksa distribusi data setelah undersampling
print(selected_data['Label'].value_counts())
```

Bagian ini memastikan bahwa jumlah data dari kelas mayoritas telah disesuaikan dengan kelas minoritas setelah proses undersampling.

3.1.3.5 Dataset Attribute Integration

Pada bagian ini, dilakukan pengecekan ulang data yang sudah diproses dan menyimpannya ke dalam file CSV untuk digunakan dalam analisis atau pelatihan model.

```
selected_data.info()
# Save the final dataset, after preparation


final_output_path = "../Dataset/Kontekstual DM/final_processed_data.csv"
selected_data.to_csv(final_output_path, index=False)
print(f"Dataset yang telah diproses dan lengkap disimpan ke
{final_output_path}.")
```

Data yang sudah dicek, disimpan dalam file CSV di Lokasi yang telah ditentukan dengan nama 'final_processed_data.csv'.

3.1.4 Modeling

Beberapa teknik *data mining* diterapkan untuk membangun model, dan mungkin model perlu diuji dan sesuaikan untuk mencapai hasil yang optimal. Adapun hal yang akan dilakukan pada tahapan ini:

a. Membangun Skenario Pengujian

   Menyusun skenario uji untuk menilai performa model, misalnya menggunakan data uji dan metrik evaluasi seperti *accuracy*, *precision*, *recall*, dan *F1 score*.

b. Membangun Model

   Menerapkan algoritma klasifikasi yang dipilih (XGBoost) dan melatih model menggunakan data latih yang sudah disiapkan.

3.1.4.1 Skenario Pengujian

Bagian ini bertujuan untuk mempersiapkan *dataset* dengan mempertimbangkan bobot setiap kelas untuk menangani ketidakseimbangan data, melakukan pembagian *dataset* menjadi tiga bagian yaitu, *train set* digunakan untuk melatih model, *validation set* digunakan untuk menyetel *hyperparameter* dan menghindari *overfitting*, dan *test set* untuk mengevaluasi performa akhir model. Normalisasi data dengan *StandardScaler* juga dilakukan.

3.1.4.1.1 Menambahkan Bobot pada Label

Bagian ini bertujuan untuk memberikan bobot lebih besar untuk kelas minoritas, sehingga model lebih memperhatikan kelas tersebut selama pelatihan.

```
def assign_weight(label):
    if label == 4:
        return 3
    elif label == 2:
        return 2
    else:
        return 1  # Kategori mayoritas


# Tambahkan kolom bobot ke DataFrame
df['Weight'] = df['Label'].apply(assign_weight)
```


p-ISSN xxxx–xxxx | e-ISSN xxxx–xxxx

Label 4 diberikan bobot 3 karena mewakili kelas minoritas yang signifikan, dan Label lainnya diberikan bobot 1, karena termasuk kelas mayoritas.

3.1.4.1.2 Memilih Kolom Fitur dan Target

Bagian ini memilih kolom yang digunakan sebagai fitur yang akan digunakan dan targetnya.

```
# Memilih atribut yang akan digunakan sebagai fitur
feature_columns = ['Kode_Provinsi',
                   'Kode_Jenis_Poliklinik',
                   'Kode_Diagnosis',
                   'Kode_Jenis_Kunjungan']

# Memisahkan fitur dan target
X = df[feature_columns]  # Hanya kolom fitur yang diinginkan
y = df['Label']          # Target adalah kolom Label
weights = df['Weight']
```

- ‘feature_columns’ digunakan untuk memilih kolom-kolom yang relevan sebagai fitur untuk pelatihan model.
- X berisikan dari dari kolom fitur yang diinginkan.
- Y berisikan kolom target yang akan diprediksi oleh model.
- ’weights’ untuk kolom bobot yang akan digunakan untuk menyeimbangkan pengaruh setiap kelas pada pelatihan model.

3.1.4.1.3 Membagi Dataset

Dataset dibagi menjadi *train set* (70%) digunakan untuk melatih model, *validation set* (15%) digunakan untuk menyetel *hyperparameter* dan menghindari *overfitting*, dan *test set* (15%) untuk evaluasi akhir model. *Weights* dibagi untuk memastikan distribusi bobot tetap seimbang antara *train* dan *test*.

```
# Pertama, bagi dataset menjadi train (70%) dan sisa (30%)
X_train, X_temp, y_train, y_temp, w_train, w_temp = train_test_split(
    X, y, weights, test_size=0.3, random_state=42
)

# Kemudian, bagi sisa (30%) menjadi validation (15%) dan test (15%)
X_val, X_test, y_val, y_test, w_val, w_test = train_test_split(
    X_temp, y_temp, w_temp, test_size=0.5, random_state=42
)
```

3.1.4.1.4 Normalisasi

Normalisasi perlu dilakukan untuk memastikan bahwa semua fitur memiliki skala yang sama untuk algoritma yang sensitif terhadap skala data, seperti *XGBoost*.

```
# Normalisasi data dengan StandardScaler
scaler = StandardScaler()

# Skala data
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)
```

StandardScaler digunakan untuk menstandarisasi data sehingga memiliki distribusi dengan rata-rata 0 dan standar deviasi 1. ‘fit_transform’ digunakan untuk menghitung rata-rata dan standar deviasi pada *train set*, lalu mengaplikasikan normalisasi, dan ‘transform’ digunakan untuk menerapkan normalisasi pada *validation* dan *test set*, menggunakan parameter yang sama dari *train set.*

3.1.4.1.5 Menghitung Total Bobot per Label

Bagian ini menghitung total bobot untuk setiap kelas pada *dataset* berdasarkan kolom ’weight’ dan memvisualisasikan menggunakan *bar plot*. Visualisasi ini membantu memeriksa apakah distribusi bobot sesuai untuk menangani ketidakseimbangan data, terutama jika kelas minoritas diberi bobot yang lebih tinggi.

```
# Mengalikan jumlah data dengan bobot per kelas
class_weight_df = df.groupby('Label').apply(lambda x: (x['Weight'] * x['Weight']).sum()).reset_index()
class_weight_df.columns = ['Class', 'Total Weight']

# Visualisasi distribusi total bobot per label
plt.figure(figsize=(10, 6))
sns.barplot(x='Class', y='Total Weight', data=class_weight_df, palette='viridis')

plt.title("Total Bobot berdasarkan Status Kunjungan Peserta")
plt.xlabel("Status Kunjungan Peserta")
plt.ylabel("Total Bobot")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

Data dikelompokkan berdasarkan ‘Label’, lalu total bobot dihitung dengan mengkuadratkan nilai bobot setiap baris dan menjumlahkannya.

3.1.4.2 Membangun Model (Healt-Sick Classification)

Model *XGBoost* dibangun untuk klasifikasi multi-kelas, mengoptimalkan *hyperparameter* menggunakan *GridSearchCV*, dan mempertimbangkan bobot untuk menangani ketidakseimbangan data.

3.1.4.2.1 Konversi Dataset ke Format Dmatrix

DMatrix adalah format khusus model *XGBoost* yang meningkatkan efisiensi pelatihan dan prediksi. *Parameter* 'weight' memungkinkan penanganan data yang tidak seimbang dengan memberikan bobot lebih tinggi untuk kelas minoritas. Data yang telah dipisahkan ke dalam *train*, *validation*, dan *test set* diubah menjadi format *DMatrix*.

```
# Konversi dataset ke format DMatrix (opsional untuk XGBoost)
dtrain = xgb.DMatrix(X_train_scaled, label=y_train, weight=w_train)
dval = xgb.DMatrix(X_val_scaled, label=y_val, weight=w_val)
dtest = xgb.DMatrix(X_test_scaled, label=y_test, weight=w_test)
```

Kolom 'Nama_Diagnosis' dan 'Jenis_Poliklinik', dikonversi menjadi tipe data `category`. Langkah ini dilakukan untuk meningkatkan efisiensi pengolahan data, khususnya dalam algoritma seperti *XGBoost* yang mendukung pengolahan data kategorikal secara langsung, sehingga pelatihan model menjadi lebih cepat dan efisien.

3.1.4.2.2 Menyusun Parameter Grid

Parameter grid adalah daftar nilai-nilai *parameter* yang diuji dalam *GridSearchCV*. *Parameter* yang diuji yaitu 'max_depth', 'learning_rate', dan 'n_estimators'. Variasi nilai parameter ini memungkinkan pencarian kombinasi terbaik untuk meningkatkan performa model.

```
# Parameter grid untuk pencarian hyperparameter
param_grid = {
    'model__max_depth': [4, 5, 6, 8],
    'model__learning_rate': [0.3, 0.2],
    'model__n_estimators': [200, 300]
}
```

3.1.4.2.3 Membuat Pipeline

Pipeline mengintegrasikan Langkah-langkah *pre-processing* dan model menjadi dalam satu alur kerja. Ada dua komponen yang digunakan yaitu *StandardScaler* untuk menormalkan data fitur agar memiliki distribusi seragam, dan model *XGBClassifier* utnuk melakukan klasifikasi multi-kelas.

```
# Membuat pipeline yang mencakup scaler dan model XGBoost
pipeline = Pipeline([
    ('scaler', StandardScaler()),  # Skala fitur
     ('model',  xgb.XGBClassifier(objective='multi:softprob',  num_class=5,
eval_metric='mlogloss', random_state=42))  # Model XGBoost
])
```

3.1.4.2.4 Membuat Scorer

Scorer adalah metrik evaluasi yang digunakan dalam *GridSearchCV* untuk menilai performa model. Metrik yang dipilih adalah *log loss* yang mengukur ketepatan probabilitas prediksi terhadap kelas target. Fungsi dirancang untuk mendukung *output* probabilitas, memastikan evaluasi akurat dalam klasifikasi multi-kelas.

```
# Tetapkan scorer log_loss untuk multi-class
scorer = make_scorer(
            lambda  y_true,  y_pred:  log_loss(y_true,  y_pred,
labels=np.unique(y_train)),
    greater_is_better=False, needs_proba=True
)
```

3.1.4.2.5 Cross-validation dengan StratifiedKFold

Bagian ini untuk memastikan bahwa setiap kelas mendapatkan representasi yang cukup dalam proses pelatihan dan validasi, terutama dalam kondisi *dataset* yang tidak seimbang.

```
# Gunakan StratifiedKFold untuk validasi silang
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
```

StratifiedKFold membagi data menjadi beberapa *folds* untuk validasi silang sambil mempertahankan distribusi kelas yang seimbang pada setiap *folds*.

3.1.4.2.6 Grid Search untuk Hyperparameter Tuning

GridSearchCV melakukan pencarian kombinasai parameter terbaik berdasarkan parameter grid yang telah ditentukan. Menggunakan *cross-validation* untuk menilai performa model.

```
# Menjalankan GridSearchCV
grid_search = GridSearchCV(
    estimator=pipeline,
    param_grid=param_grid,
    cv=cv,
    scoring=scorer,
    verbose=1,
    n_jobs=-1,
    error_score='raise'
)
```


p-ISSN xxxx–xxxx | e-ISSN xxxx–xxxx

Data akan dibagi menjadi lima *folds* untuk *cross-validation* dan memungkinkan penggunaan seluruh inti CPU untuk mempercepat proses.

3.1.4.2.7 Melatih Model dengan Bobot

Model dilatih dengan menggunakan *pipeline* dan *GridSearchCV,* dan menggunakan bobot kelas pada parameter 'model_sample_weight' untuk membantu dalam menangani ketidakseimbangan data dengan memberikan prioritas lebih tinggi pada kelas minoritas.

```
grid_search.fit(X_train, y_train, model__sample_weight=w_train)
```

3.1.4.2.8 Mendapatkan Hasil Terbaik

Kemudian, kombinasi parameter terbaik dan skor terbaik ditampilkan bersama. Skor ini akan menggambarkan performa model dengan *parameter* yang menghasilkan nilai *log loss* terendah selama *cross-validation.*

```
# Menampilkan hasil terbaik
print("Best Parameters:", grid_search.best_params_)
print("Best Score:", grid_search.best_score_)
```

Hasil yang diperoleh untuk *best parameters* yaitu:

1) Model_learning_rate: 0.2
   Dengan nilai 0.2, model belajar dalam langkah kecil untuk memperbarui bobot sehingga lebih stabil dan mengurangi resiko *overfitting*.
2) Model_max_depth: 4
   Parameter ini membatasi kedalaman maksimum yaitu sebesar 4. Dengan ini, model menjaga keseimbangan antara kompleksitas dan kemampuan generalisasi, mencegah *overfitting* pada data pelatihan.
3) Model_n_estimators: 200
   Parameter ini menentukan jumlah pohon dalam ensemble model yaitu 200. Dengan ini, model memiliki cukup kapasitas untuk menangkap pola kompleks dalam data tanpa terlalu berlebihan.

Hasil yang diperoleh untuk *best score* yaitu sebesar -0.68909. Nilai ini adalah nilai *log loss* terbaik yang dicapai model selama *cross-validation. Log loss* yang bernilai rendah menunjukkan prediksi model mendekati distribusi probabilitas kelas target yang sebenarnya.

3.1.4.2.9 Model Terbaik

'grid_search.best_estimator_' yaitu *pipeline* dengan konfigurasi parameter terbaik yang akan ditemukan selama proses *GridSearchCV* digunakan untuk memperoleh model terbaik.

```
best_model = grid_search.best_estimator_
```

Model ini digunakan untuk prediksi dan evaluasi lebih lanjut, baik pada *validation set* dan *test set*.

3.1.5 Evaluation

Fase ini mengevaluasi model untuk memastikan bahwa model telah memenuhi tujuan bisnis yang ditetapkan. Jika model belum memenuhi tujuan bisnis, maka perlu kembali ke fase sebelumnya untuk melakukan penyesuaian. Adapun beberapa hal yang dilakukan:

a. Mengevaluasi Hasil Pemodelan
   Mengevaluasi performa model dengan metrik yang relevan untuk memastikan model mampu memprediksi kunjungan sehat atau sakit dengan akurasi yang tinggi.
b. Melakukan Review Proses Pemodelan
   Meninjau ulang proses pemodelan untuk mencari kemungkinan peningkatan performa, misalnya dengan tuning *hyperparameter* atau mencoba algoritma lain.

```
# Prediksi pada data validasi
y_val_pred_prob = best_model.predict_proba(X_val)
y_val_pred = y_val_pred_prob.argmax(axis=1)

# Evaluasi pada data validasi
accuracy_val = accuracy_score(y_val, y_val_pred)
print("Validation Accuracy:", accuracy_val)

print("Validation Classification Report:")
print(classification_report(y_val, y_val_pred))

print("Validation Confusion Matrix:")
print(confusion_matrix(y_val, y_val_pred))
```

Model memprediksi probabilitas untuk setiap kelas menggunakan 'predict_proba', dan indeks dengan probabilitas tertinggi diambil sebagai prediksi kelas akhir menggunakan 'argmax'. Akurasi dihitung menggunakan *accuracy_score*, yaitu persentase prediksi yang benar dibandingkan dengan total data uji. *Classification Report* memberikan rincian *precision, recall, f1-score,* dan nilai *support* untuk setiap kelas, yang membantu menganalisis kekuatan dan kelemahan model pada tiap kategori. *Confusion Matrix* menunjukkan distribusi prediksi benar dan salah, dengan nilai diagonal utama mewakili prediksi

yang benar, sementara nilai di luar diagonal menunjukkan kesalahan klasifikasi.

```
# Prediksi pada data uji
y_pred_prob = grid_search.best_estimator_.predict_proba(X_test)

# Ambil indeks dengan probabilitas tertinggi sebagai prediksi akhir
y_pred = y_pred_prob.argmax(axis=1)

# Evaluasi performa
accuracy = accuracy_score(y_test, y_pred)
print("Akurasi:", accuracy)

print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

Proses dimulai dengan melakukan prediksi probabilitas untuk setiap kelas pada data uji `X_test` menggunakan model terbaik yang telah dihasilkan oleh grid search `grid_search.best_estimator_`)\. Metode `predict_proba` menghasilkan probabilitas untuk setiap kelas yang ada pada model. Kemudian, indeks dengan probabilitas tertinggi pada setiap sampel dipilih sebagai prediksi akhir menggunakan `argmax(axis=1)`, yang mengembalikan indeks kelas dengan nilai probabilitas tertinggi untuk setiap baris data. Akurasi dihitung menggunakan *accuracy_score*, yaitu persentase prediksi yang benar dibandingkan dengan total data uji. *Classification Report* memberikan rincian *precision, recall, f1-score,* dan nilai *support* untuk setiap kelas, yang membantu menganalisis kekuatan dan kelemahan model pada tiap kategori. *Confusion Matrix* menunjukkan distribusi prediksi benar dan salah, dengan nilai diagonal utama mewakili prediksi yang benar, sementara nilai di luar diagonal menunjukkan kesalahan klasifikasi.

3.1.6 Deployment

Fase ini mencakup pelaporan hasil, integrasi model ke sistem yang ada, atau pengembangan rencana untuk pemeliharaan model untuk kedepannya. Adapun hal yang dilakukan dalam fase ini:

a. Membuat Rencana Deployment Model
   Menyusun rencana untuk mengimplementasikan model ke dalam sistem produksi BPJS Kesehatan atau sistem informasi rumah sakit.

b. Melakukan Deployment Model
   Memasang model ke lingkungan produksi sehingga bisa digunakan untuk memprediksi jenis kunjungan secara otomatis pada data yand ada.

c. Melakukan Rencana Pemeliharaan
   Merencanakan pemeliharaan model, misalnya pemantauan performa model.

d. Melakukan Pemeliharaan
   Menjalankan pemeliharaan rutin pada model untuk memastikan akurasi dan performa tetap optimal.

3.1.6.1 Penyimpanan Model

Model *XGBoost* yang telah dilatih disimpan ke dalam file menggunakan *library dill* untuk dapat digunakan kembali tanpa perlu dilatih ulang. Hal ini dilakukan untuk memastikan model dapat dimuat ulang dengan mudah untuk digunakan dalam aplikasi produksi.

```
# Save the model
dill.dump(grid_search, open("mdl.pkl", "wb"))
```

File ‘mdl.pkl’ berisi model yang akan dipanggil dalam aplikasi *Flask*.

3.1.6.2 Pemanggilan dan Validasi Model

Model yang telah disimpan dimuat ke dalam aplikasi *Flask* yang diuji kembali untuk memastikan model dapat digunakan dengan tepat. Model juga dipastikan dapat dimuat tanpa eror dan dapat melakukan klasifikasi.

```
with open("mdl.pkl", "rb") as file:
    model = dill.load(file)
print("Model berhasil dimuat!")
# Lakukan tes sederhana
test_input = [[1, 2, 3, 4]]  # Sesuaikan dengan format input model Anda
prediction = model.predict(test_input)
print("Prediksi percobaan:", prediction)
```

Pengujian sederhana dilakukan untuk memberikan *input dummy* untuk memastikan klasifikasi bekerja.

3.1.6.3 Pembuatan Aplikasi Flask

Aplikasi dibuat dengan *Flask* untuk mengelola rute dan komunikasi antara pengguna dengan *backend*. Bagian ini juga bertujuan untuk memberikan antarmuka web yang memungkinkan pengguna mengakses fitur prediksi model.

```
# Buat aplikasi Flask
app = Flask(__name__)
```


p-ISSN xxxx–xxxx | e-ISSN xxxx–xxxx

```
# Global variables untuk model dan dataset
model = None
df = None

# Fungsi untuk memuat model
def load_model():
    global model
    try:
        with open("mdl.pkl", "rb") as file:
            model = dill.load(file)
        print("Model berhasil dimuat!")
    except FileNotFoundError:
        print("Model file 'mdl.pkl' tidak ditemukan. Pastikan file ada di direktori yang benar.")
        raise
    except Exception as e:
        print(f"Gagal memuat model: {e}")
        print(traceback.format_exc())
        raise

# Fungsi untuk memuat dataset
def load_dataset():
    global df
    try:
        df = pd.read_csv("../Dataset/Kontekstual DM/final_processed_data.csv")
        print("Dataset berhasil dimuat!")
    except FileNotFoundError:
        print("Dataset file tidak ditemukan. Pastikan file ada di direktori yang benar.")
        raise
    except Exception as e:
        print(f"Gagal memuat dataset: {e}")
        print(traceback.format_exc())
        raise
```

Rute *'home'* berfungsi untuk menampilkan halaman utama aplikasi dengan *form input* untuk klasifikasi. Data diambil dari *dataset* yang berisikan daftar provinsi, poliklinik, diagnosis, dan jenis kunjungan untuk ditampilkan dalam *dropdown* menu.

```
# Route untuk halaman utama
@app.route("/")
def home():
    # Ambil data untuk dropdown dari dataset
    provinsi_list = df['Nama_Provinsi'].unique()
    poliklinik_list = df['Jenis_Poliklinik'].unique()
    diagnosis_list = df['Nama_Diagnosis'].unique()
    jenis_kunjungan_list = df['Jenis_Kunjungan'].unique()

    # Kirim data ke template
    return render_template("index.html",
                           provinsi_list=provinsi_list,
                           poliklinik_list=poliklinik_list,
                           diagnosis_list=diagnosis_list,
                           jenis_kunjungan_list=jenis_kunjungan_list)
```

Rute *'predict'* berfungsi untuk menerima data input dari pengguna berupa provinsi, poliklinik, diagnosis, dan jenis kunjungan, yang kemudian memprosesnya menjadi format numerik, lalu memberikan klasifikasi berdasarkan model yang telah dimuat. Hasil yang diberikan berupa probabilitas setiap kelas dalam bentuk tabel.

```
# Route untuk prediksi
@app.route("/predict", methods=["POST"])
def predict():
    if model is None:
        return render_template("index.html", prediction_text="Model belum dimuat. Hubungi administrator.")

    try:
        # Ambil input dari user
        nama_provinsi = request.form.get('Nama_Provinsi')
        jenis_poliklinik = request.form.get('Jenis_Poliklinik')
        nama_diagnosis = request.form.get('Nama_Diagnosis')
        jenis_kunjungan = request.form.get('Jenis_Kunjungan')

        # Cek jika ada input yang kosong
        if not nama_provinsi or not jenis_poliklinik or not nama_diagnosis or not jenis_kunjungan:
            # Ambil data untuk dropdown agar tidak kosong
            provinsi_list = df['Nama_Provinsi'].unique()
            poliklinik_list = df['Jenis_Poliklinik'].unique()
            diagnosis_list = df['Nama_Diagnosis'].unique()
            jenis_kunjungan_list = df['Jenis_Kunjungan'].unique()

            return render_template("index.html", prediction_text="Semua kolom harus diisi!",
                                   Nama_Provinsi=nama_provinsi, Jenis_Poliklinik=jenis_poliklinik,
                                   Nama_Diagnosis=nama_diagnosis, Jenis_Kunjungan=jenis_kunjungan,
                                   provinsi_list=provinsi_list,
                                   poliklinik_list=poliklinik_list,
                                   diagnosis_list=diagnosis_list,
                                   jenis_kunjungan_list=jenis_kunjungan_list)

        # Cek dan konversi nama ke kode menggunakan dataset
        try:
            kode_provinsi = df.loc[df['Nama_Provinsi'] == nama_provinsi, 'Kode_Provinsi'].values[0]
            kode_poliklinik = df.loc[df['Jenis_Poliklinik'] == jenis_poliklinik, 'Kode_Jenis_Poliklinik'].values[0]
```

```
        kode_diagnosis = df.loc[df['Nama_Diagnosis'] == nama_diagnosis, 'Kode_Diagnosis'].values[0]
        kode_jenis_kunjungan = df.loc[df['Jenis_Kunjungan'] == jenis_kunjungan, 'Kode_Jenis_Kunjungan'].values[0]
    except IndexError:
        # Ambil data untuk dropdown agar tidak kosong
        provinsi_list = df['Nama_Provinsi'].unique()
        poliklinik_list = df['Jenis_Poliklinik'].unique()
        diagnosis_list = df['Nama_Diagnosis'].unique()
        jenis_kunjungan_list = df['Jenis_Kunjungan'].unique()

        return render_template("index.html", prediction_text="Data yang dimasukkan tidak ditemukan di dataset.",
                               Nama_Provinsi=nama_provinsi, Jenis_Poliklinik=jenis_poliklinik,
                               Nama_Diagnosis=nama_diagnosis, Jenis_Kunjungan=jenis_kunjungan,
                               provinsi_list=provinsi_list,
                               poliklinik_list=poliklinik_list,
                               diagnosis_list=diagnosis_list,
                               jenis_kunjungan_list=jenis_kunjungan_list)

    # Pastikan semua kode ditemukan
    input_features = [kode_provinsi, kode_poliklinik, kode_diagnosis, kode_jenis_kunjungan]

    # Konversi ke array numpy
    features = np.array([input_features])

    # Prediksi probabilitas
    probabilities = model.predict_proba(features)

    # Hitung jumlah pasien untuk setiap kelas (dalam bentuk prosentase)
    jumlah_pasien = {status_kunjungan_mapping[i]: round(probabilities[0][i] * 100) for i in range(len(probabilities[0]))}

    # Menyiapkan output prediksi dalam bentuk tabel
    return render_template("index.html", prediction_results=jumlah_pasien,
                           Nama_Provinsi=nama_provinsi, Jenis_Poliklinik=jenis_poliklinik,
                           Nama_Diagnosis=nama_diagnosis, Jenis_Kunjungan=jenis_kunjungan,
                           provinsi_list=df['Nama_Provinsi'].unique(),
                           poliklinik_list=df['Jenis_Poliklinik'].unique(),
                           diagnosis_list=df['Nama_Diagnosis'].unique(),
                           jenis_kunjungan_list=df['Jenis_Kunjungan'].unique())

  except Exception as e:
        return render_template("index.html", prediction_text=f"Terjadi kesalahan: {str(e)}")
```

3.1.6.4 Integrasi Dataset

Dataset yang relevan dimuat untuk mendukung fitur prediksi, seperti memetakan nama provinsi, jenis poliklinik, atau diagnosis ke dalam kode numerik untuk dapat dengan mudah dimengerti oleh model. Tujuannya untuk mendukung proses pengklasifikasian dengan menyediakan informasi tambahan yang relevan yang dibutuhkan oleh model.

```
df = pd.read_csv("../Dataset/Kontekstual DM/final_processed_data.csv")
```

3.1.6.5 Prediksi dan Penyajian Hasil

Prediksi dilakukan berdasarkan input yang diberikan pengguna, dan hasil prediksi disajikan dalam bentuk tabel atau teks di halaman web kepada pengguna.

```
# Prediksi probabilitas
    probabilities = model.predict_proba(features)

# Hitung jumlah pasien untuk setiap kelas (dalam bentuk prosentase)
    jumlah_pasien = {status_kunjungan_mapping[i]: round(probabilities[0][i] * 100) for i in range(len(probabilities[0]))}
```

3.1.6.6 Running Aplikasi

Aplikasi dijalankan di server lokal sehingga dapat diakses melalui browser. Tujuannya untuk membuka akses bagi pengguna untuk menggunakan aplikasi.

```
if __name__ == "__main__":
    app.run(debug=True, host='0.0.0.0', port=5000, use_reloader=False)
```

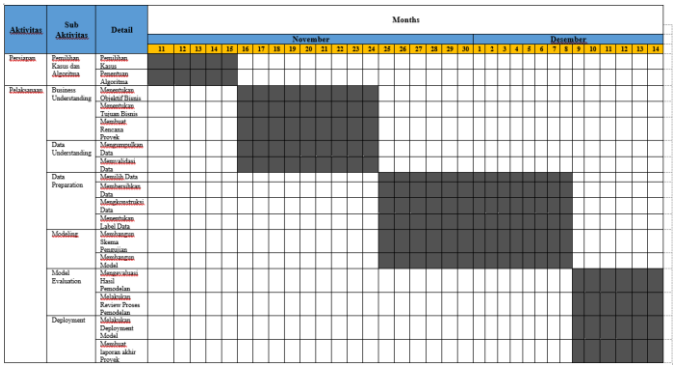
Deployment selesai, dan siap digunakan untuk melakukan pengklasifikasian berbasis model *machine learning*. Contoh tampilan halaman web yang digunakan pengguna untuk melakukan pengklasifikasian dapat dilihat pada Lampiran II.

3.2 Timeline

Berikut Tabel 3.1 yang berisikan alur kerja yang dilakukan dalam pengerjaan proyek ini yang dimulai dari pemahaman masalah hingga tahap *deployment* dan pelaporan.

p-ISSN xxxx–xxxx | e-ISSN xxxx–xxxx

Tabel 3.1 Timeline Prove



Berdasarkan *timeline* yang ditampilkan, berikut adalah penjelasan dari setiap aktivitas utama dalam proyek:

1. Identifikasi Masalah (Kasus dan Tujuan)

   Aktivitas ini berlangsung pada minggu kedua bulan November. Tahapan ini melibatkan identifikasi kasus dan tujuan proyek, yang bertujuan untuk memahami permasalahan yang akan diselesaikan melalui solusi berbasis model machine learning.

2. *Business Understanding*

   Pada tahapan ini, fokus utama adalah mendefinisikan kebutuhan bisnis dan tujuan spesifik yang akan dicapai. Aktivitas berlangsung selama pertengahan hingga akhir November. Proses ini penting untuk menyelaraskan kebutuhan bisnis dengan pengembangan model yang relevan.

3. *Data Understanding*

   Tahapan ini melibatkan eksplorasi *dataset* untuk mendapatkan wawasan awal terkait struktur data, distribusi, dan potensi tantangan seperti *missing values* atau ketidakseimbangan kelas. Aktivitas ini dilakukan pada minggu ketiga hingga keempat bulan November.

4. *Data Preparation*

   Tahapan ini fokus pada pembersihan, transformasi, dan pembobotan data untuk memastikan kualitas dataset yang optimal sebelum masuk ke pengembangan model. Aktivitas berlangsung dari akhir November hingga awal Desember, menunjukkan adanya perhatian khusus dalam menyiapkan data agar model bekerja secara optimal.

5. *Model Evaluation*

   Aktivitas ini mencakup pengujian model, optimasi *hyperparameter*, dan evaluasi kinerja menggunakan metrik yang relevan seperti *accuracy*, *f1-score*, atau *log loss*. Aktivitas ini berlangsung sepanjang minggu pertama hingga kedua bulan Desember.

6. *Deployment*

   Setelah model dievaluasi dan dioptimalkan, aktivitas *deployment* dilakukan pada pertengahan Desember. Proses ini bertujuan untuk menyimpan model dalam format yang dapat digunakan dalam produksi, seperti menyimpan model menggunakan library tertentu (misalnya, *dill*).

7. Laporan Akhir

   Penyusunan laporan proyek dilakukan bersamaan dengan tahap akhir aktivitas, yaitu pada minggu kedua Desember, setelah seluruh proses pengembangan model selesai. Tujuannya adalah mendokumentasikan keseluruhan proses, hasil, dan temuan dari proyek.

IV. HASIL DAN PEMBAHASAN

4.1 Hasil dan Pengujian Model

Pada bab ini, dijelaskan hasil pengujian yang dilakukan terhadap model yang telah dilakukan pada data yang telah diproses sebelumnya. Proses ini melibatkan pembagian *dataset* menjadi tiga bagian, yaitu *training set*, *validation set*, dan *test set*. Model XGBoost digunakan untuk melakukan prediksi kategori status kunjungan peserta. Hasil evaluasi akan dianalisis berdasarkan metrik evaluasi *accuracy*, *precision*, *recall*, *f1-score*, dan *confusion matrix*, seperti dalam Gambar 4.1.

```
Akurasi: 0.6498233758255261
Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.64      0.66      3301
           1       0.64      0.67      0.66      3019
           2       0.36      0.35      0.36       140
           3       1.00      1.00      1.00        21
           4       0.11      0.07      0.08        30

    accuracy                           0.65      6511
   macro avg       0.56      0.55      0.55      6511
weighted avg       0.65      0.65      0.65      6511

Confusion Matrix:
[[2124 1112   59    0    6]
 [ 959 2035   18    0    7]
 [  66   22   49    0    3]
 [   0    0    0   21    0]
 [  14    4   10    0    2]]
```

Gambar 4.13 Hasil Evaluasi Model

Model yang dilatih dan diuji menggunakan data uji memberikan hasil pengujian yang menunjukkan bahwa model memiliki performa yang cukup baik. Model menghasilkan akurasi sebesar 64.98%, yang menunjukkan bahwa hampir 65% dari prediksi model disesuaikan dengan label sebenarnya.

Berdasarkan *classification report*, nilai *precision* dan *recall* cukup baik untuk kelas 0 dan 1, yang masing-masing memiliki rata-rata *f1-score* sekitar 0.66. Hal ini merupakan pendekatan yang cukup baik karena setidaknya model mampu menklasifikasi class utama dalam pengklasifikasian ini. Namun, performa model untuk kelas 2 dan 4 lebih rendah, dengan *f1-score* untuk kelas 2 hanya sebesar 0.36, sementara untuk kelas 4 hanya mencapai 0.08. Ini menunjukkan bahwa model kesulitan dalam mengklasifikasikan kelas minoritas, yang didukung oleh distribusi di *confusion matrix*. Contohnya, dalam kelas 2, hanya 66 sampel yang diprediksi benar dari total 140, dan dalam kelas 4, hanya 2 sampel yang diprediksi benar dari total 30. Untuk label 3, memiliki nilai metrik evaluasi sebesar 1 karena kelas ini memiliki fitur yang sangat mendukung pelatihannya diantara kelas lainnya. Nilai rata-rata (macro avg) untuk precision, recall, dan f1-score sekitar 0.55, yang mencerminkan ketidakseimbangan performa antar kelas.

4.2 Interpretasi Hasil

Model yang diimplementasikan memiliki akurasi keseluruhan cukup baik di angka 64.98%. Namun, perlu langkah-langkah tambahan, seperti *oversampling*, *cost-sensitive learning*, atau pengembangan model khusus untuk data imbalan, guna meningkatkan kemampuan model dalam mengklasifikasikan kelas minoritas. Akurasi keseluruhan bukan metrik yang mencerminkan performa model sebenarnya dalam kasus ini.

Proyek ini berhasil membangun pipeline yang lengkap untuk klasifikasi multikelas pada data dengan ketidakseimbangan kelas. Meski hasil model masih belum optimal untuk kelas minoritas, proyek ini memberikan landasan yang kuat untuk pengembangan lebih lanjut, seperti penerapan teknik *oversampling* pada kelas minoritas, penggunaan algoritma lain seperti *cost-sensitive learning*, atau eksplorasi arsitektur model yang lebih kompleks.

V. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Sistem ini dikembangkan untuk melakukan klasifikasi pada jenis kunjungan pasien berdasarkan *input* yang diberikan, seperti provinsi, jenis poliklinik, diagnosis, dan jenis kunjungan. Model *machine learning* yang digunakan untuk prediksi ini dilatih dengan dataset yang relevan dan disimpan dalam file `mdl.pkl` menggunakan pustaka `dill`. Setelah model berhasil dilatih, aplikasi web dibangun dengan menggunakan *framework Flask* untuk menyediakan antarmuka yang interaktif. Dataset yang mencakup informasi tentang provinsi, jenis poliklinik, diagnosis, dan jenis kunjungan juga dimuat ke dalam aplikasi, sehingga memungkinkan pengguna untuk memilih data yang relevan dari *dropdown* menu yang tersedia.

Pada halaman utama aplikasi, pengguna dapat memilih provinsi, jenis poliklinik, diagnosis, dan jenis kunjungan. Setelah itu, pengguna dapat mengklik tombol 'Prediksi' untuk memproses input dan menghasilkan prediksi jumlah kunjungan untuk setiap jenis kunjungan yang tersedia. Sistem akan mengonversi data input pengguna menjadi kode numerik, kemudian menggunakan model untuk menghitung probabilitas setiap kategori kunjungan, dan hasilnya ditampilkan dalam bentuk tabel yang menunjukkan estimasi jumlah kunjungan untuk kategori seperti 'Berobat Jalan', 'Rujuk Lanjut', 'Sembuh', dan lainnya. Hasil ini memberikan gambaran yang jelas dan informatif mengenai distribusi pasien, yang dapat membantu pihak rumah sakit atau fasilitas kesehatan dalam merencanakan sumber daya dan pengelolaan pasien. Aplikasi ini memiliki potensi untuk diintegrasikan lebih lanjut, memberikan nilai lebih dalam pengelolaan data BPJS dan analisis kesehatan di masa depan.

5.2 Saran

Saran untuk meningkatkan sistem yang telah dibuat:

1. Untuk meningkatkan pengalaman pengguna, pengguna bisa diberikan pesan kesalahan yang lebih terperinci jika terjadi masalah pada saat prediksi. Misalnya, jika model gagal memuat, dapat diberikan instruksi lebih jelas mengenai langkah yang perlu diambil (seperti menghubungi administrator).
2. Jika aplikasi ini digunakan secara aktif oleh banyak pengguna, pertimbangkan untuk mengoptimalkan kinerja model dengan memanfaatkan teknik pemrosesan *batch* atau menggunakan model yang lebih ringan jika diperlukan.

p-ISSN xxxx–xxxx | e-ISSN xxxx–xxxx

3. Bisa mempertimbangkan untuk meningkatkan desain antarmuka pengguna (UI) untuk membuatnya lebih responsif dan menarik.
4. Jika aplikasi ini akan diakses oleh banyak pengguna atau diimplementasikan dalam skala besar, penting untuk menambahkan lapisan keamanan seperti autentikasi pengguna, enkripsi data, atau proteksi terhadap serangan seperti *SQL injection*.

VI. PEMBAGIAN PEKERJAAN

Anggota Kelompok	Bagian yang Dikerjakan
Naomi Elena Lumbanraja	Business Understanding, Data Preparation, dan Evalution Model
Karina Checilia Situmorang	Data Overview dan Data Understanding
Rebecca Yulyarta Bulawan Sihombing	Data Understanding, Modelling, dan Deployment
Tio Manalu	Data Overview dan Data Understanding

REFERENSI

[1] E. K. Astuti, "Peran BPJS Kesehatan Dalam Mewujudkan Hak Atas Pelayanan Kesehatan Bagi Warga Negara Indonesia," *JPeHI J. Penelit. Huk. Indones.*, vol. 01, no. 01, pp. 55–65, 2020, [Online]. Available: https://core.ac.uk/download/pdf/322612246.pdf.

[2] Solechan, "Badan Penyelenggara Jaminan Sosial (BPJS) Kesehatan," *Adm. Law Gov. J.*, vol. 2, no. 4, pp. 686–696, 2019, [Online]. Available: https://doi.org/10.14710/alj.v2i4.686-696.

[3] W. Afifah, "Perlindungan_Hukum_Hak_Kesehatan_Warga_N," no. November, pp. 150–169, 2015.

[4] Novijan Janis, "BPJS Kesehatan, Supply, dan Demand Terhadap Layanan Kesehatan," pp. 1–8, 2019.

[5] A. D. Sugiharto, S. Hidayat, and R. Rosyidah, "Pengaruh Kualitas Pelayanan Dan Kepuasan Pasien Terharap Loyalitas : Analisis Di Sebuah Fasilitas Kesehatan Tingkat Pertama (FKTP) Program Jaminan Kesehatan Nasional (JKN)," *An-Nadaa J. Kesehat. Masy.*, vol. 10, no. 2, p. 118, 2023, doi: 10.31602/ann.v10i2.10455.

[6] BPJS Kesehatan, "Data Sampel BPJS Kesehatan 2015-2021," p. 85, 2022.

[7] N. Uzir, S. Raman, S. Banerjee, and R. S. Nishant Uzir Sunil R, "Experimenting XGBoost Algorithm for Prediction and Classification of Different Datasets Experimenting XGBoost Algorithm for Prediction and Classifi cation of Different Datasets," *Int. J. Control Theory Appl.*, vol. 9, no. July, 2016, [Online]. Available: https://www.researchgate.net/publication/318132203.

[8] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, vol. 13-17-Augu, pp. 785–794, 2016, doi: 10.1145/2939672.2939785.

[9] R. Wirth and J. Hipp, "CRISP-DM: towards a standard process model for data mining. Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining, 29-39," *Proc. Fourth Int. Conf. Pract. Appl. Knowl. Discov. Data Min.*, no. 24959, pp. 29–39, 2000, [Online]. Available: https://www.researchgate.net/publication/239585378_CRISP-DM_Towards_a_standard_process_model_for_data_mining.

LAMPIRAN

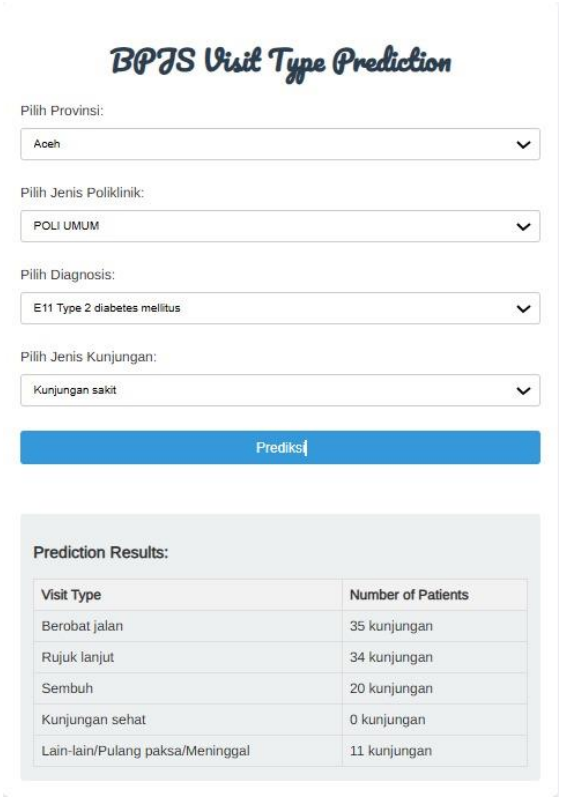
Lampiran I

Penamaan File Data Pelayanan FKTP.

No.	Variabel	Label Variabel	Deskripsi
1.	PSTV01	Nomor peserta	Nomor identifikasi peserta yang bersifat unik dan telah dideidentifikasi untuk melindungi identitas peserta sebenarnya
2.	PSTV02	Nomor keluarga	Nomor yang mengidentifikasi kepala keluarga dalam sampel dan berfungsi sebagai penanda keluarga (peserta BPJS Kesehatan dalam satu keluarga memiliki nomor kepala keluarga yang sama)
3.	PSTV15	Bobot	Faktor pengali yang menggambarkan jumlah individu di dalam populasi diwakili oleh individu di dalam sampel
4.	FKP02	ID Kunjungan FKTP	Nomor identifikasi unik untuk menandakan setiap kunjungan FKTP oleh peserta
5.	FKP03	Tanggal datang kunjungan FKTP	Tanggal melakukan kunjungan FKTP
6.	FKP04	Tanggal pulang kunjungan FKTP	Tanggal menyelesaikan kunjungan FKTP
7.	FKP05	Provinsi FKTP	Provinsi tempat peserta mengakses fasilitas kesehatan (sesuai kode wilayah BPS)
8.	FKP06	Kabupaten/Kota FKTP K	Kabupaten/kota tempat peserta mengakses fasilitas kesehatan (sesuai kode wilayah BPS)
9.	FKP07	Kepemilikan FKTP	Jenis kepemilikan fasilitas kesehatan yang diakses oleh peserta pada kunjungan FKTP
10.	FKP08	Jenis FKTP	Jenis fasilitas kesehatan yang diakses oleh peserta pada kunjungan FKTP
11.	FKP09	Tipe FKTP	Tipe faskes yang diakses oleh peserta pada kunjungan FKTP
12.	FKP10	Tingkat Pelayanan FKTP	Tingkat layanan yang diakses oleh peserta pada kunjungan FKTP
13.	FKP11	Jenis Poli FKTP	Jenis poliklinik yang diakses oleh peserta pada kunjungan FKTP
14.	FKP12	Segmen Peserta saat	Jenis segmen peserta saat peserta mengakses FKTP

		akses layanan FKTP	
15.	FKP13	Status Pulang peserta	Status peserta di akhir masa perawatan
16.	FKP14	Kode dan nama diagnosis ICD 10 (3 digit)	Kode dan nama diagnosis berdasarkan 3 digit pertama kode ICD 10 yang diperoleh dari hasil input sistem informasi BPJS Kesehatan
17.	FKP14A	Kode diagnosis ICD 10 (3 digit)	
18.	FKP15	Kode diagnosis (beragam 3-5 digit)	Kode diagnosis hasil input di sistem BPJS Kesehatan berdasarkan kode ICD 10 (jumlah digit beragam pada semua observasi dengan rentang 3-5 digit)
19.	FKP15A	Nama diagnosis berasal dari kode diagnosis	Nama diagnosis yang terbaca oleh sistem BPJS Kesehatan berdasarkan kode diagnosis (FKP15) yang ter-input dalam sistem.
20.	FKP16	Provinsi faskes tujuan rujukan	Provinsi fasilitas kesehatan tujuan rujukan (sesuai kode wilayah BPS)
21.	FKP17	Kabupaten/Kota faskes tujuan rujukan	Kabupaten/kota fasilitas kesehatan tujuan rujukan (sesuai kode wilayah BPS)
22.	FKP18	Kepemilikan faskes tujuan rujukan	Kepemilikan fasilitas kesehatan tujuan rujukan
23.	FKP19	Jenis faskes tujuan rujukan	Jenis fasilitas kesehatan tujuan rujukan
24.	FKP20	Tipe faskes tujuan rujukan	Tipe fasilitas kesehatan tujuan rujukan
25.	FKP21	Poli faskes tujuan rujukan	Poli fasilitas kesehatan tujuan rujukan
26.	FKP22	Jenis Kunjungan FKTP	Jenis kunjungan FKTP

Lampiran II



p-ISSN xxxx–xxxx | e-ISSN xxxx–xxxx