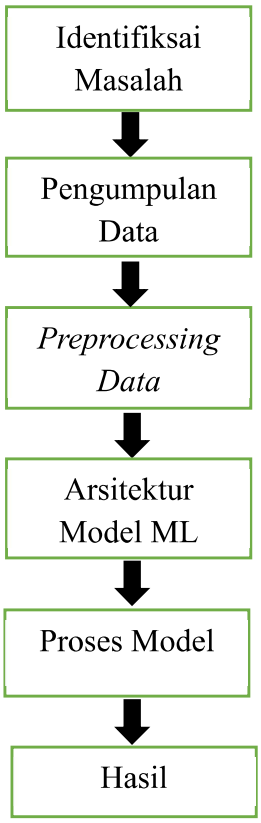
BAB III

METODE PENELITIAN

3.1 Desain Penelitian

Dari penelitian yang dilakukan, adapun desain penelitian sebagai berikut :



Gambar 3. 1 Desain Penelitian

Penjelasan dari gambar desain penelitian ini adalah :

1. Identifikasi Masalah

   Tujuan tahap awal ini adalah untuk menemukan masalah klinis, seperti cara memprediksi keberhasilan implan gigi berdasarkan informasi pasien.

Problem ini menjadi dasar penerapan pembelajaran mesin untuk membantu pengambilan keputusan klinis.

2. Pengumpulan Data

   Data ini dikumpulkan dari rekam medis pasien di Klinik Ellisa Dental, yang mencakup informasi berikut Usia pasien, jenis kelamin, riwayat medis (seperti diabetes, osteoporosis), kebiasaan merokok, kondisi tulang rahang, dan keberhasilan atau kegagalan implan (label atau target).

3. *Preprocessing Data*

   Agar algoritma pembelajaran mesin dapat digunakan, data klinik harus dibersihkan dan diproses. Ada beberapa langkah dalam proses ini, seperti menghapus data yang tidak lengkap atau duplikat, menormalkan nilai numerik, mengkodekan variabel kategorikal, seperti jenis kelamin, dan membagi data menjadi set pelatihan dan pengujian.

4. Arsitektur Model *Machine learning* (ML)

   Di Klinik Ellisa Dental, metode *Decision Tree* digunakan sebagai algoritma utama untuk membuat model prediksi keberhasilan implan gigi. Algoritma klasifikasi yang dikenal sebagai *Decision Tree* membagi dataset ke dalam berbagai cabang berdasarkan nilai-nilai fiturnya.

5. Proses Model

   Model dilatih menggunakan data latih. Proses ini termasuk: Pelatihan: melatih model dengan data pasien Validasi: menguji kinerja awal hyperparameter Pengaturan: mengatur parameter agar kinerja optimal.

6. Hasil, data uji digunakan untuk mengevaluasi model yang telah dilatih. Hasil evaluasi menggunakan metrik seperti Akurasi *Precision Recall* F1-*score Confusion Matrix* memberikan gambaran seberapa baik model dapat memprediksi keberhasilan implan gigi pada pasien baru.

3.2 Metode Pengumpulan Data

Penelitian ini mengumpulkan informasi demografis dan klinis tentang pasien di Klinik Ellisa Dental yang menerima tindakan implan gigi. Data yang dikumpulkan digunakan sebagai dasar untuk membuat dan melatih model pembelajaran mesin untuk memprediksi keberhasilan tindakan implan gigi.

1. Jenis Data, Data sekunder yang digunakan dalam penelitian ini berasal dari rekam medis digital Klinik Ellisa Dental, yang berisi informasi historis tentang pasien yang telah menjalani prosedur implan gigi.
2. Metode untuk Pengumpulan Data, Teknik dokumentasi digunakan untuk mengumpulkan data pasien dari sistem rekam medis elektronik (EMR) klinik yang ada. Pengumpulan data ini dilakukan dengan izin dan pengawasan klinik serta dengan memperhatikan prinsip kerahasiaan dan etika penelitian.
3. Variabel yang Dikumpulkan, Beberapa variabel penting yang dikumpulkan dari data pasien adalah sebagai berikut: Data Demografis: Usia, jenis kelamin, Data Klinis: Status kesehatan umum seperti diabetes dan hipertensi, kepadatan tulang rahang jika tersedia dalam hasil radiografi atau CT-scan, kebiasaan merokok, lokasi dan jumlah implan yang diberikan dan jenis implan yang digunakan.

4. Kriteria inklusi dan eksklusi adalah sebagai berikut: Pasien yang menjalani implan gigi dan memiliki semua data medis yang relevanci, pasien yang telah melewati masa evaluasi pasca-implan, seperti enam bulan. Pasien yang memiliki data yang tidak lengkap atau tidak ada, pasien yang tindakan implannya belum dievaluasi untuk hasil akhir dan pasien yang tindakan implannya belum dievaluasi secara menyeluruh.

3.3 Tahapan *Preprocessing* Data

Karena kualitas data yang baik akan meningkatkan kinerja model, preprocessing data adalah tahap penting dalam pengolahan data *machine learning*. Sebelum digunakan dalam pelatihan model, data dari rekam medis digital Klinik Ellisa Dental akan diproses.

Berikut ini adalah langkah-langkah *preprocessing* data:

1. Pembersihan Data (*Cleaning Data*), Menghapus semua data yang tidak perlu atau tidak relevan. Merawat nilai kosong atau tidak ada: Jika jumlah nilai kosong sedikit, data akan dihapus. Imputati dilakukan jika signifikan, dengan nilai rata-rata, median, atau modus (tergantung pada tipe data). Koreksi kesalahan input seperti penulisan yang salah tentang usia, kode kelamin, atau status keberhasilan.
2. Usia, kondisi kesehatan, lokasi implan, dan kebiasaan merokok adalah fitur penting yang memengaruhi keberhasilan implan gigi. Menggunakan *encoding* (seperti *Label Encoding* atau *One-Hot Encoding*) untuk mengubah fitur kategorikal seperti jenis kelamin atau status merokok. Untuk menjaga skala yang sama, fitur seperti usia atau kepadatan tulang

dinormalisasi dengan menggunakan metode *Min-Max Scaling* atau *Standardization*.

3. Data set dari penelitian berikut:

   Dataset dalam penelitian ini berasal dari data rekam medis digital pasien di Klinik Ellisa Dental yang telah menjalani prosedur implan gigi. Dataset ini termasuk ke dalam jenis data sekunder yang berisi informasi klinis penting dan mendetail, yang digunakan untuk membangun model prediksi kelayakan pemasangan implan menggunakan pendekatan *machine learning*, khususnya algoritma *Decision Tree.* Secara keseluruhan, dataset mencakup 20 *variabel input* (fitur) dan 1 *variabel output* (target), yaitu status kelayakan pemasangan implan (Layak Implan: Ya/Tidak). Atribut-atribut tersebut mencakup informasi demografis, kondisi kesehatan pasien, hasil pemeriksaan radiologi, serta sejumlah faktor risiko lain yang berpotensi memengaruhi keberhasilan prosedur implan. Berikut beberapa kategori dataset yang akan membantu penelitian dalam mengambil keputusan kelayakan implan gigi.

Tabel 3. 1 Atribut data pasien

NO	Nama Atribut	Tipe Data	Deskripsi
1	No RM	Numerik	No rekam medis pasien
2	Nama	Teks	Nama pasien
3	Usia	Numerik	Umur pasien dalam tahun
4	Jenis kelamin	Kategorikal	(laki laki/perempuan)

5	Status merokok	Kategorikal	Berat/ringan/tidak
6	Diabetes	Kategorikal	Tidak/terkontrol/tidak terkontrol
7	Hipertensi	Kategorikal	Tidak/terkontrol/tidak terkontrol
8	Osteoporosis	Kategorikal	Ya/Tidak
9	Terapi Bisfosfonat	Kategorikal	Ya/Tidak
10	Penyakit Autoimun	Kategorikal	Ya/Tidak
11	Terapi Imunosupresif	Kategorikal	Ya/Tidak
12	Status Kebersihan Mulut	Kategorikal	Baik/Sedang/Buruk
13	Periodontitis Aktif	Kategorikal	Ya/Tidak
14	Jumlah Gigi Hilang	Numerik	Jumlah gigi yang hilang
15	Lokasi Gigi Hilang	Numerik	Anterior, Posterior, dsb
16	Fraktur atau Trauma	Kategorikal	Ya/Tidak
17	Riwayat Radiasi Mandibula	Kategorikal	Ya/Tidak
18	Konsistensi Tulang (CBCT)	Kategorikal	D1-D4
19	Kondisi tulang rahang	Kategorikal	Cukup/Kurang/Parah/Baik
20	Layak Implan	kategorikal	Ya/Tidak

3.4 Arsitektur Model *Machine Learning*

Klinik Ellisa Dental menggunakan metode *Decision Tree* untuk membuat model pembelajaran mesin untuk prediksi implan gigi. Tujuan dari model ini adalah untuk membantu dokter menentukan kelayakan pasien untuk menjalani prosedur implan gigi berdasarkan data medis seperti usia, kondisi gusi, jumlah gigi yang hilang, struktur tulang rahang, dan riwayat penyakit sistemik. Proses *preprocessing* yang akan dilakukan pada data pasien termasuk pembersihan data, *encoding* variabel kategorikal, dan seleksi fitur. Selanjutnya, data dibagi menjadi data uji dan data latih. Kemudian, algoritma *Decision Tree* menggunakan kriteria pemisahan seperti *entropy* atau Gini Index untuk melakukan latihan pada masing-masing.

Hasil dari model ini mencakup prediksi tentang kesesuaian pasien untuk implan gigi, serta alasan yang mendukung untuk struktur pohon keputusan yang telah terbentuk. Model dievaluasi dengan metrik seperti akurasi, ketepatan, dan *matrix confusion*. Dokter dapat memahami dasar dari hasil prediksi dengan melihat pohon keputusan, sehingga lebih mudah untuk menjelaskan keputusan kepada pasien. Diharapkan bahwa penerapan model ini akan mempercepat diagnosis, mempercepat proses pengambilan keputusan medis, dan meningkatkan kepercayaan pasien terhadap layanan klinik berbasis data.

Dalam penelitian ini, data pasien digunakan, yang mencakup dua puluh variabel penting yang memainkan peran penting dalam menentukan kelayakan implantasi gigi. Identitas dasar, seperti nomor rekam medis, nama, dan usia, serta informasi demografis, seperti jenis kelamin dan status merokok, dan data klinis, seperti riwayat penyakit sistemik seperti diabetes, hipertensi, *osteoporosis*, dan

penyakit autoimun, termasuk dalam variabel-variabel tersebut. Faktor tambahan juga dipertimbangkan, seperti kondisi tulang rahang, kondisi kebersihan mulut, adanya *periodontitis* aktif, jumlah dan lokasi gigi yang hilang, riwayat fraktur atau trauma, paparan radiasi pada *mandibula*, dan hasil pemeriksaan konsistensi tulang CBCT. Untuk klasifikasi, algoritma *Decision Tree* digunakan untuk memasukkan semua data dari catatan medis. Status kelayakan implan pasien adalah variabel target.

3.5 Dataset Implan Gigi

Sebelum menggabungkan dua dataset, langkah pertama adalah memastikan bahwa kedua file memiliki struktur kolom, jenis data, dan korelasi antara fitur dan atribut yang sama. Peneliti menggunakan *JupyterLab* untuk mengimport file csv serangan dan normal. Mereka juga menggunakan *library pandas, numpy*, *matplotlib.pyplot*, dan *seaborn* untuk menganalisis kedua file tersebut. Pertama, peneliti melihat apakah ada perbedaan antara tipe data dan kolom yang perlu diubah. Mereka membandingkan kedua dataset dengan menggunakan perintah *if-else*; jika tipe data dan kolomnya sama, maka outputnya sama, jika tidak, maka outputnya tidak sama. Jumlah dataset dalam penelitian ini adalah 297 dan jumlah variabelnya 19, adapun dataset yang akan diolah dalam penelitian ini adalah:

Tabel 3. 2 *Sample DataSet*

	No RM	nama	Usia	Jenis Kelamin	Status Merokok	Diabetes	Hipertensi	Osteoporosis	Te
1	1	Salma raka	48	Pria	Perokok berat	Tidak	Terkontrol	Tidak	
2	2	Bintang arianto pardede	70	Pria	Perokok ringan	Tidak	Terkontrol	Ya	
3	3	welman	54	Pria	Perokok berat	Tidak terkontrol	Tidak	Tidak	
4	4	Santo	37	Pria	Perokok berat	Terkontrol	Tidak terkontrol	Tidak	
5	5	Afika lista pangaribuan	29	Wanita	Perokok ringan	Tidak terkontrol	Tidak terkontrol	Ya	
6	6	Lesnan pria	57	Pria	Perokok berat	Terkontrol	Tidak terkontrol	Tidak	
7	7	Hotmartua sitorus	56	Pria	Tidak	Tidak terkontrol	Terkontrol	Tidak	
8	8	Muhammad abidin	33	Pria	Perokok ringan	Tidak terkontrol	Tidak	Tidak	
9	9	Tika lestari	32	Wanita	Tidak	Tidak terkontrol	Terkontrol	Tidak	
10	10	bianca	49	Wanita	Perokok berat	Tidak terkontrol	Tidak	Tidak	
11	11	Arianti sinaga	57	Wanita	Perokok ringan	Tidak terkontrol	Terkontrol	Tidak	
12	12	Benjuma suryo	55	Pria	Perokok berat	Terkontrol	Tidak terkontrol	Ya	
13	13	Banje hita tamba	24	Wanita	Perokok ringan	Tidak terkontrol	Tidak	Ya	
14	14	Novita	44	Wanita	Tidak	Tidak terkontrol	Terkontrol	Tidak	
15	15	Ganda nasution	46	Pria	Tidak	Terkontrol	Terkontrol	Ya	
16	16	Bela ima	49	Wanita	Perokok ringan	Tidak terkontrol	Tidak terkontrol	Tidak	
17	17	Imanuel	39	Pria	Tidak	Tidak terkontrol	Tidak	Tidak	
18	18	Wika oki andela	45	Wanita	Tidak	Tidak terkontrol	Terkontrol	Ya	
19	19	niczal	67	Pria	Perokok berat	Terkontrol	Tidak	Tidak	
20	20	Ita lestari	75	Wanita	Perokok ringan	Tidak terkontrol	Terkontrol	Tidak	
21	21	Sitio manalu	53	Wanita	Perokok berat	Tidak	Terkontrol	Tidak	
22	22	Hetri eka wau	71	Wanita	Perokok berat	Terkontrol	Tidak terkontrol	Tidak	
23	23	Pirhot panggabean	43	Pria	Tidak	Terkontrol	Tidak	Tidak	

Dilanjutkan dengan perintah *df.info()* pada pustaka pandas di *Python* memberikan ringkasan struktur *DataFrame*, yang menunjukkan bahwa objek

tersebut adalah sebuah *DataFrame* dengan 297 baris (*entry*) dan 20 kolom, serta menggunakan *RangeIndex* dari 0 hingga 296. Ringkasan ini berguna untuk memahami ukuran data, tipe indeks, dan jumlah kolom sebelum melakukan analisis lebih lanjut atau pembersihan data seperti pada hasil berikut:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 297 entries, 0 to 296
Data columns (total 20 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   No RM                     297 non-null    int64
 1   nama                      297 non-null    object
 2   Usia                      297 non-null    int64
 3   Jenis Kelamin             297 non-null    object
 4   Status Merokok            297 non-null    object
 5   Diabetes                  297 non-null    object
 6   Hipertensi                297 non-null    object
 7   Osteoporosis              297 non-null    object
 8   Terapi Bisfosfonat        297 non-null    object
 9   Penyakit Autoimun         297 non-null    object
 10  Terapi Imunosupresif      297 non-null    object
 11  Kondisi Tulang Rahang     297 non-null    object
 12  Status Kebersihan Mulut   297 non-null    object
 13  Periodontitis Aktif       297 non-null    object
 14  Jumlah Gigi Hilang        297 non-null    int64
 15  Lokasi Gigi Hilang        297 non-null    object
 16  Fraktur atau Trauma       297 non-null    object
 17  Riwayat Radiasi Mandibula 297 non-null    object
 18  Konsistensi Tulang (CBCT) 297 non-null    object
 19  Layak Implan              297 non-null    object
dtypes: int64(3), object(17)
memory usage: 46.5+ KB
None
```

Gambar 3. 1 *Dataframe*

3.6 Proses Model Algoritma *Decition Tree*

Mulai dengan pengumpulan data dari klinik seperti Ellisa Dental, algoritma *Decision Tree* digunakan untuk membuat model machine learning untuk prediksi implan gigi. Data ini mencakup usia, jenis kelamin, kondisi tulang rahang,

kebiasaan merokok, kesehatan gusi, riwayat penyakit, dan apakah implan berhasil atau tidak. Setelah itu, proses pra-pemrosesan dilakukan untuk memproses data; ini termasuk membersihkan nilai kosong, mengubah data kategorikal menjadi data numerik, dan memilih fitur yang relevan. Data kemudian dibagi menjadi dua bagian: data latihan dan data uji. Ini dilakukan dengan menggunakan metode train_test_split. Kemudian, *Decision Tree Classifier* dari scikit-learn digunakan untuk membangun model *Decision Tree*. Algoritma belajar membuat aturan klasifikasi berdasarkan pola dari data latih. Untuk menilai performanya model yang telah dilatih, data uji digunakan untuk menilai akurasi, *precision, recall*, dan *confusion matrix*. Untuk memudahkan interpretasi dokter atau analis, hasil model dapat divisualisasikan dalam bentuk diagram pohon dengan menggunakan plot_tree(). Terakhir, model digunakan pada data pasien baru dan memiliki kemampuan untuk divalidasi ulang melalui metode *cross-validation* untuk menjamin keandalannya dalam prediksi dunia nyata.

Tahapan awal dalam melakukan proses-proses atau langkah-langkahnya adalah:

3.6.1 *Data Loading*

Muat kumpulan data **"Dataset_Implan_Gigi.csv"** ke dalam pandas *DataFrame*. Tampilkan informasi setelah memasukkan kumpulan data ke dalam pandas *DataFrame*.

Program ini digunakan untuk membaca file CSV bernama Dataset_Implan_Gigi.csv menggunakan pustaka pandas dengan *encoding* 'latin-1', dan menampilkan isi awal file dan informasi struktur datanya melalui *df.head*() dan

df.info. Program ini memiliki blok *try-except* untuk menangani berbagai kesalahan membaca file yang mungkin terjadi. Jika file tidak ditemukan, akan muncul pesan "Kesalahan: 'Dataset_Implan_Gigi.csv' *not found*." Jika terjadi kesalahan parsing karena format atau garis besar file yang salah, akan muncul pesan "Kesalahan: 'Tidak dapat memparse file CSV. Periksa format dan garis besar file.'" Selain kesalahan ini, program akan menemukan kesalahan dan mencetaknya. Metode ini membuat program lebih aman dan menginformasikan tentang kemungkinan kegagalan pembacaan file data.

3.6.2 *Data Exploration*

Lihat kumpulan data untuk memahami strukturnya, menemukan nilai yang hilang, menemukan tipe data untuk setiap kolom, dan menemukan distribusi fitur utama. Periksa data untuk anomali atau ketidakkonsistenan. Tentukan variabel prediksi Anda, seperti probabilitas keberhasilan implan atau kegagalan, dan metrik terkait.

Penalaran: Memeriksa jenis data, nilai yang hilang, distribusi, dan bagaimana fitur dan variabel target berhubungan satu sama lain.

Program yang dibuat untuk skrip eksplorasi data awal (*Exploratory Data Analysis/EDA*), yang menggunakan pustaka *pandas, matplotlib.pyplot*, dan *seaborn* untuk menganalisis dataset yang berkaitan dengan kelayakan implan gigi. Tujuan utamanya adalah mendapatkan pemahaman tentang struktur data, menangani data yang hilang, mengevaluasi distribusi fitur, dan mengeksplorasi bagaimana fitur-fitur berhubungan dengan target Layak Implan. Ini adalah penjelasan terstruktur yang menjelaskan maksud tiap bagian:

1. Menampilkan Tipe Data Tiap Kolom

   Print (df.dtypes) supaya menampilkan jenis data dari setiap kolom dalam *DataFrame*, yang dapat berupa *integer, float*, atau objek.

   Sehingga daftar kolom dan tipe data, seperti: Usia int64, Jumlah Gigi Hilang int64, Status Merokok *object* dan seterusnya.

2. Mengecek Nilai yang Hilang

   print(df.isnull().sum()) supaya menghitung nilai kosong (*NaN*) untuk setiap kolom. Jumlah nilai kosong per kolom sehingga Usia 0, Jumlah Gigi Hilang 3, Status Merokok 0 dan seterusnya.

3. Analisis Distribusi Fitur Numerik

   numerical_features = ['Usia', 'Jumlah Gigi Hilang'] supaya melihat histogram dengan kurva *density* KDE untuk masing-masing fitur numerik dan *plot box* untuk melihat *outlier*. Hasil *histogram* akan menunjukkan bentuk distribusi usia dan jumlah gigi yang hilang (apakah normal, miring, dll.), dan plot kotak akan menunjukkan median, rentang data, dan jika ada *outlier*.

4. Analisis Fitur Kategorikal

   for feature in categorical_features:

   print(df[feature].value_counts())

   Supaya menampilkan daftar nilai khusus untuk setiap kategori dan frekuensi mereka dengan hasil misal dateset status merokok yang Tidak Merokok 180, Perokok ringan 80, Perokok berat 37 agar ini meningkatkan pemahaman distribusi kelas untuk setiap atribut kategori.

5. Distribusi Target Variable (Layak Implan)

   sns.countplot (x='Layak Implan', data=df) supaya menampilkan grafik batang jumlah data berdasarkan kategori target, seperti Layak dan Tidak Layak dengan menghasilkan visualisasi seimbang atau tidaknya kelas target. Contoh: Layak: 200, Tidak Layak: 97.

6. Hubungan Fitur Numerik dengan Target

   sns.boxplot (x='Layak Implan', *y=feature, data=df*) supaya membandingkan distribusi fitur numerik berdasarkan kategori Layak Implan, termasuk usia dan jumlah gigi yang hilang apakah pasien yang layak cenderung memiliki usia/gigi hilang yang lebih tinggi/rendah atau mempermudah melihat tren atau perbedaan.

7. Hubungan Fitur Kategorikal dengan Target

   sns.countplot (*x=feature*, hue='Layak Implan', data=df) supaya dengan menggunakan pewarnaan untuk masing-masing kelas target, bandingkan frekuensi tiap kategori fitur dengan target Layak Implan dengan menunjukkan distribusi Layak vs Tidak Layak dalam setiap kategori. Contoh: pada Status Merokok, mungkin "Perokok berat" lebih banyak yang tidak layak.

   Untuk memahami struktur dataset implan gigi, program ini melakukan eksplorasi data statistik dan visual. Pertama, program melihat tipe data dan menemukan nilai yang hilang. Kemudian, menggunakan *box plot* dan *histogram* untuk menunjukkan distribusi data numerik, seperti usia dan jumlah gigi yang hilang. Selanjutnya, program melihat nilai khusus untuk

fitur kategorikal, seperti jenis kelamin, status merokok, dan kondisi kesehatan, dan menganalisis bagaimana fitur tersebut berhubungan dengan variabel target Layak Implan. Hasilnya memberikan gambaran mendalam yang membantu pemahaman awal data dan menjadi dasar untuk pemodelan atau analisis lebih lanjut.

3.6.3 *Data Cleaning*

Pembersihan dan Praproses Data Penalaran: Menstandardisasi nilai, mengonversi tipe data jika perlu, menangani *outlier* dalam kategori "Usia" dan "Jumlah Gigi Hilang", dan kemudian mengodekan fitur kategoris.

Tujuan membuat nilai kategori konsisten (misalnya, membuat nilai "pria" berbeda dengan "pria"), mengonversi data numerik ke tipe yang tepat, menangani *outlier* dengan *winsorization*, mengubah fitur kategori ke bentuk numerik (menggunakan *encoding* satu-*hot*), dan menghasilkan dataset terakhir yang siap untuk pelatihan model. Langkah-langkahnya yaitu:

1. Standarisasi Nilai Kategorikal

   Supaya Nilai-nilai yang tidak konsisten, seperti "pria" versus "perokok ringan" atau "pria" versus "perokok ringan", dianggap seragam dan bersih. Setiap kolom kategori memiliki nilai standar. Ini membantu menghindari masalah saat pelatihan atau analisis model.

2. Konversi Kolom Numerik

   df[col] = pd.to_numeric(df[col], errors='coerce') supaya Pastikan kolom Jumlah Gigi Hilang dan Usia bertipe numerik. Karakter non-numerik, seperti "35 tahun", akan diubah menjadi NaN dengan hasil

kolom numerik yang bersih dari karakter aneh, dan dapat dihitung secara matematis.

3. Penanganan *Outlier* dengan *Winsorization*

   df[col] = winsorize(df[col], limits=[0.05, 0.05]) supaya Untuk membuat hasil distribusi data numerik lebih stabil dan tidak terpengaruh oleh nilai ekstrem saat pelatihan model, *outlier* (nilai ekstrem) dihapus dengan *winsorization*, yang berarti memotong 5% dari nilai tertinggi dan terendah, lalu menggantinya dengan nilai ambang.

4. *Encoding* Fitur Kategorikal (*One-Hot Encoding*)

   df_encoded = pd.get_dummies(df, drop_first=True) supaya mengubah kolom kategorikal menjadi bentuk biner 0 dan 1, yang diperlukan untuk memproses data oleh algoritma pembelajaran mesin. *Drop_first=True* memastikan bahwa *dummy variable trap*, juga dikenal sebagai redundansi data, tidak akan menghasilkan dataset baru *df_encoded. Dataset* ini hanya memiliki nilai numerik dan dapat digunakan untuk jenis *modeling* seperti r*egresi*, pohon keputusan, SVM, dll.

5. Tampilkan Data Akhir

   display(df_encoded.head()) supaya menampilkan dataset hasil encoding dengan lima baris pertama, bersama dengan hasil dataset numerik, siap digunakan untuk pelatihan model ML. Contoh hasil (teori):

   Contoh hasil (teori):

Usia	Jumlah Gigi Hilang	Jenis Kelamin_Wanita	Status Merokok_Perokok berat	...	Layak Implan_Ya
0.42	0.23	1	0	...	1

3.6.4 *Peature Engineering*

Rekayasa Fitur untuk Peningkatan Performa Model, menganalisis korelasi antar fitur, membuat fitur polinomial, menskalakan fitur numerik, dan membuat kerangka data akhir. Tujuan program *Python* untuk melakukan pra-pemrosesan data sebelum digunakan dalam model pembelajaran mesin, terutama untuk kasus klasifikasi seperti memprediksi kelayakan implan gigi. Semua langkah dijelaskan di sini:

1. *Correlation Analysis*

   corr_matrix = df_encoded.corr()

   highly_correlated = corr_matrix[np.abs(corr_matrix) > 0.8]

   Agar program menghitung matriks korelasi untuk setiap fitur dalam *df_encoded. highly_correlated* memilih fitur dengan korelasi tinggi yang lebih besar dari 0.8.Tujuannya adalah untuk menemukan multikolinearitas, yaitu karakteristik yang sangat mirip satu sama lain dan dapat dihapus untuk menghindari *overfitting*.

2. *Polynomial Features*

   poly = PolynomialFeatures(degree=2, include_bias=False)

   numerical_cols = ['Usia', 'Jumlah Gigi Hilang']

   poly_features = poly.fit_transform(df_encoded[numerical_cols])Feature Scaling

   Langkah membuat fitur baru dari kombinasi fitur numerik dengan menggunakan fitur polinomial. Misalnya, jika kita menggabungkan dua

fitur: Usia, Jumlah Gigi Hilang, Usia^2, Usia * Jumlah Gigi Hilang, Jumlah Gigi Hilang^2**.**

3. *Feature Scaling*

   scaler = StandardScaler()

   scaled_features = scaler.fit_transform(poly_df)

   Supaya standarisasi karakteristik hasil polinomial (*mean = 0, std = 1)*. Ini sangat penting untuk algoritma pembelajaran mesin yang sensitif terhadap skala, seperti SVM, KNN, dan regresi logistik.

4. *Final DataFrame Construction*

   df_final = pd.concat([df_encoded.drop(columns=numerical_cols), scaled_df, df_encoded['Layak Implan_Ya']], *axis=1)* langkah untuk menggabungkan suatu fitur kategorikal (hasil *encoding one-hot* di *df_encoded*, kecuali usia dan jumlah gigi yang hilang) *Fitur polinomial* yang sudah diskalakan Kolom target: Layak Implan_Ya *df_final* menjadi dataset akhir yang siap digunakan untuk pelatihan model.

3.6.5 *Data Splitting*

Program membagi data menjadi tiga bagian utama (data *splitting*). Set pelatihan, set validasi, dan set tes adalah langkah penting dalam proses pelatihan dan evaluasi model pembelajaran mesin, langkah-langkah pemisahan data untuk machine learning:

1. *Impor Library*

   Untuk membagi dataset, *import* fungsi *train_test_split* dari *sklearn.model_selection*.

2. Pisahkan Fitur dan Target (Label)

   Variable *independen* (fitur) adalah X, yang diambil dari *df_final* dengan menghapus kolom 'Layak Implan_Ya', dan variabel dependen (target/label) adalah y.

3. Bagi Data Menjadi *Training* dan *Temporary Set (Validation + Testing)*

   Dengan menggunakan *train_test_split*, data dibagi menjadi dua bagian. Data pelatihan (*X_train, y_train*) mengandung 80 persen, dan 20 persen sisanya disimpan dalam set sementara (*X_temp, y_temp*), yang akan dibagi lagi untuk validasi dan pengujian.

4. Bagi *Temporary Set* Menjadi Validasi dan *Testing Set*

   Set sementara dibagi 50:50 menjadi 10 persen data validasi *(X_val, y_val*) dan 10 persen data pengujian *(X_test, y_test)*.

5. Cetak Bentuk *(Shape)* dari Masing-Masing Set

   Dicetak ukuran masing-masing setnya:

   a. *X_train dan y_train*: (237, 328) dan (237, 2)

   b. *X_val dan y_val*: (30, 328) dan (30, 2)

   c. *X_test dan y_test*: (30, 328) dan (30, 2)

   Langkah Perhitungan Pembagian Dataset:

   1. Dataset Awal

      Anda dapat menghitung jumlah total data dari hasil pembagian sebagai berikut:

      umlah total baris (data observasi):

      237 (train)+30 (val)+30 (test)=297 data

Jumlah fitur (kolom) untuk X: 328 fitur

Target y memiliki 2 kolom, ini menunjukkan bahwa target telah **one-hot encoded** menjadi dua kelas.

2. Pembagian Data

Program membagi data menjadi 3 bagian:

a. *Training Set* (80%)

Digunakan untuk melatih model, ukurannya:

$80\% \times 297 = 237.6 \Rightarrow 237$ data

b. *Temporary Set* (20%)

Sisanya (20%) disimpan sementara sebelum dibagi lagi untuk validasi dan pengujian, ukuran temporary set: $297 - 237 = 60$ data

c. *Validation dan Test Set* (masing-masing 10%)

Temporary set dibagi **50:50**:

Validation: 30 data (50% dari 60)

Test: 30 data (50% dari 60)

Hasil Bentuk Dataset (*Shapes*) adalah:

Dataset	**Jumlah Data (Baris)**	**Jumlah Fitur**	**Bentuk (Shape)**
X_train	237	328	(237, 328)
y_train	237	2 (one-hot)	(237, 2)
X_val	30	328	(30, 328)
y_val	30	2	(30, 2)

X_test	30	328	(30, 328)
y_test	30	2	(30, 2)

Kesimpulan Perhitungan :

a. Jumlah total data = 297 baris.
b. *Training* = 80% (237 data)
c. *Validation* = 10% (30 data)
d. *Testing* = 10% (30 data
e. Pembagian dilakukan dengan random_state=42 agar hasilnya konsisten saat diulang.

3.6.6 *Model Training*

Langkah-langkah Evaluasi Model *Machine Learning* dengan Berbagai Algoritma:

1. *Impor Library* yang Dibutuhkan

   Logistic Regression, SVC, Random Forest Classifier, dan Gradient Boosting Classifier diimpor dari *Sklearn.*

   Mengimpor skor akurasi, ketepatan, *recall, skor f1,* dan *skor roc_auc* untuk evaluasi.

2. Inisialisasi Model

   Membuat kamus model dengan empat model klasifikasi: *Logistic Regression* (dengan max_iter=1000), SVM (dengan probability=True untuk menghitung AUC), dan *Random Forest Gradient Boosting*.

3. Pelatihan dan Evaluasi Model

   Setiap model dilatih dengan perulangan for: Pelatihan: Model dilatih pada *X_train dan y_train.iloc*[:, 0], yang merupakan kolom pertama target.

   Prediksi: Model memproyeksikan data validasi X_val. Probabilitas: probabilitas prediksi positif ([:, 1]) untuk perhitungan AUC.

4. Perhitungan Metrik Evaluasi

   Metrik untuk data validasi dihitung untuk setiap model:

   Accuracy menunjukkan seberapa sering prediksi benar. *Precision* menunjukkan seberapa tepat dari semua prediksi yang positif. *Recall* menunjukkan seberapa berhasil model ditemukan dari semua data yang positif.

   F1-nilai menunjukkan keharmonisan rata-rata antara ketepatan dan *recall*.

5. Penanganan Error *(Exception Handling*)

   Saat melatih model, kesalahan akan ditangani dan disimpan dalam hasil sebagai "*error*".

6. Menampilkan Hasil Evaluasi

   Metrik evaluasi setiap model dicetak ke layer:

Gambar 3. 2 Hasil Evaluasi Model

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC**
Logistic Regression	0.60	0.4286	0.2727	0.3333	0.4211
SVM	0.5333	0.3846	0.4545	0.4167	0.4976

Random Forest	0.5667	0.3333	0.1818	0.2353	0.4425
Gradient Boosting	0.4667	0.2727	0.2727	0.2727	0.3876

Kesimpulan dari tabel berikut adalah Pada data validasi, SVM memiliki nilai *F1-score* dan recall tertinggi. *Logistic Regression* memiliki ketepatan tertinggi sebesar 60%. Semua model memiliki nilai AUC yang rendah, yang menunjukkan bahwa diskriminasi antar kelas masih lemah. Saya bisa bantu lanjutkan jika Anda ingin menyempurnakan model dengan cara-cara seperti tuning, pemilihan fitur, atau balancing data.

Langkah Perhitungan Metrik Evaluasi:

Dari data aktual dan hasil prediksi model dibandingkan dengan data validasi (X_val), yang menghasilkan *confusion matrix* berikut:

	Prediksi Positif (1)	**Prediksi Negatif (0)**
Aktual Positif (1)	TP (*True Positive*)	FN (*False Negative*)
Aktual Negatif (0)	FP (*False Positive*)	TN (*True Negative*)

Berikut adalah perhitungan yang dilakukan dari matrix ini:

3. *Accuracy* (Akurasi)

   Akurasi menunjukkan seberapa besar proporsi prediksi yang benar terhadap semua data.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

4. *Precision* (Presisi)

   Seberapa akurat prediksi positif dinilai oleh presisinya:

$$\text{Precision} = \frac{TP}{TP + FP}$$

5. *Recall (Sensitivity / True Positive Rate*)

   Seberapa baik model mendeteksi kasus positif ditunjukkan oleh recall:

$$\text{Recall} = \frac{TP}{TP + FN}$$

6. *F1 Score*

   Saat kita ingin keseimbangan antara *precision* dan *recall*, *skor F1* adalah rata-rata harmonik.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

7. *ROC AUC Score*

   Kemampuan model untuk membedakan kelas positif dan negatif ditunjukkan oleh AUC (*Area Under the Curve*) dari ROC (*Receiver Operating Characteristic*).

   Dengan menggunakan integral dari kurva TPR versus FPR, nilai probabilitas *(model.predict_proba(X_val)*[:, 1]) dan label aktual (y_val) dihitung secara internal:

   a. **TPR (*True Positive Rate*)** = *Recall*

   b. **FPR *(False Positive Rate*)** = FP / (FP + TN)

   AUC bernilai:

   a. **0.5** jika model acak (tidak bisa membedakan)

b. **1.0** jika model sempurna

Jika : TP = 10, TN 12, FP = 4, FN = 6.

Maka:

a. *Accuracy* = (10 + 12) / (10 + 12 + 4 + 6) = 22 / 32 = 0.6875
b. *Precision* = 10 / (10 + 4) = 10 / 14 = 0.714
c. *Recall* = 10 / (10 + 6) = 10 / 16 = 0.625
d. *F1 Score* = 2 * (0.714 * 0.625) / (0.714 + 0.625) ≈ 0.666
e. AUC dihitung dari probabilitas skor prediksi terhadap nilai aktual.

3.6.7 *Model Aptimization*

Optimalkan hiperparameter model Regresi Logistik, yang memiliki kinerja terbaik yang diidentifikasi pada langkah sebelumnya, dengan menggunakan GridSearchCV.

Program *Python* di atas melakukan pemodelan klasifikasi dan optimasi hyperparameter dengan menggunakan *GridSearchCV* dan *Logistic Regression* dari *library scikit-learn*. Ini adalah penjelasan langkah-langkahnya:

1. *Import Library*

   Mengimpor: *GridSearchCV* untuk menemukan kombinasi parameter yang optimal. *Logistic Regression* sebagai model untuk kategorisasi metrik evaluasi yang diakses melalui *sklearn.metrics*.

2. Definisikan *Grid Parameter*

   param_grid = {

   'C': [0.001, 0.01, 0.1, 1, 10, 100],

'penalty': ['l1', 'l2'],

'solver': ['liblinear', 'saga'] # solver yang kompatibel dengan L1

}

Penjelasannya *Grid* ini terdiri dari kombinasi dari: C: parameter regularisasi (regulasi yang lebih kecil menunjukkan regulasi yang lebih *kuat), penalty*: jenis *penalti* (l1 = *Lasso*, l2 = *Ridge*), *solver*: metode optimisasi yang sesuai dengan penalty.

3. Inisialisasi dan Jalankan *GridSearchCV*

   logreg = LogisticRegression(max_iter=10000)

   grid_search = GridSearchCV(estimator=logreg,

   param_grid=param_grid, scoring='roc_auc', cv=5)

   maksudnya dalam pencarian parameter terbaik berdasarkan nilai AUC (*Area Under Curve*) menggunakan cross-validation lima kali.

4. Latih Model dan Ambil Model Terbaik

   grid_search.fit(X_train, y_train.iloc[:, 0])

   *best_logreg = grid_search.best_estimator_*

   Setelah *training*, diambil model dengan kombinasi parameter terbaik:

   Best Hyperparameters: {'C': 10, 'penalty': 'l1', 'solver': 'liblinear'}

5. Evaluasi Model Terbaik pada Data Validasi

   Model yang paling efektif diuji pada X_val, dan hasil ramalan dievaluasi dengan menggunakan: *accuracy_score, precision_score, recall_score, f1_score, roc_auc_score.*

Dilanjutkan dengan langkah perhitungan metrik evaluasi, didalam prediksi menghasilkan *matrix confusion* berikut:

	Prediksi Positif (1)	**Prediksi Negatif (0)**
Aktual Positif (1)	TP = 9	FN = 24
Aktual Negatif (0)	FP = 12	TN = 15

Nilai-nilai metrik adalah:

a. ***Accuracy***: 0.6

$$= \frac{TP + TN}{TP + TN + FP + FN} = \frac{9 + 15}{9 + 15 + 12 + 24} = \frac{24}{60} = 0.6$$

b. ***Precision***: 0.4286

$$= \frac{TP}{TP + FP} = \frac{9}{9 + 12} = \frac{9}{21} \approx 0.4286$$

c. ***Recall****:* 0.2727

$$= \frac{TP}{TP + FN} = \frac{9}{9 + 24} = \frac{9}{33} \approx 0.2727$$

d. ***F1-score***: 0.3333

$$= 2 \times \frac{0.4286 \times 0.2727}{0.4286 + 0.2727} \approx 0.3333$$

e. ***AUC***: 0.3971

Dihitung dengan menggunakan kurva ROC untuk membandingkan probabilitas prediksi positif dengan label aktual.

Kesimpulan dari penjelasan model terbaik menggunakan C = 10, *penalty = l1, dan solver = liblinear*. Namun, kinerjanya kurang baik karena *recall* dan AUC masih rendah. Perlu dipelajari lebih lanjut, seperti teknik

fitur, perimbangan data (jika kelas tidak seimbang), dan model lain (misalnya, berbasis pohon).

3.6.7 Model Evaluation

Dengan menggunakan program ini, kinerja model klasifikasi (dalam hal ini, *best_logreg,* model regresi logistik) dibandingkan dengan set data uji. Menghitung berbagai metrik performa, seperti: *Accuracy* (Akurasi), *Precision* (Presisi), *Recall* (Sensitivitas), *F1 Score*, *AUC (Area Under Curve*), langkah-langkahnya sebagai berikut:

1. *Import Library*:

   from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

   Menjelaskan menggunakan sklearn.metrics untuk mengimpor fungsi evaluasi.

2. Prediksi pada Data Uji:

   y_pred_test = best_logreg.predict(X_test)

   y_prob_test = best_logreg.predict_proba(X_test)[:, 1]

   Menjelaskan prediksi_proba memberikan probabilitas prediksi, terutama kolom [1] untuk kelas positif (1). Prediksi memberikan label hasil prediksi.

3. Hitung Metrik Evaluasi:

   accuracy_test = accuracy_score(y_test.iloc[:, 0], y_pred_test)

   precision_test = precision_score(y_test.iloc[:, 0], y_pred_test)

   recall_test = recall_score(y_test.iloc[:, 0], y_pred_test)

   f1_test = f1_score(y_test.iloc[:, 0], y_pred_test)

auc_test = roc_auc_score(y_test.iloc[:, 0], y_prob_test)

Menjelaskan setiap metrik dibandingkan dengan label sebenarnya (y_test) dan prediksi model (*y_pred_test atau y_prob_test).*

Dialanjutkan dengan Penjelasan & Langkah Perhitungan Metrik:

Jika jumlah data uji = 30, maka metrik dihitung dari confusion matrix berikut (asumsi):

	Pred: Positif	Pred: Negatif
Actual: Positif	TP = 6	FN = 4
Actual: Negatif	FP = 9	TN = 11

Maka:

Accuracy = (TP + TN) / total

= (6 + 11) / 30 = 17 / 30 = **0.5667**

Precision = TP / (TP + FP)

= 6 / (6 + 9) = 6 / 15 = **0.4**

Recall = TP / (TP + FN)

= 6 / (6 + 4) = 6 / 10 = **0.6**

F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

= 2 * (0.4 * 0.6) / (0.4 + 0.6) = 0.48

AUC = Mengukur kemampuan model membedakan kelas dengan **nilai probabilitas prediksi**. Nilai AUC 0.705 berarti model bisa membedakan kelas positif dan negatif dengan performa **di atas acak**.

Kesimpulannya tingkat akurasi dan F1 yang rendah menunjukkan bahwa model tidak cukup baik untuk memprediksi secara seimbang semua kelas.

Precision yang rendah dapat menunjukkan banyak positif palsu.

Dengan AUC 0,705, model masih memiliki kemampuan klasifikasi yang layak, meskipun tidak ideal.

Mungkin ada *overfitting* atau *underfitting* model jika hasil ini berbeda dari data pelatihan dan validasi.

3.7 Lokasi Penelitian

Penelitina dilakukan di Klinik Ellisa *Dental care* yang berlokasi di Kota Batam, di Ruko SP Plaza 3 Blok Mawar No. 30 Batu Aji. Penelitian dilakukan dari maret 2025 sampai dengan Juli 2025 yang dilakukan selama satu semestre.

Gambar 3.2 peta lokasi klinik

3.8 Jadwal Penelitian

Jadwan penelitian dimulai dari dan sampai selesai penelitian ini dibuat:

Tabel 3. 3 jadwal penelitian

No	Kegiatan	Waktu Penelitian																		
		Maret 2025				April 2025				Mei 2025				Juni 2025				Juli 2025		
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Menentukan Objek dan Judul Penelitian																				
2	BAB 1 sampai BAB 3																				
3	BAB 4 sampai BAB 5																				
4	Jurnal Penelitian																				