BAB 3
METODE PENELITIAN

Dalam metode penelitian ini, digunakan algoritma Decision Tree C4.5 adalah sebuah pendekatan yang populer dalam pemodelan prediktif dan pembelajaran mesin. Algoritma ini digunakan untuk menghasilkan pohon keputusan yang mewakili aturan keputusan hierarkis berdasarkan fitur-fitur dari data *input*.

3.1 BAHAN DAN ALAT PENELITIAN

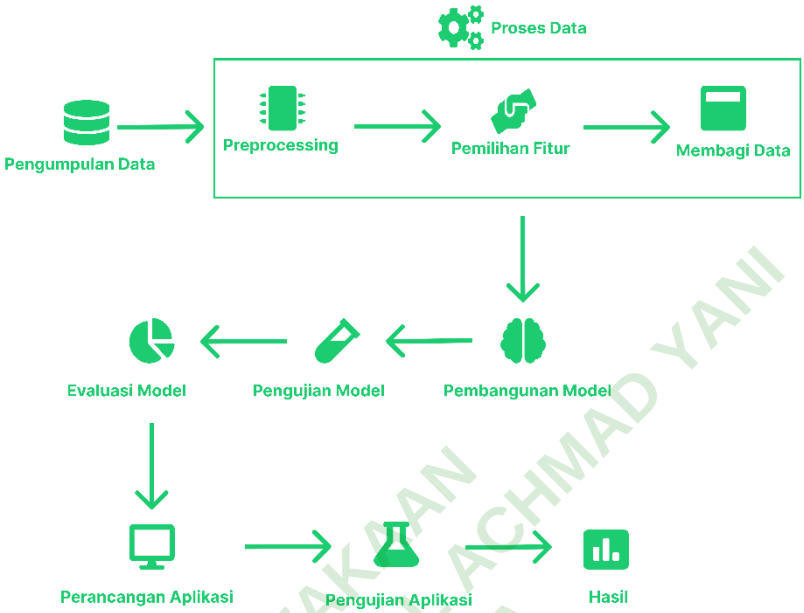
Bahan yang digunakan dalam penelitian ini berasal dari *dataset* yang diperoleh dari sumber data publik. Sumber data yang berhasil dikumpulkan adalah melalui situs web Kaggle (www.kaggle.com) dengan judul *"Early Stage Diabetes Risk Prediction"*. *Dataset* ini disediakan oleh rumah sakit di Sylhet, Bangladesh, dan telah disetujui untuk penggunaan penelitian. Terdapat total 520 catatan data dalam *dataset* ini, dengan 17 atribut yang beragam. Dari jumlah tersebut, sebanyak 320 data menunjukkan hasil positif terdiagnosis diabetes, sedangkan 200 data menunjukkan hasil negatif terdiagnosis diabetes.

Alat yang digunakan dalam penelitian ini adalah komputer dengan spesifikasi yang memadai untuk menjalankan sistem operasi, perangkat lunak pengembangan, dan memiliki koneksi internet. Sistem operasi dan program-program aplikasi yang digunakan dalam pengembangan aplikasi ini adalah:

1. Sistem Operasi: Ubuntu 22.04 atau Windows 10.
2. Teks editor: Visual Studio Code
3. Bahasa pemrograman: Python dengan *framework* Flask.
4. Environment: Anaconda.

3.2 JALAN PENELITIAN

Pada tahapan ini penelitian akan dilakukan secara berurutan. Berikut ini merupakan alur penelitian aplikasi deteksi risiko diabetes berbasis web menggunakan metode Decision Tree C4.5, ditunjukan pada Gambar 3.1.



Gambar 3.1 Alur penelitian

3.2.1 Pengumpulan Data

Langkah pertama adalah mengumpulkan *dataset* yang sesuai dengan tujuan penelitian, yakni *dataset* yang berhubungan dengan informasi medis atau kesehatan yang mencakup variabel-variabel yang relevan untuk deteksi risiko diabetes. *Dataset* yang digunakan berasal dari Kaggle dan diambil pada tahun 2020. *Dataset* ini terdiri dari 520 data yang dikategorikan ke dalam kelas hasil positif atau negatif. Atribut-atribut yang digunakan dalam dataset ini termasuk:

Tabel 3.1 Atribut yang digunakan dalam *dataset*

No	Nama Atribut	Keterangan
1	Usia *(Age)*	16-90 Tahun
2	Jenis Kelamin *(Gender)*	*Male/Female*
3	Sering Buang Air Kecil *(Polyuria)*	*Yes/No*
4	Sering Haus *(Polydipsia)*	*Yes/No*

5	Cepat Kehilangan berat badan *(sudden weight loss)*	*Yes/No*
6	Kelelahan *(weakness)*	*Yes/No*
7	Banyak Makan *(Polyphagia)*	*Yes/No*
8	Infeksi Jamur *(Genital trush)*	*Yes/No*
9	Penghilatan Kabur *(visual blurring)*	*Yes/No*
10	Gatal-Gatal *(Itching)*	*Yes/No*
11	Mudah marah *(Irritability)*	*Yes/No*
12	Lama Penyembuhan Luka *(delayed healing)*	*Yes/No*
13	Kelumpuhan sebagian *(Partial Paresis)*	*Yes/No*
14	Kekakuan Otot *(muscle stiffness)*	*Yes/No*
15	Kerontokan Rambut *(Alopecia)*	*Yes/No*
16	Kegemukan *(Obesity)*	*Yes/No*
17	*Class*	Positif/Negatif

3.2.2 Proses Data

Proses data adalah rangkaian langkah-langkah yang dilakukan untuk mempersiapkan data mentah sehingga dapat digunakan dalam analisis atau pemodelan. Langkah-langkah ini meliputi *preprocessing* data, pemilihan fitur, dan pembagian data.

a. Preprocessing Data

Preprocessing data adalah langkah awal dalam proses data yang bertujuan untuk membersihkan dan mempersiapkan data mentah sebelum dianalisis lebih lanjut. Berikut adalah Langkah-langkah melakukan *preprocessing* data meliputi memeriksa nilai yang hilang, memeriksa tipe data, mengkodekkan kolom objek menjadi numerik, dan menampilkan informasi data setelah *preprocessing*.

1. Memeriksa nilai yang hilang

Langkah pertama dalam *preprocessing* adalah memeriksa apakah ada nilai yang hilang *(missing values)* dalam *dataset*. Nilai yang hilang dapat mempengaruhi analisis dan hasil model, sehingga perlu diidentifikasi terlebih dahulu. Berikut kode untuk memeriksa nilai yang hilang.

```
# Memeriksa nilai yang hilang
df.isna().sum()
```

Kode tersebut digunakan untuk mencari nilai yang hilang dan menjumlahkan nilai yang hilang pada setiap kolom. *Output* dari kode memeriksa nilai yang hilang dapat dilihat pada Gambar 3.2.

```
Out[18]: Age                 0
         Gender              0
         Polyuria            0
         Polydipsia          0
         sudden weight loss  0
         weakness            0
         Polyphagia          0
         Genital thrush      0
         visual blurring     0
         Itching             0
         Irritability        0
         delayed healing     0
         partial paresis     0
         muscle stiffness    0
         Alopecia            0
         Obesity             0
         class               0
         dtype: int64
```


Gambar 3.2 *Output* kode memeriksa nilai yang hilang

Hasil menunjukan jumlah nilai yang hilang di setiap kolom. Pada hasil tersebut tidak terdapat nilai yang hilang.

2. Memeriksa tipe data

Langkah berikutnya adalah memeriksa tipe data dari setiap kolom untuk memastikan bahwa tipe data sesuai dengan yang diharapkan (misalnya, angka, teks, dll.). Tipe data yang tepat penting untuk langkah-langkah *preprocessing* berikutnya dan untuk model machine learning. Berikut kode untuk memeriksa tipe data pada setiap kolom.

```
# memeriksa tipe data
df.info()
```

Output dari kode memeriksa tipe data dapat dilihat pada Gambar 3.3.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 520 entries, 0 to 519
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Age                 520 non-null    int64
 1   Gender              520 non-null    object
 2   Polyuria            520 non-null    object
 3   Polydipsia          520 non-null    object
 4   sudden weight loss  520 non-null    object
 5   weakness            520 non-null    object
 6   Polyphagia          520 non-null    object
 7   Genital thrush      520 non-null    object
 8   visual blurring     520 non-null    object
 9   Itching             520 non-null    object
 10  Irritability        520 non-null    object
 11  delayed healing     520 non-null    object
 12  partial paresis     520 non-null    object
 13  muscle stiffness    520 non-null    object
 14  Alopecia            520 non-null    object
 15  Obesity             520 non-null    object
 16  class               520 non-null    object
dtypes: int64(1), object(16)
memory usage: 69.2+ KB
```

Gambar 3.3 *Output* kode memeriksa tipe data

Output menunjukkan tipe data dari setiap kolom. Kolom *'Age'* memiliki tipe data int64 (angka bulat), sementara kolom lainnya bertipe *object* (teks).

3. Mengkodekan kolom objek menjadi numerik

Kolom yang berisi data kategori perlu diubah menjadi data numerik agar dapat digunakan dalam model *machine learning*. Pemecahan masalah ini dapat dilakukan dengan menggunakan LabelEncoder untuk mengubah data kategori menjadi angka. Berikut kode untuk mengubah data kategorikal menjadi numerik.

```
# Mengkodekan kolom objek menjadi numerik
label_encoder = LabelEncoder()
for column in df.columns:
    df[column] = label_encoder.fit_transform(df[column])
```

Output dari kode mengkodekan kolom objek menjadi numerik dapat dilihat pada Gambar 3.4.

Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Alopecia	Obesity	class
16	1	0	1	0	1	0	0	0	1	0	1	0	1	1	1	1
34	1	0	0	0	1	0	0	1	0	0	0	1	0	1	0	1
17	1	1	0	0	1	1	0	0	1	0	1	0	1	1	0	1
21	1	0	0	1	1	1	1	0	1	0	1	0	0	0	0	1
36	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
15	0	1	1	1	0	1	0	0	1	0	1	1	0	0	0	1
24	0	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1

Gambar 3.4 *Output* kode mengkodekan kolom objek menjadi numerik

4. Menampilkan informasi data setelah *preprocessing*

Setelah melakukan *preprocessing*, langkah berikutnya adalah menampilkan informasi dari *dataset* untuk memastikan bahwa semua tahapan telah dilakukan dengan benar dan dataset siap digunakan untuk analisis atau pemodelan. Berikut kode untuk menampilkan informasi data setelah *preprocessing*.

```
# Tampilan informasi tentang data setelah preprocessing
print("Informasi Data setelah Preprocessing:")
print(df.info())
```

Output setelah *preprocessing* menunjukkan bahwa tidak ada nilai yang hilang setelah *preprocessing*, dan tipe data sudah sesuai. Pada *output* tersebut juga menunjukkan kolom yang sebelumnya bertipe kategorikal telah dikodekan menjadi numerik. *Output* dari kode menampilkan informasi sata setelah *preprocessing* dapat dilihat pada Gambar 3.5.

```
Informasi Data setelah Preprocessing:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 520 entries, 0 to 519
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Age                 520 non-null    int64
 1   Gender              520 non-null    int32
 2   Polyuria            520 non-null    int32
 3   Polydipsia          520 non-null    int32
 4   sudden weight loss  520 non-null    int32
 5   weakness            520 non-null    int32
 6   Polyphagia          520 non-null    int32
 7   Genital thrush      520 non-null    int32
 8   visual blurring     520 non-null    int32
 9   Itching             520 non-null    int32
 10  Irritability        520 non-null    int32
 11  delayed healing     520 non-null    int32
 12  partial paresis     520 non-null    int32
 13  muscle stiffness    520 non-null    int32
 14  Alopecia            520 non-null    int32
 15  Obesity             520 non-null    int32
 16  class               520 non-null    int32
dtypes: int32(16), int64(1)
memory usage: 36.7 KB
None
```

Gambar 3.5 *Output Preprocessing*

b. Pemilihan Fitur

Pemilihan fitur adalah proses penting dalam analisis data dan pembuatan model yang melibatkan identifikasi fitur-fitur yang paling relevan untuk digunakan dalam memprediksi variabel target. Berikut ini adalah Langkah - langkah pemilihan fitur meliputi memisahkan kumpulan data, menghitung korelasi dengan variabel target, visualisasi matriks korelasi, dan visualisasi korelasi fitur dengan variabel target.

1. Memisahkan kumpulan data

Tahapan pertama dalam pemilihan fitur adalah memisahkan kumpulan data menjadi fitur-fitur (X) dan variabel target (y). Fitur-fitur adalah variabel yang digunakan sebagai *input* untuk model, sementara variabel target adalah variabel yang ingin diprediksi. Berikut kode untuk memisahkan Kumpulan data menjadi fitur dan variabel target.

```
# Memisahkan kumpulan data menjadi fitur dan variabel target
X = df.drop('class', axis=1)
y = df['class']
```

Kode diatas memisahkan *dataset* menjadi fitur dan variabel target. X berisi fitur-fitur, sedangkan y berisi variabel target. Kolom *'class'* dihapus dari X, karena merupakan variabel yang ingin diprediksi.

2. Menghitung korelasi dengan variabel target

Setelah memisahkan fitur dan variabel target, langkah selanjutnya adalah menghitung korelasi antara setiap fitur dalam X dengan variabel target y. Korelasi ini mengindikasikan seberapa erat hubungan antara fitur-fitur (X) dengan variabel target (y). Korelasi yang tinggi menunjukkan bahwa fitur tersebut memiliki pengaruh yang signifikan terhadap variabel target. Berikut kode untuk menghitung korelasi dengan variabel target.

```
# Korelasi antara setiap fitur dengan target
correlation_with_target = X.corrwith(y)
print(correlation_with_target)
```

Pada kode diatas, korelasi antara setiap fitur dengan variabel target dihitung menggunakan metode .corrwith(). Hasilnya adalah deretan korelasi antara setiap fitur dan variabel target. *Output* dari kode korelasi antara setiap fitur dengan target dapat dilihat pada Gambar 3.6.

```
Age                 0.106419
Gender             -0.449233
Polyuria            0.665922
Polydipsia          0.648734
sudden weight loss  0.436568
weakness            0.243275
Polyphagia          0.342504
Genital thrush      0.110288
visual blurring     0.251300
Itching            -0.013384
Irritability        0.299467
delayed healing     0.046980
partial paresis     0.432288
muscle stiffness    0.122474
Alopecia           -0.267512
Obesity             0.072173
dtype: float64
```

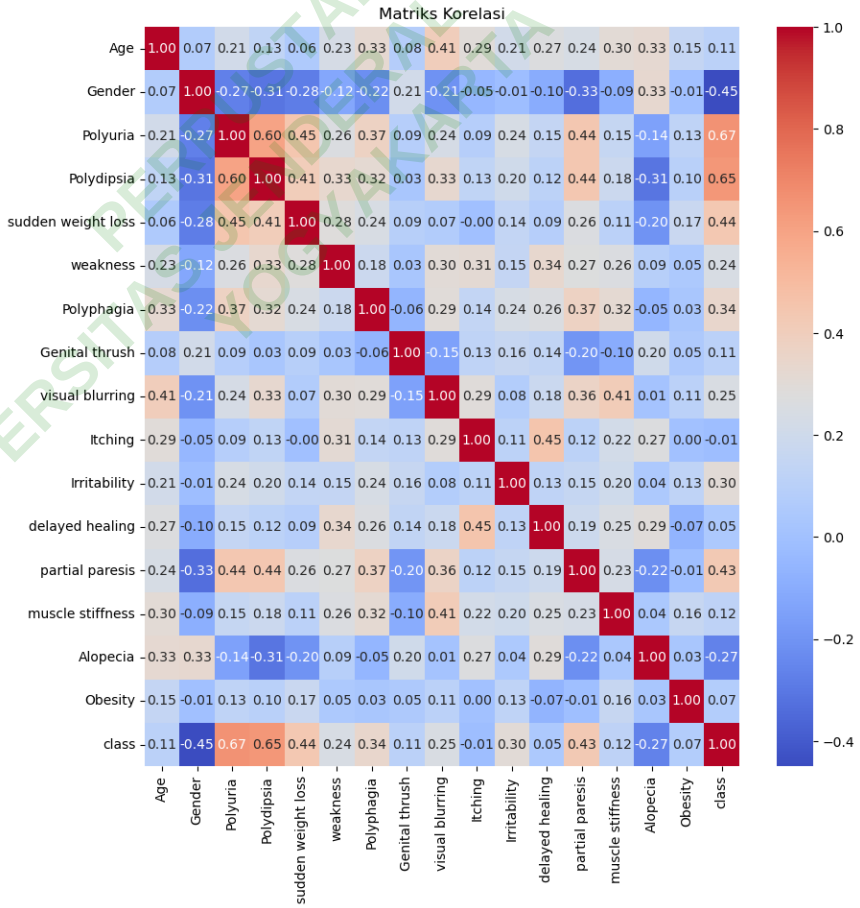
Gambar 3.6 *Output* dari korelasi fitur dengan target

3. Visualisasi matriks korelasi

Langkah ini melibatkan visualisasi matriks korelasi antara semua fitur dalam kumpulan data. *Heatmap* digunakan untuk memvisualisasikan korelasi antara setiap pasangan fitur. Anotasi dengan angka *float* dua digit membantu dalam memberikan nilai korelasi yang lebih jelas. Berikut kode untuk visualisasi matriks korelasi.

```
plt.figure(figsize=(10, 10))
corr = df.corr()
sns.heatmap(corr, annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Matriks Korelasi')
plt.show()
```

Heatmap digunakan untuk menampilkan korelasi antara setiap pasangan fitur. Anotasi dengan angka *float* dua digit (fmt='.2f') memberikan nilai korelasi yang lebih jelas. Berikut *output* dari kode visualisasi matriks korelasi dapat dilihat pada Gambar 3.7.



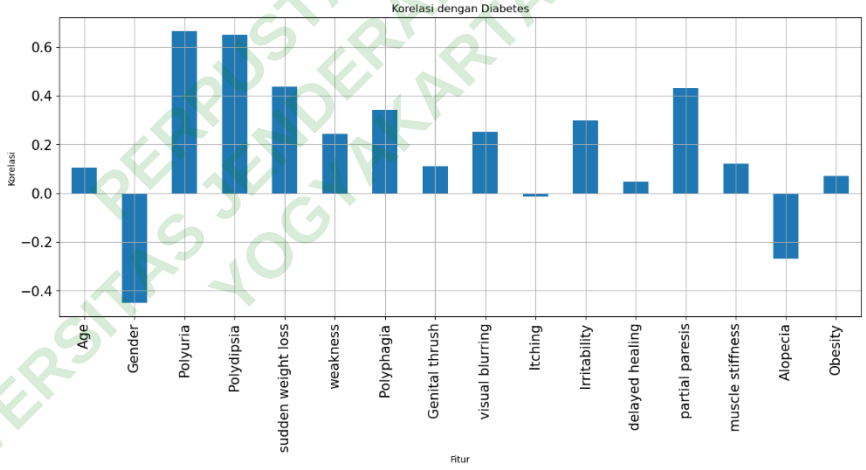
Gambar 3.7 *Heatmap* matriks korelasi

4. Visualisasi korelasi fitur dengan variabel target

Langkah terakhir adalah memvisualisasikan korelasi antara setiap fitur dan variabel target dalam bentuk diagram batang. Setiap batang dalam diagram tersebut mewakili korelasi antara fitur tertentu dengan variabel target. Visualisasi ini membantu dalam menyoroti fitur-fitur yang memiliki korelasi tinggi atau rendah dengan variabel target, yang dapat memengaruhi kinerja model. Berikut kode untuk memvisualisasikan korelasi fitur dengan variabel target.

```
# Korelasi Fitur
plt.figure(figsize=(16, 6))
correlation_with_target.plot.bar(title="Korelasi dengan
Diabetes", fontsize=15, rot=90, grid=True)
plt.xlabel("Fitur")
plt.ylabel("Korelasi")
plt.show()
```

Output dari kode korelasi fitur dengan variabel target dapat dilihat pada Gambar 3.8.



Gambar 3.8 Diagram korelasi fitur dengan variabel target

c. Pembagian Data

Pembagian data merupakan tahapan penting dalam proses pembangunan model. Tujuannya adalah untuk memisahkan data menjadi dua bagian yakni set pelatihan *(train set)* dan set pengujian *(test set)*. Set pelatihan digunakan untuk melatih model, sedangkan set pengujian digunakan untuk memvalidasi kinerja model. Berikut adalah penjelasan langkah-langkah pembagian data meliputi memisahkan data menjadi data latih dan data uji, memeriksa dimensi data latih dan

data uji, dan memeriksa distribusi variabel target dalam pembagian data latih dan uji.

1. Memisahkan data menjadi data latih dan data uji

Pada tahapan ini, data dibagi menjadi dua bagian menggunakan fungsi train_test_split dari library scikit-learn. Set pelatihan biasanya memiliki proporsi yang lebih besar dari set pengujian, dan proporsi ini dapat disesuaikan melalui parameter test_size. Berikut kode untuk memisahkan data menjadi data latih dan data uji.

```
# Memisahkan data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, stratify=y, random_state=42)
```

Pada kode tersebut menggunakan fungsi train_test_split dari library scikit-learn untuk membagi *dataset* menjadi data latih dan data uji. Di sini, X adalah kumpulan fitur, dan y adalah variabel target. Parameter test_size=0.2 menentukan proporsi data uji, yang dalam kasus ini adalah 20% dari total data. Pada stratify=y memastikan bahwa pembagian data dilakukan dengan mempertahankan distribusi kelas yang sama seperti pada variabel target y. Sedangkan random_state=42 bertujuan untuk membuat proses pembagian data menjadi teratur dan dapat diprediksi, sehingga hasilnya dapat diulang dengan konsistensi yang tinggi.

2. Memeriksa dimensi data latih dan data uji

Setelah pembagian data dilakukan, penting untuk memeriksa dimensi dari setiap bagian. Dimensi ini mencakup jumlah baris (data) dan jumlah kolom (fitur). Berikut kode untuk memeriksa dimensi data latih dan data uji.

```
# Memeriksa dimensi data latih dan data uji
print('Dimensi Data Latih dan Data Uji:')
print("Data Latih - Fitur:", X_train.shape,
"Target:",y_train.shape)
print("Data Uji - Fitur:", X_test.shape, "Target:",
y_test.shape)
```

Dalam kode memeriksa dimensi data latih dan data uji. X_train.shape dan y_train.shape digunakan untuk menunjukkan jumlah baris (data) dan kolom (fitur dan target) dalam data latih. Sedangkan X_test.shape dan y_test.shape digunakan untuk menunjukkan jumlah baris (data) dan kolom (fitur dan target) dalam data uji.

Output dari kode memeriksa dimensi data latih dan data uji, dapat dilihat pada Gambar 3.9.

```
Dimensi Data Latih dan Data Uji:
Data Latih - Fitur: (416, 16) Target: (416,)
Data Uji - Fitur: (104, 16) Target: (104,)
```

Gambar 3.9 Output kode dimensi data latih dan uji

Dari *output* tersebut, Data latih memiliki 416 baris dan 16 fitur, sedangkan Data uji memiliki 104 baris dan 16 fitur. Variabel target memiliki 416 sampel dalam data *training* dan 104 sampel dalam data *testing*, mengikuti pembagian proporsi 80:20 yang telah ditetapkan.

3. Memeriksa distribusi variabel target dalam pembagian data latih dan uji

Pemeriksaan distribusi variabel target dalam setiap bagian data membantu memastikan bahwa pembagian data dilakukan secara merata, terutama untuk kasus klasifikasi. Distribusi yang tidak seimbang dapat memengaruhi kinerja model. Berikut kode untuk memeriksa distribusi variabel target dalam pembagian data latih dan data uji.

```
# Memeriksa distribusi variabel target dalam pemisahan data
latih dan uji
print('Distribusi variabel target dalam data latih:')
print(y_train.value_counts())

print('Distribusi variabel target dalam data uji:')
print(y_test.value_counts())
```

Tujuan dari kode tersebut adalah untuk memeriksa distribusi variabel target dalam data latih dan data uji setelah dilakukannya pembagian. Fungsi value_counts() digunakan untuk menghitung frekuensi munculnya setiap nilai dalam variabel target, sehingga dapat mengetahui seberapa seimbang distribusi kelasnya. Output dari kode memeriksa distribusi variabel target dalam pemisahan data latih dan uji, dapat dilihat pada Gambar 3.10.

```
Distribusi variabel target dalam data latih:
1    256
0    160
Name: class, dtype: int64
Distribusi variabel target dalam data uji:
1    64
0    40
Name: class, dtype: int64
```

Gambar 3.10 Gambar hasil distribusi variabel target

Dalam output tersebut, dalam data latih, terdapat 256 sampel dengan nilai variabel target 1 (positif) dan 160 sampel dengan nilai variabel target 0 (negatif). Sementara dalam data uji, terdapat 64 sampel dengan nilai variabel target 1 (positif) dan 40 sampel dengan nilai variabel target 0 (negatif). Distribusi kelas dalam kedua *dataset* terlihat cukup seimbang, yang merupakan hal yang baik karena dapat mengurangi risiko bias pada model.

3.2.3 Pembangunan Model

Pembangunan model adalah langkah penting dalam proses *machine learning*, di mana pengimplementasian algoritma untuk membuat prediksi berdasarkan data yang tersedia. Pada tahap ini, digunakan algoritma Decision Tree C4.5 untuk membangun model prediksi risiko diabetes. Langkah pertama adalah membuat model Decision Tree menggunakan algoritma C4.5. Berikut kode untuk membuat model dan melatih model.

```
from sklearn.base import BaseEstimator, ClassifierMixin
class C45Wrapper(BaseEstimator, ClassifierMixin):
  def __init__(self):
      self.model = C45Classifier()

  def fit(self, X, y):
      self.model.fit(X, y)
      return self

  def predict(self, X):
      return self.model.predict(X)

  def fit_predict(self, X, y):
      self.fit(X, y)
      return self.predict(X)
```

```
# Membuat model Decision Tree C4.5
decision_tree_c45 = C45Wrapper()

# Melatih model pada data latih
decision_tree_c45.fit(X_train, y_train)
```

Pada kode tersebut, kelas C45Wrapper digunakan untuk membuat model Decision Tree C4.5. Kelas ini membungkus C45Classifier agar dapat digunakan dengan antarmuka dan *pipeline* dari scikit-learn. Setelah model dibuat, langkah berikutnya adalah melatih model menggunakan data latih dengan kode: decision_tree_c45.fit(X_train, y_train). Proses ini memungkinkan model untuk belajar dari data yang diberikan dan mengenali pola-pola yang ada. Setelah model dilatih, model ini dapat digunakan untuk membuat prediksi pada data baru dan mengevaluasi performanya.

3.2.4 Perancangan Aplikasi Web

Perancangan aplikasi web adalah proses yang melibatkan serangkaian langkah yang dirancang untuk mengembangkan aplikasi berbasis web yang *user-friendly*. Langkah-langkah ini mencakup definisi konsep aplikasi, pengembangan antarmuka pengguna, dan integrasi model.

a. Konsep Aplikasi

Konsep aplikasi mencakup beberapa komponen utama. Pertama, pengguna akan diminta untuk memasukkan data pribadi dan kesehatan mereka melalui formulir yang disediakan di aplikasi. Data ini meliputi informasi tentang usia, jenis kelamin, gejala, serta faktor-faktor risiko lain yang terkait dengan diabetes. Data yang di *input* oleh pengguna akan digunakan sebagai dasar untuk menghitung risiko diabetes. Aplikasi akan menggunakan algoritma decision tree yang telah dilatih sebelumnya dengan data historis untuk mengevaluasi faktor-faktor risiko ini. Berdasarkan evaluasi tersebut, aplikasi akan menghasilkan prediksi risiko diabetes. Hasil prediksi ini akan ditampilkan kepada pengguna dalam bentuk yang mudah dimengerti, seperti grafik atau papan skor, untuk memudahkan interpretasi. Selain hasil prediksi, aplikasi juga akan memberikan informasi tambahan seperti faktor-faktor utama yang mempengaruhi risiko diabetes dan saran pencegahan. Hal ini

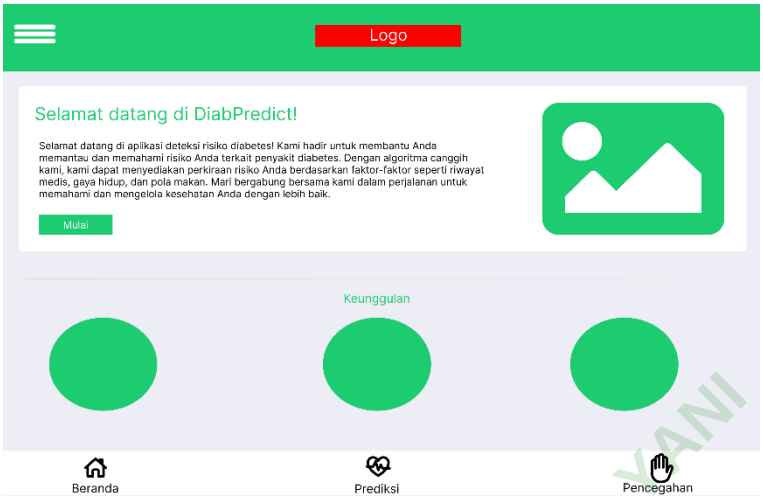
membantu pengguna untuk lebih memahami kondisi mereka dan langkah-langkah yang dapat diambil untuk mengurangi risiko. Aplikasi juga dapat menyediakan informasi tentang penyakit diabetes, saran gaya hidup sehat, dan langkah-langkah pencegahan yang dapat diambil oleh pengguna. Ini memastikan bahwa pengguna tidak hanya mendapatkan prediksi risiko, tetapi juga edukasi yang komprehensif tentang bagaimana menjaga kesehatan mereka.

b. Pengembangan Antarmuka Pengguna

Pengembangan antarmuka pengguna adalah langkah penting dalam perancangan aplikasi web. Antarmuka harus dirancang agar mudah digunakan, memungkinkan pengguna untuk dengan mudah memasukkan data yang diperlukan tanpa kebingungan. Hasil prediksi risiko diabetes harus disajikan dalam format yang mudah dipahami, seperti grafik, diagram, atau papan skor. Visualisasi ini membantu pengguna memahami hasil dengan cepat dan jelas. Selain itu, aplikasi harus menyediakan akses yang mudah ke informasi tambahan tentang diabetes, gaya hidup sehat, dan langkah-langkah pencegahan. Informasi ini harus tersusun dengan rapi dan mudah diakses oleh pengguna, sehingga memastikan mereka mendapatkan semua informasi yang dibutuhkan dari satu sumber. Berikut adalah desain antarmuka yang direncanakan untuk setiap halaman. Berikut rancangan desain antarmuka pada setiap halaman.

- Halaman Beranda

Halaman beranda adalah tampilan pertama yang dilihat pengguna saat mengakses sistem. Halaman ini dirancang untuk menyambut pengguna dan memberikan pengenalan singkat tentang tujuan serta manfaat dari sistem prediksi risiko diabetes. Berikut tampilan rancangan halaman beranda dapat dilihat pada Gambar 3.11.



Gambar 3.11 Desain halaman beranda

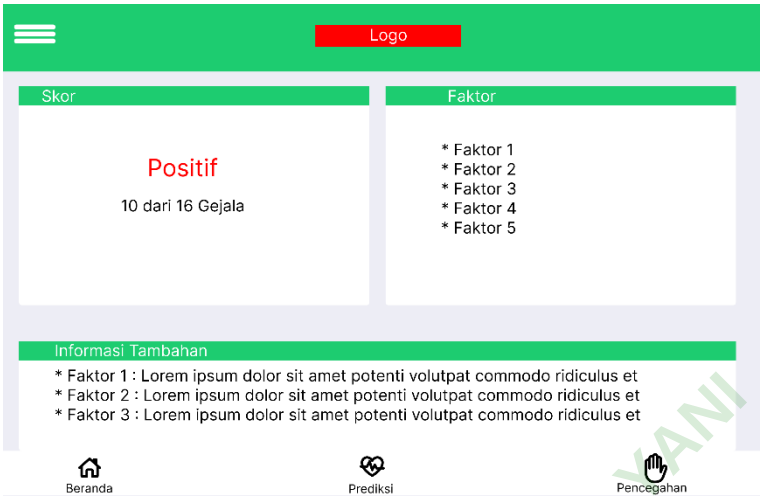
- Halaman Prediksi

Halaman ini memungkinkan pengguna untuk memasukkan data pribadi dan medis yang diperlukan untuk memprediksi risiko diabetes. Data yang dimasukkan akan diproses oleh model untuk menghasilkan prediksi. Berikut tampilan halaman prediksi dapat dilihat pada Gambar 3.12.

Gambar 3.12 Desain halaman prediksi

- Halaman Hasil

Halaman hasil menampilkan output dari model prediksi berdasarkan data yang telah dimasukkan oleh pengguna. Berikut tampilan halaman hasil dapat dilihat pada Gambar 3.13.



Gambar 3.13 Desain halaman hasil

- Halaman Pencegahan

Halaman ini menyediakan informasi mengenai langkah-langkah pencegahan yang dapat diambil untuk mengurangi risiko diabetes. Berikut tampilan halaman pencegahan dapat dilihat pada Gambar 3.14.

Gambar 3.14 Desain halaman pencegahan

c. Integrasi dengan Model

Integrasi model decision tree adalah langkah krusial dalam memastikan aplikasi berfungsi sesuai tujuan. Model decision tree yang telah dilatih harus diintegrasikan ke dalam aplikasi web untuk memungkinkan pengguna melakukan evaluasi risiko diabetes berdasarkan data yang mereka masukkan.

3.2.5 Pengujian Aplikasi

Langkah awal dalam melakukan pengujian usability ini adalah menyusun pernyataan-pernyataan untuk angket yang akan didistribusikan kepada responden pengguna aplikasi. Pernyataan-pernyataan ini didasarkan pada USE Questionnaire, yang terdiri dari 12 pernyataan yang dibagi ke dalam 4 faktor: kegunaan, kepuasan, kemudahan penggunaan, dan kemudahan pembelajaran, seperti yang tercantum dalam Tabel 3.2.

Tabel 3.2 Peryataan kuesioner

Faktor	No	Peryataan	STS	TS	S	SS
Usefulness	1	Aplikasi ini membantu saya memahami risiko diabetes saya				
	2	Aplikasi ini menyediakan informasi yang berguna tentang diabetes				
	3	Aplikasi ini membantu saya dalam mengambil langkah-langkah pencegahan diabetes				
Ease of Use	4	Aplikasi ini mudah digunakan				
	5	Antarmuka aplikasi ini menarik dan *user-friendly*				
	6	Interaksi dengan aplikasi ini terasa lancar dan bebas hambatan				
Ease of Learning	7	Saya dengan cepat dapat belajar bagaimana menggunakan aplikasi ini				
	8	Aplikasi ini mudah dipelajari bahkan bagi orang yang tidak terlalu teknis				
	9	Setelah beberapa kali penggunaan, saya merasa cukup nyaman menggunakan aplikasi ini				
Satisfaction	10	Saya puas dengan kinerja aplikasi ini secara keseluruhan				
	11	Aplikasi ini memenuhi harapan saya				

	12	Saya merasa nyaman menggunakan aplikasi ini untuk memantau risiko diabetes saya				

Keterangan:

STS = Sangat Tidak Setuju

TS = Tidak Setuju

S = Setuju

SS = Sangat Setuju

Rumus persamaan untuk menghitung tingkat usability menggunakan USE Questionnaire (Weichbroth, 2020) dapat dirujuk pada persamaan (5):

$$Pk\ (\%) = \frac{skor\ observasi}{skor\ ekspetasi} \times 100\% \qquad (5)$$

Keterangan:

P*k* (%) = Tingkat Usability dalam persen

Data yang diperoleh kemudian dikonversi berdasarkan tabel kategori kelayakan pada Tabel 3.3.

Tabel 3.3 Kategori Kelayakan Angka (%) Klasikasi (Hidayat et al., 2021)

Angka (%)	**Klasifikasi**
<21	Sangat Tidak Layak
21-40	Tidak Layak
41-60	Cukup
61-80	Layak
81-100	Sangat Layak