Laporan Pembelajaran Mesin

Tugas 3

Logistic Regression, K-NN, Naive Bayes dan Decision Tree

disusun untuk

memenuhi tugas mata kuliah

Pembelajaran Mesin

Oleh :

Kelompok VII

Nazwa Salsabila (2208107010010)

Rizky Yusmansyah (2208107010024)

Della Rahmatika (2208107010041)

Zuwi Pertiwi (2208107010061)

Berliani Utami (2208107010082)

JURUSAN INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SYIAH KUALA

DARUSSALAM, BANDA ACEH

2025

A. Data Deskripsi

Dataset yang digunakan adalah “Heart Disease Prediction – 2020 Cleaned” dari Github. Dataset ini berisi data kesehatan individu dengan berbagai parameter seperti BMI, status merokok, kategori usia, dan jenis kelamin. Target klasifikasi ini adalah kolom HeartDisease, yang menunjukkan apakah seseorang memiliki risiko penyakit jantung atau tidak(Yes/No). Jumlah total data adalah sekitar 319.795 sampel dengan berbagai fitur kategorikal dan numerik.

B. Data Loading

Untuk membantu dalam memahami data, ada beberapa library yang digunakan pada tahapan awal eksplorasi data yaitu mengimport pandas, numpy, matpolib dan seaborn. Dataset kemudian dimuat ke dalam lingkungan pemrograman melalui URL dataset yang sudah di upload ke repository Github.

```
# URL dataset
url = "https://raw.githubusercontent.com/rizkyyus/Kelompok_7_Tugas03_Classification/refs/heads/main/heart_2020_cleaned.csv"
```

Gambar 1 Menambahkan dataset melalui URL ke dalam lingkungan pemrograman

C. Data Understanding

Pemahaman awal pada dataset adalah dengan menampilkan statistic dasar dataset serta menampilkan visualisasi sederhana dari dataset tersebut.

1. Menampilkan Struktur Dataset

a. Informasi Umum Dataset

Dataset Heart Disease Prediction terdiri dari 14 kolom yang mencakup berbagai fitur terkait kesehatn individu, seperti umur, jenis kelamin, Tingkat kolestrol, tekanan darah, dan apakah seseorang memiliki penyakit jantung atau tidak. Dataset ini memiliki 319.795 sampel, dengan beberapa kolom memiliki nilai numerik (seperti umur, tekanan darah, dan kolestrol) dan kategorikal (seperti jenis kelamin atau status merokok).

```
# Menampilkan beberapa data awal
print(df.head())
```

```
  HeartDisease    BMI Smoking AlcoholDrinking Stroke  PhysicalHealth  \
0           No  16.60     Yes              No     No             3.0
1           No  20.34      No              No    Yes             0.0
2           No  26.58     Yes              No     No            20.0
3           No  24.21      No              No     No             0.0
4           No  23.71      No              No     No            28.0

   MentalHealth DiffWalking     Sex  AgeCategory   Race Diabetic  \
0          30.0          No  Female        55-59  White      Yes
1           0.0          No  Female  80 or older  White       No
2          30.0          No    Male        65-69  White      Yes
3           0.0          No  Female        75-79  White       No
4           0.0         Yes  Female        40-44  White       No

  PhysicalActivity  GenHealth  SleepTime Asthma KidneyDisease SkinCancer
0              Yes  Very good        5.0    Yes            No        Yes
1              Yes  Very good        7.0     No            No         No
2              Yes       Fair        8.0    Yes            No         No
3               No       Good        6.0     No            No        Yes
4              Yes  Very good        8.0     No            No         No
```

Gambar 2 Menampilkan beberapa data awal

b. Informasi Struktur Dataset

Dataset ini memiliki 14 kolom dan 319.795 sampel. Tipe data pada dataset ini terdiri dari int64 untuk data numerik (seperti Age, RestingBP, Cholestrol) dan object untuk data kategorikal (seperti sex, CgestPainType, RestingECG). Semua kolom memiliki 319.795 nilai non-null, sehingga tidak ada nilai yang hilang (missing values).

```
# Informasi dataset
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 319795 entries, 0 to 319794
Data columns (total 18 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   HeartDisease      319795 non-null  object
 1   BMI               319795 non-null  float64
 2   Smoking           319795 non-null  object
 3   AlcoholDrinking   319795 non-null  object
 4   Stroke            319795 non-null  object
 5   PhysicalHealth    319795 non-null  float64
 6   MentalHealth      319795 non-null  float64
 7   DiffWalking       319795 non-null  object
 8   Sex               319795 non-null  object
 9   AgeCategory       319795 non-null  object
 10  Race              319795 non-null  object
 11  Diabetic          319795 non-null  object
 12  PhysicalActivity  319795 non-null  object
 13  GenHealth         319795 non-null  object
 14  SleepTime         319795 non-null  float64
 15  Asthma            319795 non-null  object
 16  KidneyDisease     319795 non-null  object
 17  SkinCancer        319795 non-null  object
dtypes: float64(4), object(14)
memory usage: 43.9+ MB
None
```

Gambar 3 Menampilkan Informasi Dataset

c. Statistik Deskriptif

Dataset ini menunjukkan adanya variasi yang cukup besar pada beberapa fitur numerik, seperti kolestrol, tekanan darah, dan detak jantung maksimum. Kolom seperti Age dan MaxHR menunjukkan kisaran nilai yang lebar, sementara kolom Cholestrol dan Oldpeak memiliki distribusi yang lebih terpusat pada nilai Tengah.

```
# Statistik deskriptif
print(df.describe())
```

```
                BMI   PhysicalHealth    MentalHealth      SleepTime
count  319795.000000    319795.00000  319795.000000  319795.000000
mean       28.325399         3.37171       3.898366       7.097075
std         6.356100         7.95085       7.955235       1.436007
min        12.020000         0.00000       0.000000       1.000000
25%        24.030000         0.00000       0.000000       6.000000
50%        27.340000         0.00000       0.000000       7.000000
75%        31.420000         2.00000       3.000000       8.000000
max        94.850000        30.00000      30.000000      24.000000
```

Gambar 4 Menampilkan Statistik Deskriptif

d. Mengecek Missing Values

Disini memeriksa adanya missing values pada dataset menggunakan **isnull()> sum().** Hasilnya menunjukkan bahwa tidak ada kolom yang memiliki missing values, artinya setiap kolom memiliki nilai lengkap tanpa data yang hilang.

```
# Memeriksa missing values
print("Missing values per kolom:")
print(df.isnull().sum())
```

```
Missing values per kolom:
HeartDisease        0
BMI                 0
Smoking             0
AlcoholDrinking     0
Stroke              0
PhysicalHealth      0
MentalHealth        0
DiffWalking         0
Sex                 0
AgeCategory         0
Race                0
Diabetic            0
PhysicalActivity    0
GenHealth           0
SleepTime           0
Asthma              0
KidneyDisease       0
SkinCancer          0
dtype: int64
```

Gambar 5 Mengecek Missing Values

e. Mengecek dan Mengonversi Tipe Data

Pengecekan tipe data pada dataset menunjukkan bahwa sebagian besar kolom kategorikal masih berstatus "object", sementara kolom numerik sudah memiliki tipe data yang sesuai. Untuk melanjutkan ke tahap pemodelan, kolom-kolom kategorikal

perlu dikonversi menjadi tipe data numerik menggunakan teknik seperti *Label Encoding* atau *One-Hot Encoding*. Dengan konversi yang tepat, data dapat diproses lebih lanjut dan digunakan dalam model machine learning untuk klasifikasi.

```
# Mengecek tipe data dan melakukan konversi jika diperlukan
print("\nTipe data masing-masing kolom:")
print(df.dtypes)
```

```
Tipe data masing-masing kolom:
HeartDisease         object
BMI                 float64
Smoking              object
AlcoholDrinking      object
Stroke               object
PhysicalHealth      float64
MentalHealth        float64
DiffWalking          object
Sex                  object
AgeCategory          object
Race                 object
Diabetic             object
PhysicalActivity     object
GenHealth            object
SleepTime           float64
Asthma               object
KidneyDisease        object
SkinCancer           object
dtype: object
```

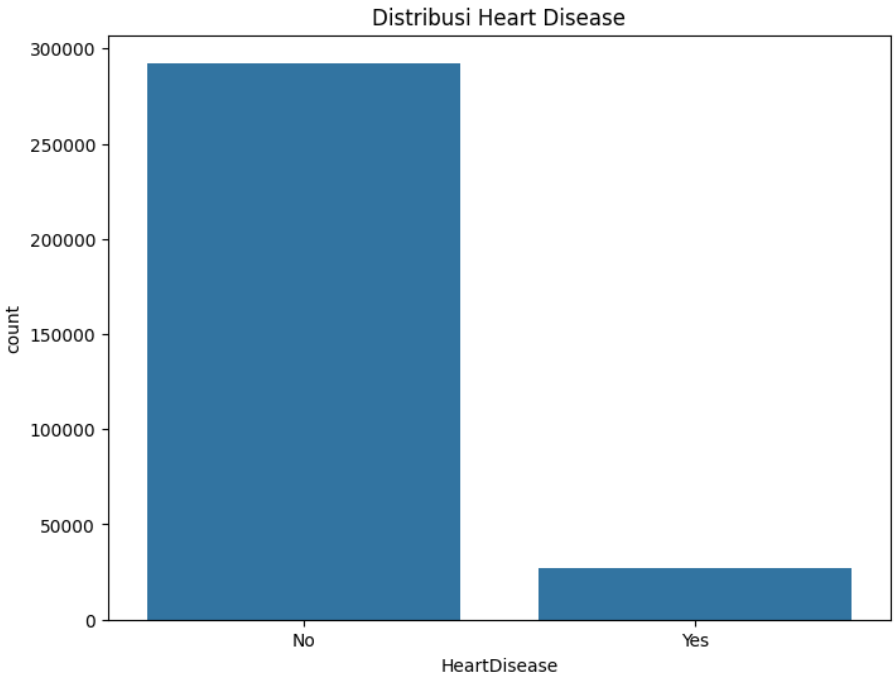
Gambar 6 Mengecek dan Mengonversi Tipe Data

2. Statistik Deskriptif dan Visualisasi Awal

a. Memeriksa Distribusi Kelas Target

```
# Memeriksa distribusi kelas target
plt.figure(figsize=(8, 6))
sns.countplot(x='HeartDisease', data=df)
plt.title('Distribusi Heart Disease')
plt.show()
```

Gambar 7 Memeriksa Deskriptif dan Visualisasi Awal



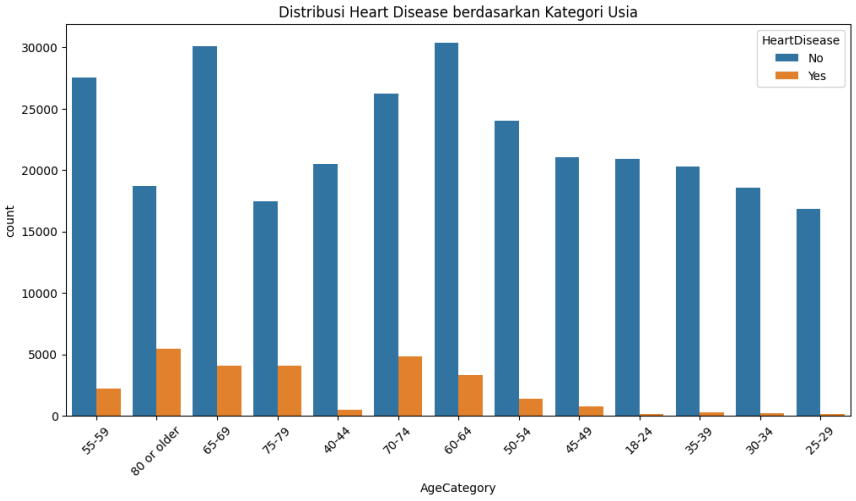
Gambar 7.1 Visualisasi Distribusi Heart Disease

Visualisasi diatas menunjukkan distribusi data Heart Disease menunjukkan bahwa mayoritas individu dalam dataset tidak menderita penyakit jantung, sedangkan jumlah penderita jauh lebih sedikit. Hal ini menandakan ketidakseimbangan kelas yang dapat memengaruhi performa model prediksi, sehingga perlu penanganan khusus.

b. Cek Distribusi Usia

```
# Cek distribusi usia
plt.figure(figsize=(12, 6))
sns.countplot(x='AgeCategory', data=df, hue='HeartDisease')
plt.title('Distribusi Heart Disease berdasarkan Kategori Usia')
plt.xticks(rotation=45)
plt.show()
```

Gambar 8 Cek Distribusi Usia



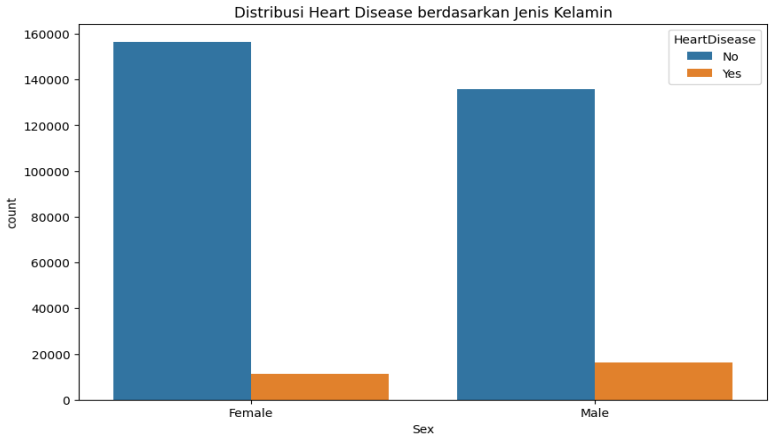
Gambar 8.1 Visualisasi Heart Disease Usia

Distribusi penyakit jantung berdasarkan kategori usia menunjukkan bahwa proporsi penderita penyakit jantung meningkat secara signifikan pada kelompok usia yang lebih tua, terutama pada usia **55 tahun ke atas**. Sebaliknya, pada kelompok usia yang lebih muda (khususnya di bawah 40 tahun), jumlah penderita sangat rendah. Ini mengindikasikan bahwa risiko penyakit jantung cenderung meningkat seiring bertambahnya usia.

c. Distribusi Berdasarkan Jenis Kelamin

```
# Distribusi berdasarkan jenis kelamin
plt.figure(figsize=(10, 6))
sns.countplot(x='Sex', data=df, hue='HeartDisease')
plt.title('Distribusi Heart Disease berdasarkan Jenis Kelamin')
plt.show()
```

Gambar 9 Distribusi Berdasarkan Jenis Kelamin



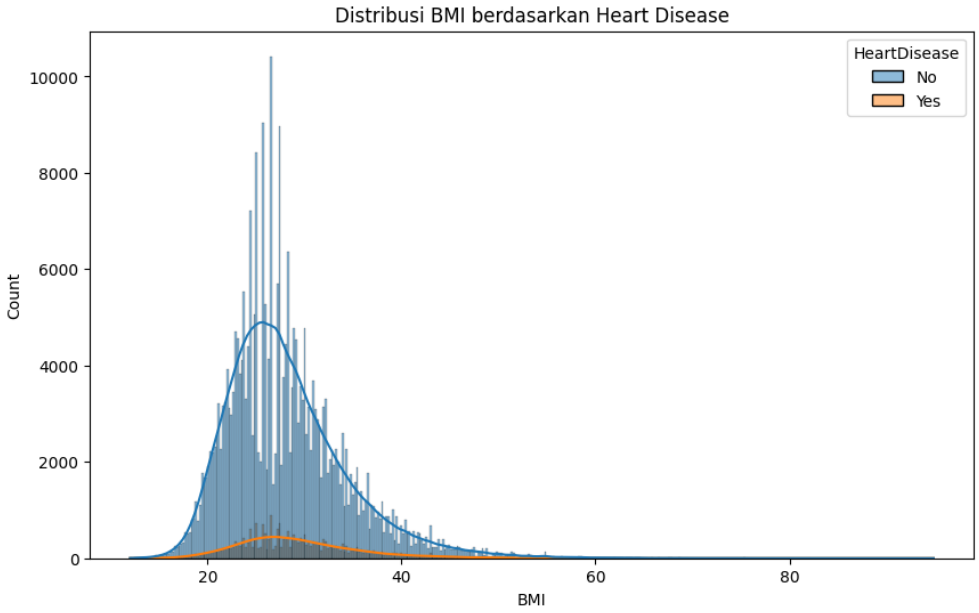
Gambar 9.1 Visualisasi Heart Disease Jenis Kelamin

Distribusi penyakit jantung berdasarkan jenis kelamin menunjukkan bahwa **laki-laki (Male)** memiliki jumlah penderita penyakit jantung yang **lebih tinggi** dibandingkan perempuan (Female), meskipun jumlah keseluruhan responden perempuan lebih banyak. Hal ini menunjukkan bahwa **risiko penyakit jantung cenderung lebih besar pada laki-laki** dibandingkan perempuan dalam dataset ini.

d. Distribusi BMI

```
# Distribusi BMI
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='BMI', hue='HeartDisease', kde=True)
plt.title('Distribusi BMI berdasarkan Heart Disease')
plt.show()
```

Gambar 10 Distribusi BMI



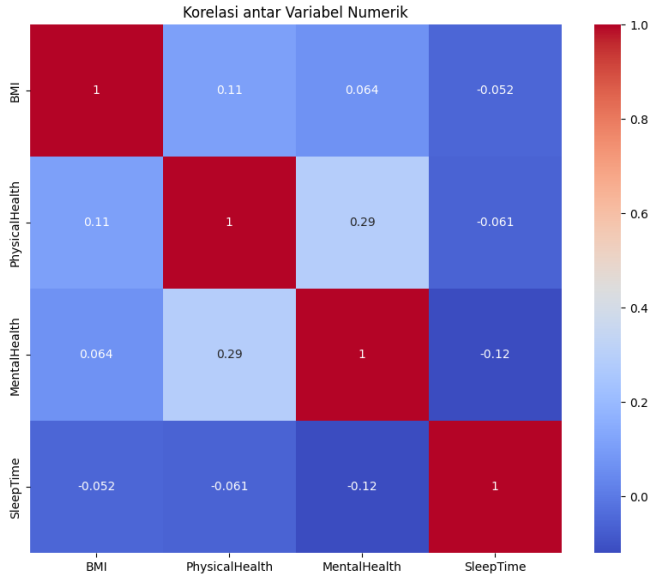
Gambar 10.1 Visualisasi Heart Disease

Distribusi BMI berdasarkan penyakit jantung menunjukkan bahwa sebagian besar individu, baik yang memiliki penyakit jantung maupun tidak, memiliki BMI di kisaran **20 hingga 40**. Namun, **penderita penyakit jantung (warna oranye)** cenderung mulai muncul lebih banyak pada rentang BMI yang lebih tinggi, menunjukkan bahwa **semakin tinggi nilai BMI, risiko terkena penyakit jantung juga meningkat**, meskipun proporsinya tetap lebih kecil dibandingkan yang tidak memiliki penyakit jantung.

e. Korelasi antara Variabel Numerik

```
# Korelasi antara variabel numerik
numerical_features = ['BMI', 'PhysicalHealth', 'MentalHealth', 'SleepTime']
plt.figure(figsize=(10, 8))
sns.heatmap(df[numerical_features].corr(), annot=True, cmap='coolwarm')
plt.title('Korelasi antar Variabel Numerik')
plt.show()
```

Gambar 11 Korelasi antar Variabel Numerik



Gambar 11.1 Visualisasi Korelasi antar Variabel Numerik

Berdasarkan visualisasi, korelasi antar variabel numerik seperti **BMI, PhysicalHealth, MentalHealth, dan SleepTime** menunjukkan hubungan yang **lemah atau sangat lemah**. Korelasi tertinggi terdapat antara **PhysicalHealth dan MentalHealth (0.29)**, yang berarti semakin buruk kesehatan fisik, cenderung semakin buruk pula kesehatan mental seseorang. Namun, semua nilai korelasi berada di bawah 0.3, menunjukkan **tidak ada hubungan linier yang kuat antara variabel-variabel tersebut**.

D. Eksplorasi Data dan Pra-pemrosesan

1. Encoding Variabel Kategorikal

```
# Encoding variabel kategorikal
categorical_features = ['Smoking', 'AlcoholDrinking', 'Stroke', 'DiffWalking', 'Sex',
                        'AgeCategory', 'Race', 'Diabetic', 'PhysicalActivity',
                        'GenHealth', 'Asthma', 'KidneyDisease', 'SkinCancer']
```

Gambar 11 Encoding Variabel Kategorikal

Kolom-kolom kategorikal dalam dataset, seperti 'Smoking', 'AlcoholDrinking', 'Stroke', dan seterusnya, memerlukan konversi menjadi data numerik agar dapat diproses oleh model machine learning. Karena model ini hanya dapat bekerja dengan data berbentuk angka, kita akan menggunakan teknik encoding, seperti Label Encoding atau One-Hot Encoding, untuk mengubah kategori menjadi nilai numerik yang dapat dimengerti oleh model.

2. Membuat Encoding

```
# Membuat encoder
le = LabelEncoder()
```

Gambar 12 Membuat Encoder

Membuat sebuah objek encoder menggunakan LabelEncoder dari pustaka **scikit-learn**. LabelEncoder digunakan untuk mengubah variabel kategorikal menjadi numerik dengan mengganti setiap kategori dengan angka unik.

3. Membuat Salinan data frame

```
# Buat salinan dataframe untuk menghindari warning
df_encoded = df.copy()
```

Gambar 13 Membuat Salinan data frame

Untuk menghindari peringatan (warning), kita membuat salinan dataframe asli dengan menyalinnya ke dalam variabel df_encoded. Ini memastikan bahwa perubahan yang dilakukan pada df_encoded tidak mempengaruhi dataframe asli (df).

4. Encode Target Variabel

```
# Encode target variable
df_encoded['HeartDisease'] = le.fit_transform(df_encoded['HeartDisease'])
```

Gambar 14 Encode Target Variabel

Setiap kolom kategorikal dalam daftar categorical_features di-encode menggunakan LabelEncoder. Proses ini mengganti setiap kategori dalam kolom dengan angka yang sesuai. Dengan demikian, variabel kategorikal yang sebelumnya berbentuk teks kini sudah dalam bentuk numerik dan siap digunakan dalam analisis lebih lanjut atau pemodelan.

5. Cek Hasil Encoding

```
# Cek hasil encoding
print("\nDataframe setelah encoding:")
print(df_encoded.head())
```

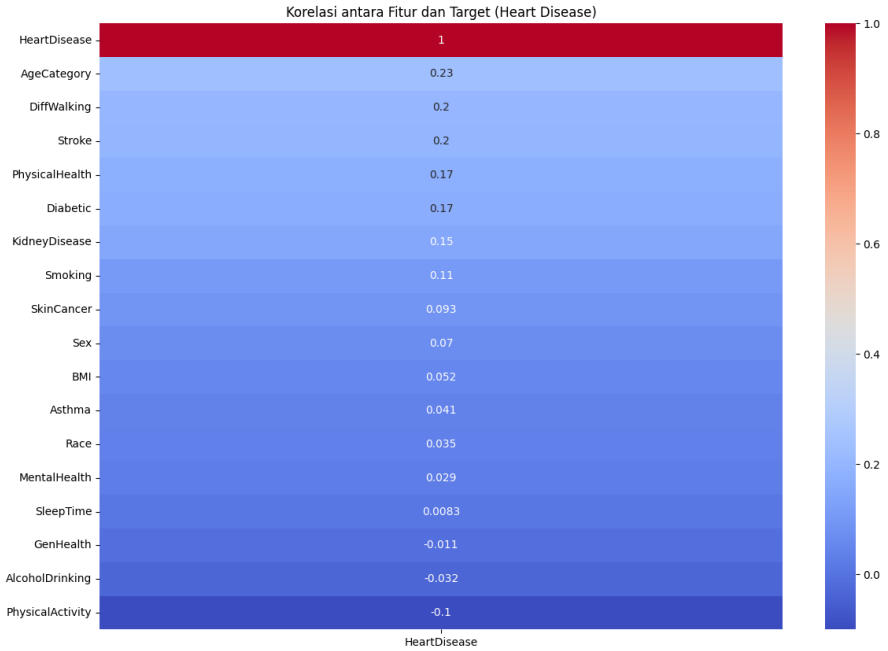
```
Dataframe setelah encoding:
   HeartDisease    BMI  Smoking  AlcoholDrinking  Stroke  PhysicalHealth  \
0             0  16.60        1                0       0            3.0
1             0  20.34        0                0       1            0.0
2             0  26.58        1                0       0           20.0
3             0  24.21        0                0       0            0.0
4             0  23.71        0                0       0           28.0

   MentalHealth  DiffWalking  Sex  AgeCategory  Race  Diabetic  \
0          30.0            0    0            7     5         2
1           0.0            0    0           12     5         0
2          30.0            0    1            9     5         2
3           0.0            0    0           11     5         0
4           0.0            1    0            4     5         0

   PhysicalActivity  GenHealth  SleepTime  Asthma  KidneyDisease  SkinCancer
0                 1          4        5.0       1              0           1
1                 1          4        7.0       0              0           0
2                 1          1        8.0       1              0           0
3                 0          2        6.0       0              0           1
4                 1          4        8.0       0              0           0
```

Gambar 15 Cek Hasil Encoding

6. Analisis Korelasi dengan Target

```
# Analisis korelasi dengan target
plt.figure(figsize=(14, 10))
correlation = df_encoded.corr()['HeartDisease'].sort_values(ascending=False)
sns.heatmap(correlation.to_frame(), annot=True, cmap='coolwarm')
plt.title('Korelasi antara Fitur dan Target (Heart Disease)')
plt.show()
```

Gambar 16 Menganalisis korelasi dengan tepat



Gambar 16.1 Visualisasi Korelasi antar fitur dan target

Hasil analisis menunjukkan beberapa temuan kunci terkait penyakit jantung. Meskipun jumlah perempuan lebih banyak, laki-laki memiliki risiko lebih tinggi terkena penyakit jantung. Mayoritas individu memiliki BMI normal hingga overweight, dengan penderita penyakit jantung cenderung memiliki BMI sedikit lebih tinggi. Tidak ada korelasi kuat antar variabel numerik seperti BMI, SleepTime, PhysicalHealth, dan MentalHealth. Faktor usia, kesulitan berjalan, dan riwayat stroke memiliki pengaruh paling besar terhadap penyakit jantung, diikuti oleh kondisi fisik dan diabetes. Menariknya, aktivitas fisik berhubungan negatif dengan penyakit jantung, yang berarti semakin tinggi aktivitas fisik, semakin kecil risikonya. Faktor-faktor seperti jenis kelamin, tidur, dan BMI memiliki korelasi lemah dengan penyakit jantung.

7. Split Data Fitur dan Target

```
# Split data into features and target
X = df_encoded.drop('HeartDisease', axis=1)
y = df_encoded['HeartDisease']
```

Gambar 17 Split fitur dan target

Dataset dibagi menjadi dua bagian: fitur dan target. Fitur (X) terdiri dari semua kolom selain **HeartDisease**, yang berfungsi sebagai input untuk model prediksi, sedangkan **HeartDisease** dipilih sebagai target (y), yaitu variabel yang ingin diprediksi. Pembagian ini penting agar model machine learning dapat mempelajari hubungan antara fitur dan target untuk melakukan klasifikasi penyakit jantung pada individu.

8. Train test-split

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Gambar 18 Train test-split

Data dibagi menjadi dua subset, yaitu data latih (**X_train**, **y_train**) dan data uji (**X_test**, **y_test**), dengan proporsi 80% untuk pelatihan dan 20% untuk pengujian. Pembagian ini bertujuan untuk melatih model menggunakan data pelatihan dan mengevaluasi kinerjanya pada data yang belum pernah dilihat sebelumnya (data uji), guna menghindari overfitting dan menilai kemampuan generalisasi model.

9. Standarisasi Fitur Numerik

```
# Standardisasi fitur numerik
scaler = StandardScaler()
numerical_features = ['BMI', 'PhysicalHealth', 'MentalHealth', 'SleepTime']
X_train[numerical_features] = scaler.fit_transform(X_train[numerical_features])
X_test[numerical_features] = scaler.transform(X_test[numerical_features])
```

Gambar 19 Standarisasi fitur numerik

E. Implementasi Model dan Evaluasi

a. Model 1 : Logistic Regression

```
# Inisialisasi model Logistic Regression
lr_model = LogisticRegression(max_iter=1000, random_state=42)

# Train model
lr_model.fit(X_train, y_train)

# Prediksi pada data test
y_pred_lr = lr_model.predict(X_test)
y_prob_lr = lr_model.predict_proba(X_test)[:, 1]

# Cross-validation
cv_scores_lr = cross_val_score(lr_model, X, y, cv=5, scoring='accuracy')
```

Gambar 20 Logistic Regression

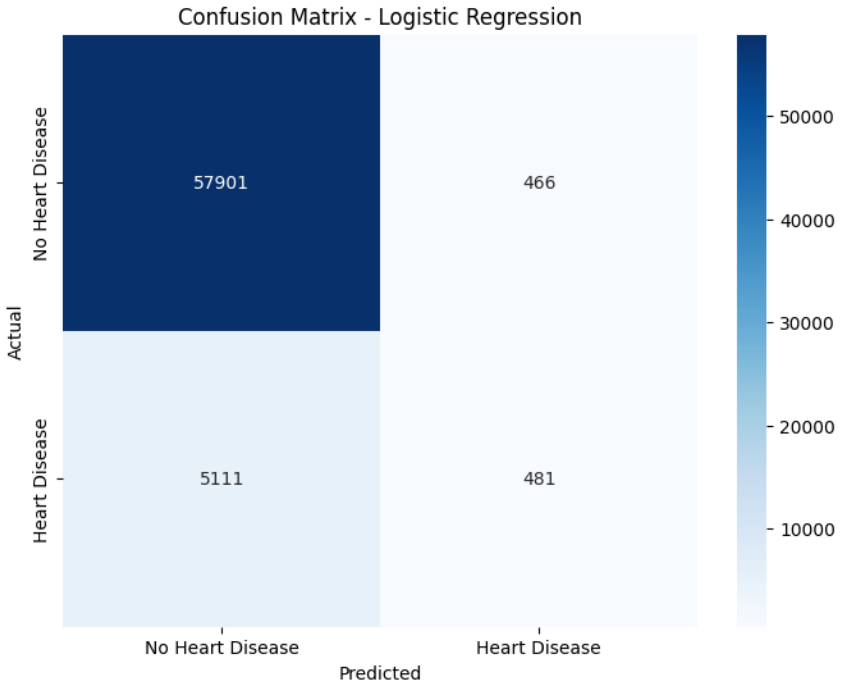
Kode ini digunakan untuk melatih dan mengevaluasi model regresi logistik. Model diinisialisasi dengan max_iter=1000 agar konvergen dan random_state=42 untuk hasil yang konsisten. Setelah dilatih dengan data pelatihan, model digunakan untuk memprediksi label dan probabilitas pada data uji. Terakhir, cross-validation dengan 5 lipatan digunakan untuk mengevaluasi akurasi model pada dataset secara keseluruhan.

a.1 Evaluasi Model Logistic Regression

```
# Evaluasi model
print("Model Logistic Regression:")
print("Accuracy:", accuracy_score(y_test, y_pred_lr))
print("\nConfusion Matrix:")
cm_lr = confusion_matrix(y_test, y_pred_lr)
print(cm_lr)

# Visualisasi confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm_lr, annot=True, fmt='d', cmap='Blues',
            xticklabels=['No Heart Disease', 'Heart Disease'],
            yticklabels=['No Heart Disease', 'Heart Disease'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Logistic Regression')
plt.show()

# Classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred_lr))

# ROC Curve
fpr_lr, tpr_lr, _ = roc_curve(y_test, y_prob_lr)
auc_lr = roc_auc_score(y_test, y_prob_lr)

plt.figure(figsize=(8, 6))
plt.plot(fpr_lr, tpr_lr, label=f'Logistic Regression (AUC = {auc_lr:.3f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Logistic Regression')
plt.legend()
plt.show()
```

Gambar 20.1 Evaluasi model logistic regression

- **Confusion Matrix**

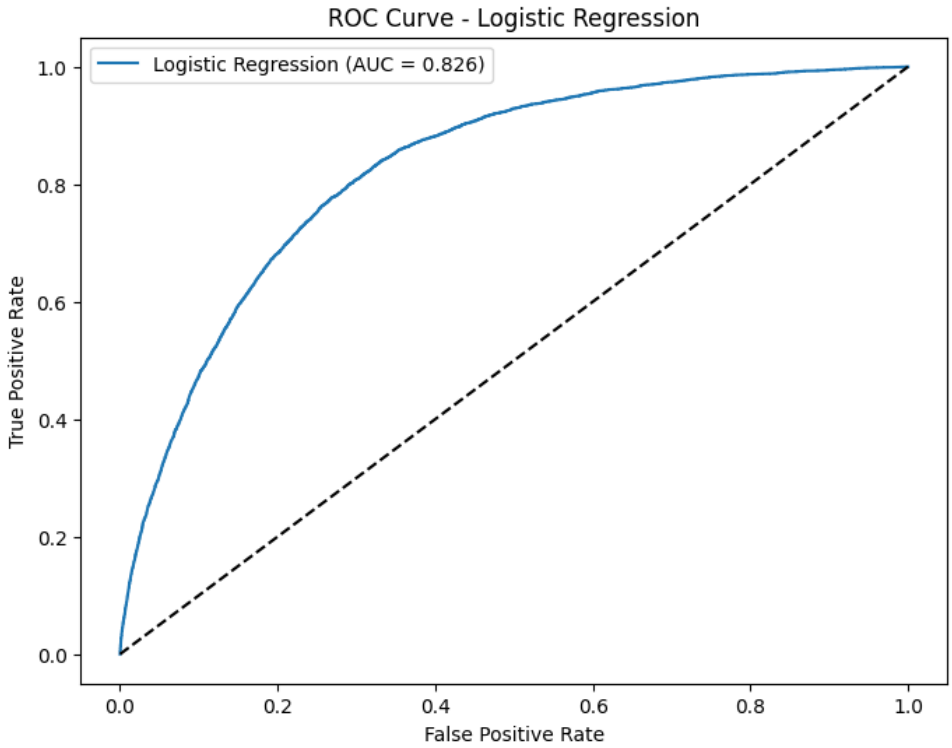



```
Model Logistic Regression:
Accuracy: 0.9128035147516378

Confusion Matrix:
[[57901   466]
 [ 5111   481]]
```

Gambar 20.2 Confusion Matrix – Logistic regression

Gambar di atas menunjukkan evaluasi model **Logistic Regression** untuk klasifikasi penyakit jantung menggunakan **confusion matrix**. Model memiliki akurasi tinggi sebesar **91,28%**, dengan prediksi benar sebanyak 57.901 kasus tanpa penyakit dan 481 kasus dengan penyakit jantung. Namun, model juga salah mengklasifikasikan 5.111 kasus penyakit jantung sebagai tidak sakit (False Negative), menunjukkan kelemahan serius dalam mendeteksi kasus positif. Kesimpulannya, meskipun akurasi tinggi, model kurang efektif dalam mendeteksi penyakit jantung akibat ketidakseimbangan data. Diperlukan evaluasi lanjutan dengan precision, recall, dan F1-score, serta penanganan data imbalance agar hasil lebih akurat dan dapat diandalkan dalam konteks medis.

- **Roc Curve – Logistic Regression**



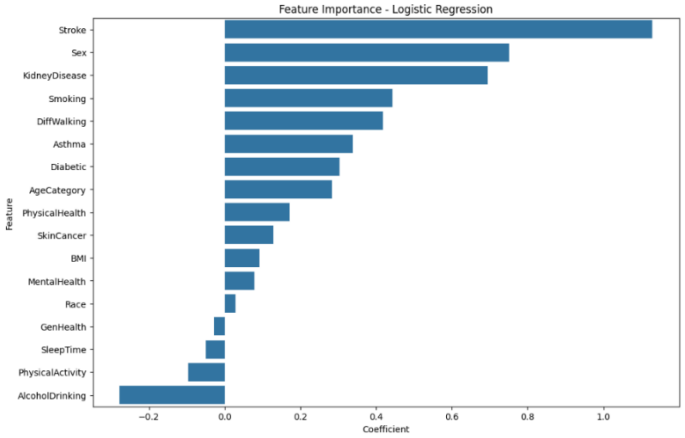
Classification Report:

	precision	recall	f1-score	support
0	0.92	0.99	0.95	58367
1	0.51	0.09	0.15	5592
accuracy			0.91	63959
macro avg	0.71	0.54	0.55	63959
weighted avg	0.88	0.91	0.88	63959

Gambar 20.3 Roc Curve – Logistic Regression

Meskipun model Logistic Regression menunjukkan akurasi tinggi sebesar 91% dan AUC sebesar 0,826, performanya dalam mendeteksi kasus penyakit jantung sangat buruk. Hal ini terlihat dari nilai recall yang sangat rendah (0,09) dan f1-score yang rendah (0,15) untuk kelas positif (penyakit jantung). Model sangat bias terhadap kelas mayoritas (tidak sakit), yang membuatnya tidak andal dalam konteks medis. Oleh karena itu, perlu perbaikan dengan menangani ketidakseimbangan data dan mengevaluasi ulang model menggunakan metrik seperti recall dan f1-score yang lebih representatif.

- **Feature Importance – Logistic Regression**




```
Log Loss: 0.2376
Cross-validation accuracy: 0.9150 (±0.0005)
```

Gambar 20.4 Feature Importance – Logistic Regression

Gambar diatas menunjukkan hasil analisis feature importance dari model Logistic Regression dalam memprediksi penyakit jantung. Fitur yang paling berpengaruh terhadap prediksi positif penyakit jantung adalah riwayat stroke, jenis kelamin, penyakit ginjal, dan kebiasaan merokok. Koefisien positif menunjukkan bahwa keberadaan fitur tersebut meningkatkan kemungkinan prediksi penyakit jantung, sedangkan fitur seperti konsumsi alkohol, aktivitas fisik, dan waktu tidur memiliki koefisien negatif, artinya cenderung menurunkan risiko menurut model. Selain itu, model menunjukkan performa yang cukup baik secara keseluruhan dengan nilai log loss sebesar 0.2376 yang mengindikasikan prediksi probabilitas yang cukup akurat, serta akurasi rata-rata cross-validation sebesar 91,5% dengan deviasi sangat kecil (±0.0005), yang mencerminkan kestabilan model saat diuji pada berbagai subset data. Namun, meskipun fitur-fitur penting telah teridentifikasi dengan baik, performa model tetap perlu dievaluasi secara menyeluruh mengingat kelemahan sebelumnya dalam mengenali kasus penyakit jantung (kelas minoritas).

b. Model 2: K-Nearest Neighbors (KNN)

```python
# Inisialisasi model KNN
knn_model = KNeighborsClassifier(n_neighbors=5)

# Train model
knn_model.fit(X_train, y_train)

# Prediksi pada data test
y_pred_knn = knn_model.predict(X_test)
y_prob_knn = knn_model.predict_proba(X_test)[:, 1]

# Cross-validation
cv_scores_knn = cross_val_score(knn_model, X, y, cv=5, scoring='accuracy')
```

Gambar 21 K- Nearest Neighbors

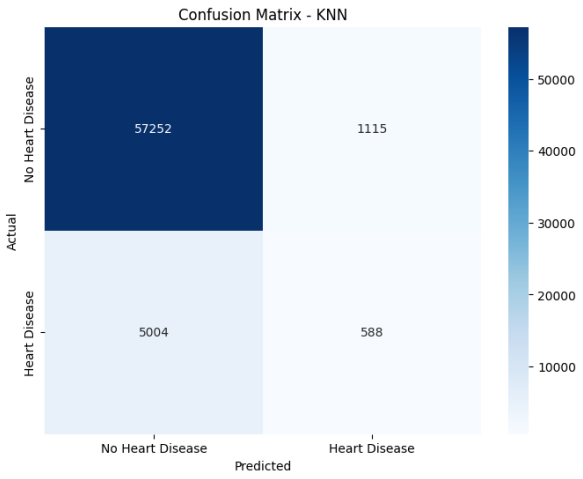
Kode di atas menginisialisasi model **K-Nearest Neighbors (KNN)** dengan n_neighbors=5, melatihnya menggunakan data latih (X_train, y_train), lalu memprediksi label dan probabilitas pada data uji (X_test). Model dievaluasi dengan **cross-validation** 5 kali menggunakan akurasi, dan hasilnya disimpan dalam cv_scores_knn untuk menilai kestabilan model.

b.1 Evaluasi Model K-Nearest Neighbors

```
# Evaluasi model
print("Model K-Nearest Neighbors:")
print("Accuracy:", accuracy_score(y_test, y_pred_knn))
print("\nConfusion Matrix:")
cm_knn = confusion_matrix(y_test, y_pred_knn)
print(cm_knn)

# Visualisasi confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm_knn, annot=True, fmt='d', cmap='Blues',
            xticklabels=['No Heart Disease', 'Heart Disease'],
            yticklabels=['No Heart Disease', 'Heart Disease'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - KNN')
plt.show()

# Classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred_knn))

# ROC Curve
fpr_knn, tpr_knn, _ = roc_curve(y_test, y_prob_knn)
auc_knn = roc_auc_score(y_test, y_prob_knn)

plt.figure(figsize=(8, 6))
plt.plot(fpr_knn, tpr_knn, label=f'KNN (AUC = {auc_knn:.3f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - KNN')
plt.legend()
plt.show()
```

Gambar 21. 1 Evaluasi model KNN

- **Confusion Matrix - KNN**

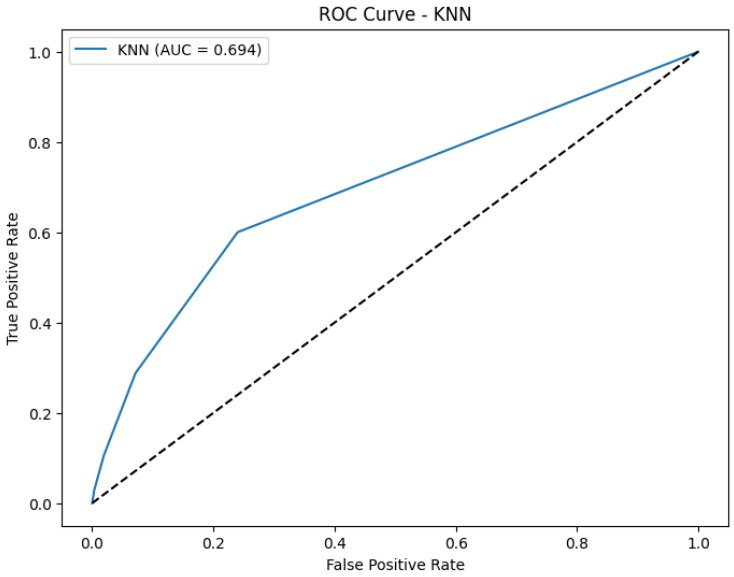



```
Model K-Nearest Neighbors:
Accuracy: 0.9043293359808627

Confusion Matrix:
[[57252  1115]
 [ 5004   588]]
```

Gambar 21.2 Confusion Matrix – K-Nearest Neighbors

Berdasarkan hasil evaluasi model K-Nearest Neighbors (KNN), diperoleh akurasi sebesar 90,43%, yang menunjukkan bahwa secara keseluruhan model mampu mengklasifikasikan data dengan tingkat ketepatan yang cukup tinggi. Namun, ketika dilihat lebih dalam melalui confusion matrix, terlihat bahwa model jauh lebih baik dalam mendeteksi kasus tidak ada penyakit jantung (True Negative sebanyak 57.252) dibandingkan mendeteksi kasus penyakit jantung (True Positive hanya 588). Terdapat 5.004 kasus penyakit jantung yang tidak berhasil terdeteksi dengan benar (False Negative), yang berarti model sering gagal mengenali individu yang sebenarnya menderita penyakit jantung. Hal ini merupakan kelemahan serius, terutama dalam konteks medis, karena kesalahan tersebut dapat berdampak pada keterlambatan penanganan pasien. Oleh karena itu, meskipun akurasinya tinggi, model ini masih kurang andal untuk mendeteksi penyakit jantung secara tepat, dan perlu dievaluasi lebih lanjut menggunakan metrik lain seperti precision, recall, dan F1-score.

- **Roc Curve - KNN**



Classification Report:

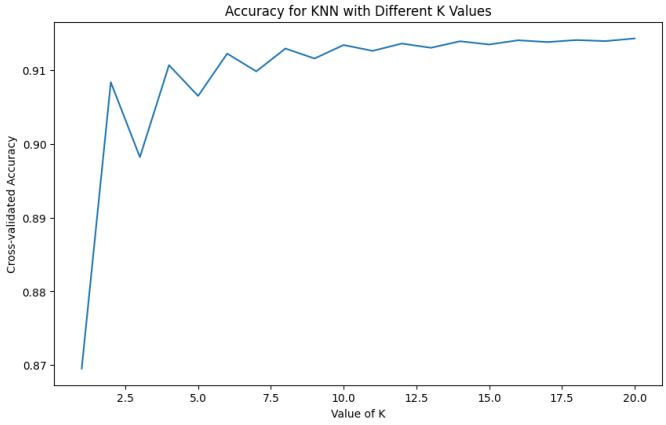
	precision	recall	f1-score	support
0	0.92	0.98	0.95	58367
1	0.35	0.11	0.16	5592
accuracy			0.90	63959
macro avg	0.63	0.54	0.56	63959
weighted avg	0.87	0.90	0.88	63959

Gambar 21.3 ROC Curve – KNN

Berdasarkan grafik ROC dan classification report untuk model K-Nearest Neighbors (KNN), dapat disimpulkan bahwa performa model cukup tinggi dalam mengidentifikasi kelas mayoritas, yaitu kasus tanpa penyakit jantung (kelas 0), dengan precision sebesar

0.92 dan recall sebesar 0.98. Hal ini menunjukkan bahwa model sangat andal dalam mengenali individu yang memang tidak menderita penyakit jantung. Namun, performa model dalam mendeteksi kasus penyakit jantung (kelas 1) sangat rendah, dengan recall hanya sebesar 0.11 dan f1-score sebesar 0.16. Artinya, dari seluruh kasus penyakit jantung yang ada, hanya sekitar 11% yang berhasil dikenali dengan benar oleh model. Skor AUC sebesar 0.694 juga menunjukkan bahwa kemampuan model dalam membedakan antara kedua kelas masih terbatas. Meskipun akurasi total model mencapai 90%, metrik tersebut menjadi kurang bermakna karena adanya ketidakseimbangan data (jumlah kelas 0 jauh lebih besar daripada kelas 1). Oleh karena itu, dapat disimpulkan bahwa model KNN ini kurang cocok untuk diterapkan dalam deteksi penyakit jantung karena cenderung gagal mengidentifikasi kasus positif, yang justru sangat penting dalam konteks kesehatan. Diperlukan perbaikan model atau pendekatan lain, seperti penyeimbangan data atau algoritma klasifikasi yang lebih sensitif terhadap kelas minoritas.

- **Evaluasi Performa Model K-Nearest Neighbors (KNN)**




```
Log Loss: 1.4112
Cross-validation accuracy: 0.9065 (±0.0006)
```

Gambar 21.3 Evaluasi model performa KNN

Gambar di atas menunjukkan evaluasi performa model K-Nearest Neighbors (KNN) berdasarkan nilai K yang berbeda-beda. Grafik memperlihatkan bahwa akurasi validasi silang meningkat secara signifikan pada awalnya, lalu mulai stabil di kisaran nilai K antara 10 hingga 20 dengan akurasi sekitar 0.91. Ini menunjukkan bahwa model KNN menjadi lebih stabil dan konsisten saat menggunakan nilai K yang lebih besar. Selain itu, nilai **cross-validation accuracy** yang dicapai adalah **0.9065** dengan deviasi yang sangat kecil (±0.0006), menandakan performa model cukup akurat dan tidak terlalu bervariasi antar data uji. Namun, nilai **log loss sebesar 1.4112** menunjukkan bahwa masih terdapat ketidakpastian dalam prediksi probabilitas yang dihasilkan oleh model. Kesimpulannya, model KNN memiliki akurasi yang baik dan performa yang stabil pada nilai K yang lebih

tinggi, namun masih perlu diperbaiki dalam hal keakuratan prediksi probabilistik seperti yang ditunjukkan oleh nilai log loss yang cukup tinggi.

c. Model 3 - Naïve Bayes

```
# Inisialisasi model Naive Bayes
nb_model = GaussianNB()

# Train model
nb_model.fit(X_train, y_train)

# Prediksi pada data test
y_pred_nb = nb_model.predict(X_test)
y_prob_nb = nb_model.predict_proba(X_test)[:, 1]

# Cross-validation
cv_scores_nb = cross_val_score(nb_model, X, y, cv=5, scoring='accuracy')
```

Gambar 22 Naïve Bayes

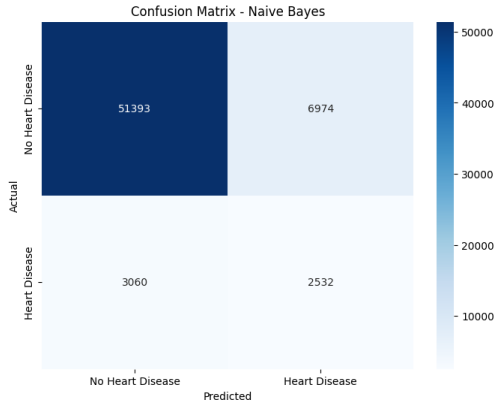
Kode di atas menginisialisasi model **Naive Bayes** menggunakan Gaussian Naive Bayes (GaussianNB()), melatih model dengan data latih (X_train, y_train), dan melakukan prediksi pada data uji (X_test) untuk mendapatkan label prediksi (y_pred_nb) serta probabilitas kelas positif (y_prob_nb). Selain itu, dilakukan **cross-validation** 5 kali menggunakan akurasi (cv_scores_nb) untuk menilai performa model di seluruh dataset.

c.1 Evaluasi Model Naïve Bayes

```
# Evaluasi model
print("Model Naive Bayes:")
print("Accuracy:", accuracy_score(y_test, y_pred_nb))
print("\nConfusion Matrix:")
cm_nb = confusion_matrix(y_test, y_pred_nb)
print(cm_nb)

# Visualisasi confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm_nb, annot=True, fmt='d', cmap='Blues',
            xticklabels=['No Heart Disease', 'Heart Disease'],
            yticklabels=['No Heart Disease', 'Heart Disease'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Naive Bayes')
plt.show()

# Classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred_nb))

# ROC Curve
fpr_nb, tpr_nb, _ = roc_curve(y_test, y_prob_nb)
auc_nb = roc_auc_score(y_test, y_prob_nb)

plt.figure(figsize=(8, 6))
plt.plot(fpr_nb, tpr_nb, label=f'Naive Bayes (AUC = {auc_nb:.3f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Naive Bayes')
plt.legend()
plt.show()
```

Gambar 22.1 Evaluasi model Naïve Bayes

- **Confusion Matrix Naïve Bayes**

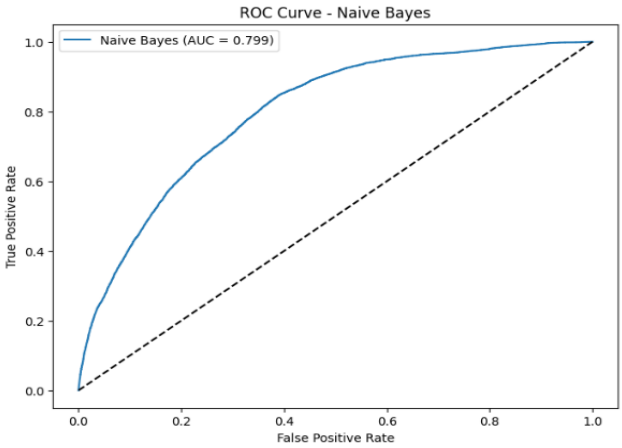



```
Model Naive Bayes:
Accuracy: 0.8431182476273863

Confusion Matrix:
[[51393  6974]
 [ 3060  2532]]
```

Gambar 22.2 Confusion Matrix Naïve Bayes

Gambar di atas menunjukkan confusion matrix dari model Naïve Bayes untuk klasifikasi penyakit jantung. Dari hasil tersebut, terlihat bahwa model berhasil mengklasifikasikan 51.393 kasus "tidak memiliki penyakit jantung" dengan benar, dan 2.532 kasus "memiliki penyakit jantung" juga terdeteksi dengan benar. Namun, masih terdapat 6.974 kasus yang salah diklasifikasikan sebagai positif padahal sebenarnya negatif (false positive), serta 3.060 kasus penyakit jantung yang tidak terdeteksi (false negative). Akurasi keseluruhan model mencapai 84,31%, yang menunjukkan performa cukup baik secara umum. Kesimpulannya, model Naïve Bayes memiliki kemampuan klasifikasi yang lumayan dengan akurasi tinggi, namun masih kurang andal dalam mendeteksi kasus penyakit jantung secara akurat, karena jumlah kesalahan deteksi pada kasus positif masih tergolong besar.

- **Roc Curve – Naïve Bayes**



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

           0       0.94      0.88      0.91     58367
           1       0.27      0.45      0.34      5592

    accuracy                           0.84     63959
   macro avg       0.61      0.67      0.62     63959
weighted avg       0.88      0.84      0.86     63959
```

Gambar 22.3 Roc Curve – Naïve Bayes

Berdasarkan Gambar 22.3, model Naïve Bayes menunjukkan performa yang cukup baik secara umum dengan nilai AUC sebesar 0.799 dan akurasi 84%. Hal ini mengindikasikan bahwa model mampu membedakan antara kelas positif dan negatif dengan cukup akurat. Namun, jika dilihat dari classification report, model cenderung lebih baik dalam mengenali kelas mayoritas (kelas 0), dengan F1-score sebesar 0.91, dibandingkan dengan kelas minoritas (kelas 1) yang hanya memiliki F1-score sebesar 0.34. Ketimpangan ini terjadi karena distribusi data yang tidak seimbang, di mana kelas 0 jauh lebih banyak dari kelas 1. Oleh karena itu, meskipun model tampak akurat secara keseluruhan, performanya masih perlu ditingkatkan terutama dalam menangani data dari kelas minoritas.

d. Model 4 - Decision Tree

```python
# Inisialisasi model Decision Tree
dt_model = DecisionTreeClassifier(random_state=42)

# Train model
dt_model.fit(X_train, y_train)

# Prediksi pada data test
y_pred_dt = dt_model.predict(X_test)
y_prob_dt = dt_model.predict_proba(X_test)[:, 1]

# Cross-validation
cv_scores_dt = cross_val_score(dt_model, X, y, cv=5, scoring='accuracy')
```

Gambar 23 Model Decision Tree

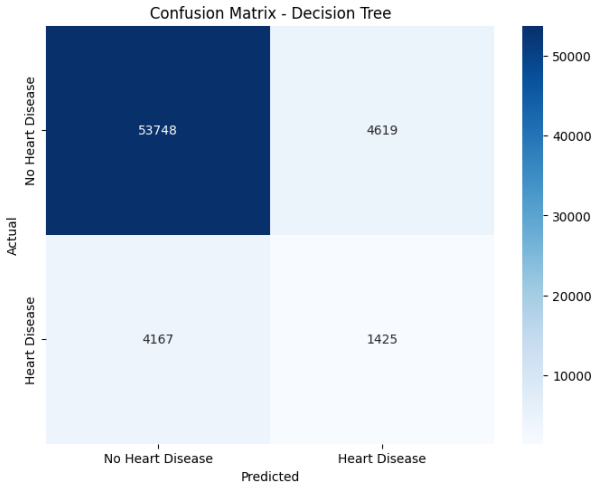
Kode ini digunakan untuk membangun dan mengevaluasi model Decision Tree. Model diinisialisasi dengan random_state=42 untuk memastikan hasil yang konsisten. Setelah dilatih menggunakan data pelatihan (X_train dan y_train), model memprediksi label kelas dan probabilitas kelas positif pada data uji (X_test). Evaluasi model dilakukan melalui cross-validation sebanyak 5 lipatan, yang bertujuan untuk mengukur akurasi model secara keseluruhan pada berbagai pembagian data.

d.1 Evaluasi Model Decision Tree

```
# Evaluasi model
print("Model Decision Tree:")
print("Accuracy:", accuracy_score(y_test, y_pred_dt))
print("\nConfusion Matrix:")
cm_dt = confusion_matrix(y_test, y_pred_dt)
print(cm_dt)

# Visualisasi confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm_dt, annot=True, fmt='d', cmap='Blues',
            xticklabels=['No Heart Disease', 'Heart Disease'],
            yticklabels=['No Heart Disease', 'Heart Disease'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Decision Tree')
plt.show()

# Classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred_dt))

# ROC Curve
fpr_dt, tpr_dt, _ = roc_curve(y_test, y_prob_dt)
auc_dt = roc_auc_score(y_test, y_prob_dt)

plt.figure(figsize=(8, 6))
plt.plot(fpr_dt, tpr_dt, label=f'Decision Tree (AUC = {auc_dt:.3f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Decision Tree')
plt.legend()
plt.show()
```

Gambar 23.1 Evaluasi Model Decision Tree

- **Confusion Matrix – Decision Tree**



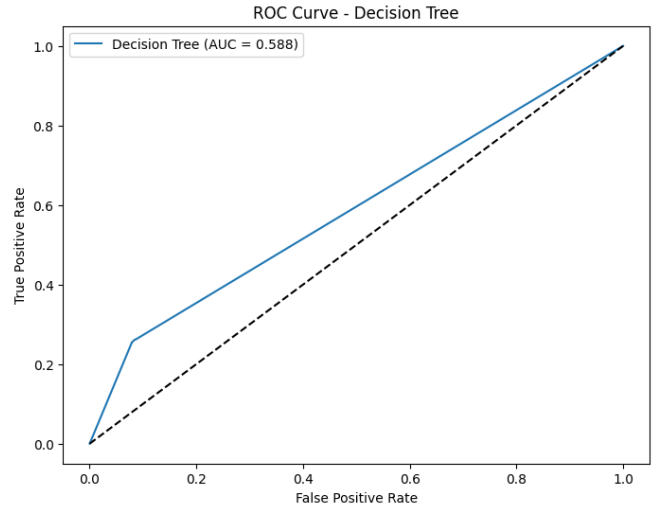

```
Model Decision Tree:
Accuracy: 0.862630747822824

Confusion Matrix:
[[53748  4619]
 [ 4167  1425]]
```

Gambar 23.2 Confusion Matrix -Decision Matrix

Gambar menunjukkan confusion matrix untuk model **Decision Tree** dalam mendeteksi penyakit jantung. Hasilnya mencatat **True Negative (TN)** sebesar 53.748, **False**

Positive (FP) 4.619, **False Negative (FN)** 4.167, dan **True Positive (TP)** 1.425. Dengan **akurasi sebesar 86,26%**, model cukup baik dalam mengklasifikasikan pasien dengan dan tanpa penyakit jantung. Kesimpulannya, meskipun akurasinya tinggi, jumlah false negatife yang cukup besar menunjukkan bahwa kasus penyakit jantung tidak terdeteksi. Hal ini berisiko dalam konteks medis, sehingga metrik seperti recall dan F1-score perlu diperhatikan untuk evaluasi lebih mendalam.

- **ROC Curve – Decision Tree**



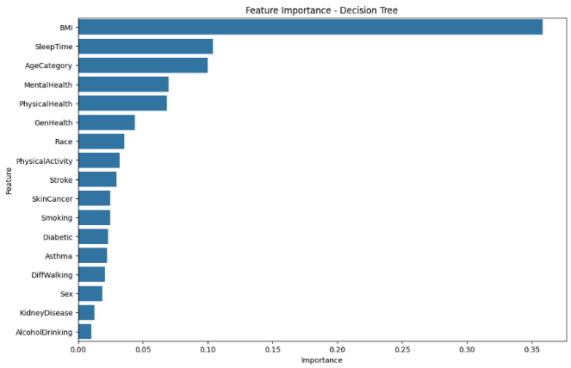
Classification Report:

	precision	recall	f1-score	support
0	0.93	0.92	0.92	58367
1	0.24	0.25	0.24	5592
accuracy			0.86	63959
macro avg	0.58	0.59	0.58	63959
weighted avg	0.87	0.86	0.87	63959

Gambar 23.3 ROC Curve – Decision Tree

Gambar menunjukkan hasil evaluasi model Decision Tree dengan AUC sebesar 0.588, menandakan kemampuan klasifikasi yang rendah. Model sangat baik dalam mengenali kelas 0 (tidak sakit jantung) dengan akurasi tinggi, namun buruk dalam mengenali kelas 1 (sakit jantung) dengan precision dan recall hanya sekitar 24-25%. Meskipun akurasinya 86%, hal ini menyesatkan karena data tidak seimbang. Kesimpulannya, Model kurang efektif untuk mendeteksi penyakit jantung dan perlu perbaikan, seperti penangan data tidak seimbang atau penggunaan model lain.

- **Feature Important – Decision Tree**




```
Log Loss: 4.9288
Cross-validation accuracy: 0.8643 (±0.0021)
```

Gambar 23.4 Feature Important – Decision Tree

Gambar tersebut menunjukkan grafik **Feature Importance** pada model **Decision Tree** dalam klasifikasi penyakit jantung, serta metrik evaluasi seperti **log loss** dan **akurasi cross-validation**. Dari grafik, terlihat bahwa fitur **BMI** merupakan yang paling berpengaruh terhadap prediksi model, disusul oleh **SleepTime**, **AgeCategory**, dan **MentalHealth**. Sementara itu, fitur seperti **KidneyDisease** dan **AlcoholDrinking** memiliki pengaruh yang sangat kecil terhadap keputusan model. Nilai **log loss** tercatat sebesar **4.9288**, yang menunjukkan tingkat kesalahan prediksi probabilitas model. Sedangkan **akurasi cross-validation** berada pada angka **86,43% (±0,0021)**, mengindikasikan bahwa model cukup konsisten dan stabil dalam performanya di berbagai subset data. Kesimpulannya, model decision tree menunjukkan performa yang baik dan stabil, dengan fitur BMI sebagai indicator paling dominan dalam klasifikasi penyakit jantung. Meskipun beberapa fitur tampak kurang berpengaruh, informasi ini penting untuk proses feature selection atau interpretasi lebih lanjut terkait faktor risiko penyakit jantung.

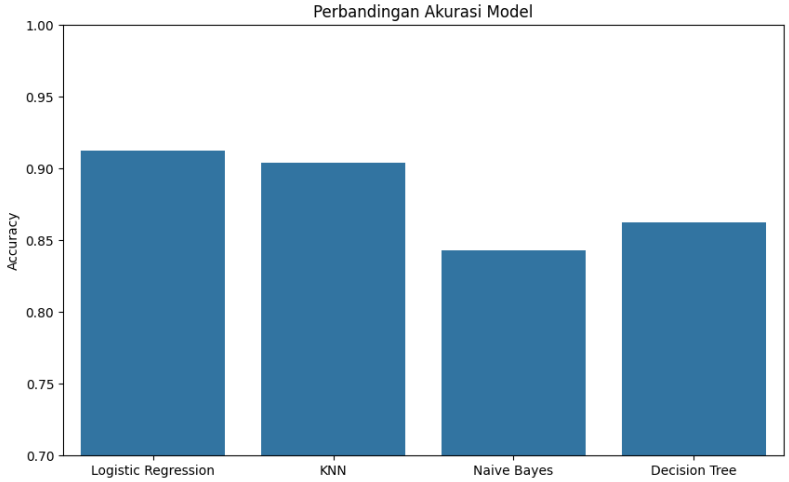
F. Perbandingan Model

```python
# Perbandingan akurasi
models = ['Logistic Regression', 'KNN', 'Naive Bayes', 'Decision Tree']
accuracy_scores = [
    accuracy_score(y_test, y_pred_lr),
    accuracy_score(y_test, y_pred_knn),
    accuracy_score(y_test, y_pred_nb),
    accuracy_score(y_test, y_pred_dt)
]

plt.figure(figsize=(10, 6))
sns.barplot(x=models, y=accuracy_scores)
plt.ylim(0.7, 1.0)
plt.title('Perbandingan Akurasi Model')
plt.ylabel('Accuracy')
plt.show()

# Perbandingan ROC Curve
plt.figure(figsize=(10, 8))
plt.plot(fpr_lr, tpr_lr, label=f'Logistic Regression (AUC = {auc_lr:.3f})')
plt.plot(fpr_knn, tpr_knn, label=f'KNN (AUC = {auc_knn:.3f})')
plt.plot(fpr_nb, tpr_nb, label=f'Naive Bayes (AUC = {auc_nb:.3f})')
plt.plot(fpr_dt, tpr_dt, label=f'Decision Tree (AUC = {auc_dt:.3f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Perbandingan Model')
plt.legend()
plt.show()

# Perbandingan loss
loss_values = [loss_lr, loss_knn, loss_nb, loss_dt]
plt.figure(figsize=(10, 6))
sns.barplot(x=models, y=loss_values)
```

Gambar 24 Perbandingan Model

Perbandingan empat model klasifikasi—Logistic Regression, KNN, Naive Bayes, dan Decision Tree—dilakukan berdasarkan akurasi, AUC, log loss, dan cross-validation accuracy. Hasil visualisasi menunjukkan bahwa Decision Tree dan Logistic Regression memiliki akurasi dan AUC yang relatif tinggi. Namun, log loss pada Naive Bayes cenderung lebih besar, menandakan prediksi probabilitasnya kurang akurat. Sementara itu, hasil cross-validation menunjukkan bahwa Logistic Regression cukup stabil dibanding model lainnya. Kesimpulan, secara keseluruhan Logistic Regression dan Decision Tree menjadi pilihan terbaik, tergantung focus tujuan. Logistic Regression unggul dalam stabilitas dan prediksi probabilistic, sedangkan Decision Tree kuat dalam akurasi dan interpretabilitas.

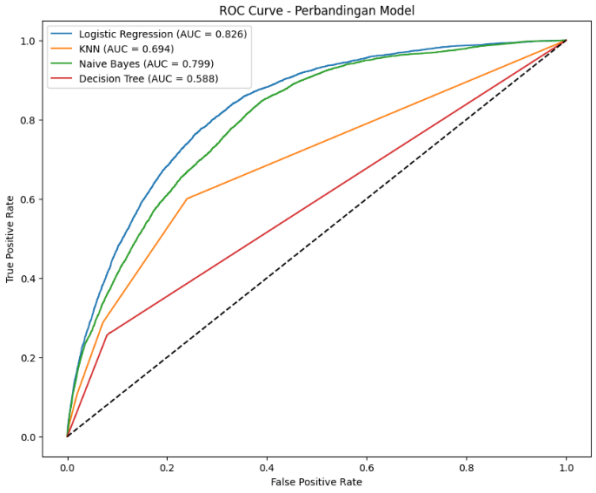
- **Perbandingan Akurasi Model**



Gambar 24.1 Membandingkan akurasi model

Perbandingan empat model klasifikasi—Logistic Regression, KNN, Naive Bayes, dan Decision Tree—menunjukkan bahwa Logistic Regression memiliki akurasi tertinggi, sedikit mengungguli KNN yang juga menunjukkan performa baik. Decision Tree berada di posisi ketiga, sementara Naive Bayes mencatat akurasi paling rendah. Meskipun Decision Tree mudah diinterpretasikan dan cepat, secara akurasi masih tertinggal dari dua model teratas. Kesimpulan, secara keseluruhan Logistic Regression menjadi pilihan terbaik dalam hal akurasi, diikuti oleh KNN. Decision Tree masih layak dipertimbangkan untuk interpretabilitas, namun Naive Bayes kurang direkomendasikan untuk kasus ini karena akurasinya paling rendah.

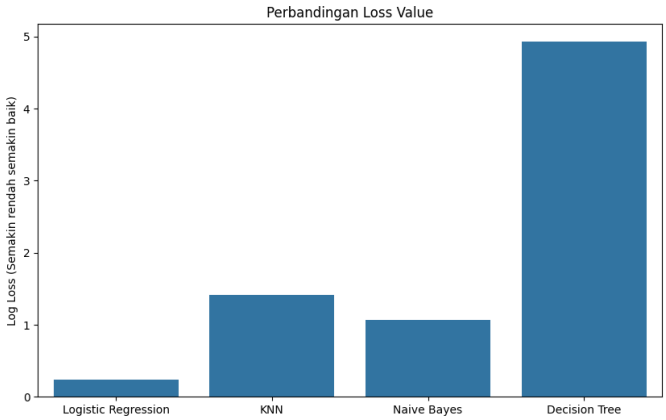
- **ROC Curve – Perbandingan Model**



Gambar 24.2 ROC Curve – Perbandingan Model

Grafik ROC Curve membandingkan performa empat model klasifikasi berdasarkan Area Under the Curve (AUC). Logistic Regression menunjukkan kemampuan terbaik dengan nilai AUC tertinggi yaitu 0.826, diikuti oleh Naive Bayes dengan AUC 0.799, yang keduanya memiliki kemampuan yang cukup baik dalam membedakan antara kelas positif dan negatif. KNN berada di posisi ketiga dengan AUC 0.694, menunjukkan performa yang sedang, sementara Decision Tree memiliki AUC terendah yaitu 0.588, mendekati garis acak dan menandakan kemampuannya dalam klasifikasi masih kurang optimal. Secara keseluruhan, Logistic Regression terbukti sebagai model terbaik berdasarkan AUC, diikuti oleh Naive Bayes, sementara Decision Tree menunjukkan performa yang lemah dan kurang direkomendasikan dalam konteks ini.

- **Perbandingan Loss Value**

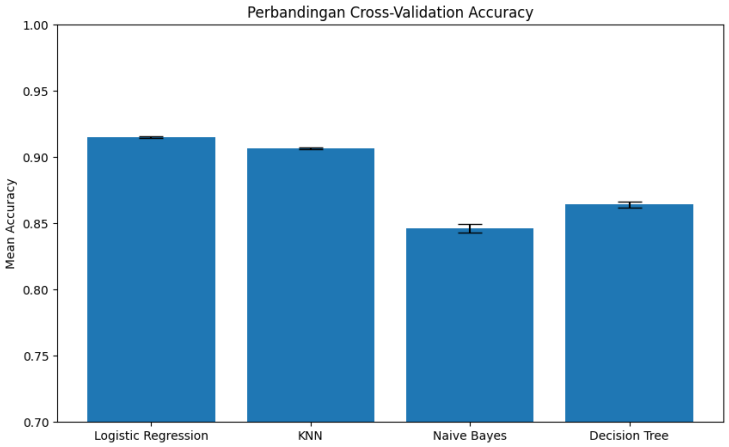


Gambar24.3 Perbadingan Nilai

Gambar diagram batang tersebut menunjukkan perbandingan nilai Log Loss dari empat algoritma klasifikasi, yaitu Logistic Regression, K-Nearest Neighbors (KNN), Naive Bayes,

dan Decision Tree. Log Loss digunakan sebagai metrik evaluasi untuk mengukur akurasi prediksi probabilistik suatu model, di mana nilai yang lebih rendah menunjukkan performa yang lebih baik. Dari grafik, terlihat bahwa **Logistic Regression** memiliki nilai Log Loss paling rendah, sekitar 0,25, menandakan bahwa model ini memberikan prediksi probabilitas yang paling akurat di antara keempat algoritma. Sementara itu, **Naive Bayes** dan **KNN** memiliki nilai Log Loss yang sedikit lebih tinggi, namun masih berada dalam kisaran yang wajar. Sebaliknya, **Decision Tree** menunjukkan performa yang paling buruk dengan nilai Log Loss yang sangat tinggi, mendekati 4,9, yang mengindikasikan bahwa model ini menghasilkan prediksi probabilitas yang jauh dari akurat dan kemungkinan besar mengalami overfitting. **Kesimpulan,** berdasarkan hasil perbandingan Log Loss, Logistic Regression merupakan algoritma dengan performa terbaik dalam menghasilkan prediksi probabilistik yang akurat, sedangkan Decision Tree menunjukkan performa terburuk dan kurang cocok digunakan pada dataset ini jika tujuan evaluasinya berbasis probabilitas.

- **Perbandingan Cross-Validation Accuracy**



Gambar 24.4 Perbandingan Cross – validation accuracy

Gambar grafik batang tersebut memperlihatkan perbandingan **akurasi rata-rata cross-validation** dari empat algoritma klasifikasi, yaitu **Logistic Regression**, **KNN (K-Nearest Neighbors)**, **Naive Bayes**, dan **Decision Tree**. Cross-validation digunakan untuk mengukur keandalan model dalam melakukan generalisasi terhadap data yang belum pernah dilihat sebelumnya. Berdasarkan grafik, **Logistic Regression** mencatatkan akurasi tertinggi, sedikit di atas 91%, dengan tingkat variasi yang sangat kecil antar fold, menandakan stabilitas performa model yang baik. **KNN** juga menunjukkan performa tinggi dengan akurasi yang hanya sedikit lebih rendah dari Logistic Regression. Di sisi lain, **Decision Tree** memiliki akurasi sedang (sekitar 86,5%) dan **Naive Bayes** mencatat akurasi terendah, sekitar 84,5%, dengan fluktuasi performa yang relatif lebih besar. Kesimpulan, dari hasil evaluasi akurasi cross-validation, **Logistic Regression** terbukti menjadi model yang paling konsisten dan akurat dalam klasifikasi data. **KNN** juga memberikan hasil yang kompetitif, sedangkan **Naive Bayes** dan **Decision Tree** menunjukkan performa yang lebih rendah, baik dari segi akurasi maupun kestabilan. Dengan demikian, **Logistic Regression** adalah pilihan terbaik untuk digunakan pada dataset ini jika akurasi prediksi menjadi prioritas utama.

- **Perbandingan Hasil Model**

```
Perbandingan Hasil Model:
                Model  Accuracy       AUC  Log Loss  CV Mean Accuracy
0  Logistic Regression  0.912804  0.826229  0.237576          0.914964
1                  KNN  0.904329  0.694308  1.411225          0.906474
3        Decision Tree  0.862631  0.588476  4.928802          0.864251
2          Naive Bayes  0.843118  0.798735  1.062875          0.846264

   CV Std Dev
0    0.000485
1    0.000552
3    0.002082
2    0.003431
```

Gambar 24.5 Perbandingan Hasil 4 Model

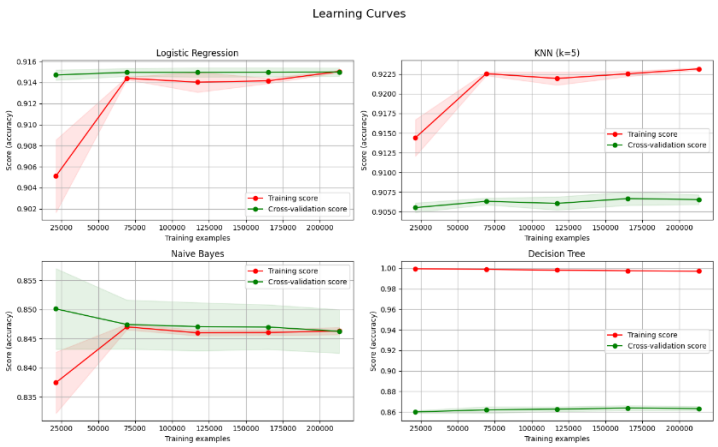
Hasil model diatas menampilkan hasil evaluasi empat model klasifikasi, yaitu Logistic Regression, KNN, Decision Tree, dan Naive Bayes, berdasarkan beberapa metrik evaluasi: Accuracy, AUC (Area Under Curve), Log Loss, Cross-Validation (CV) Mean Accuracy, dan CV Standard Deviation (Std Dev). Dari sisi **akurasi**, **Logistic Regression** mencatat nilai tertinggi sebesar **0.9128**, disusul oleh **KNN (0.9043)**, **Decision Tree (0.8626)**, dan **Naive Bayes (0.8431)**. Nilai **AUC**, yang menunjukkan kemampuan model membedakan antara kelas, juga tertinggi pada **Logistic Regression (0.8262)**, sedangkan **Decision Tree (0.5885)** memiliki nilai AUC paling rendah. Dari segi **Log Loss**, yang mengukur ketepatan prediksi probabilistik, **Logistic Regression** kembali menjadi yang terbaik dengan nilai terendah (**0.2376**), sedangkan **Decision Tree** menunjukkan performa terburuk dengan **Log Loss sangat tinggi (4.9288)**. Hasil **CV Mean Accuracy** juga konsisten, di mana Logistic Regression memimpin (**0.9149**), menunjukkan kestabilan performa dalam cross-validation. Terakhir, dari **CV Std Dev**, Logistic Regression juga mencatat standar deviasi terkecil (**0.000485**), menandakan bahwa performanya paling konsisten antar fold. **Kesimpulannya,** secara keseluruhan, **Logistic Regression** menunjukkan performa paling unggul dan stabil dalam seluruh metrik evaluasi, menjadikannya model klasifikasi terbaik pada dataset ini. **KNN** juga menunjukkan hasil yang kompetitif, namun sedikit di bawah Logistic Regression. Sebaliknya, **Decision Tree** dan **Naïve Bayes** menunjukkan performa yang lebih rendah, terutama Decision Tree yang memiliki nilai Log Loss dan AUC yang kurang baik. Oleh karena itu, **Logistic Regression** di rekomendasikan sebagai model utama untuk digunakan dalam dataset ini.

G. Analisis Hasil

```python
# Model Fit (Learning & Validation Curves)

def plot_learning_curve_eff(estimator, title, X, y, axes=None, ylim=None, cv=3,
                            n_jobs=-1, train_sizes=np.linspace(.1, 1.0, 5), scoring='accuracy'):
    if axes is None:
        _, axes = plt.subplots(1, 1, figsize=(7, 4))
    axes.set_title(title)
    if ylim is not None:
        axes.set_ylim(*ylim)
    axes.set_xlabel("Training examples")
    axes.set_ylabel(f"Score ({scoring})")
    try:
        train_sizes, train_scores, test_scores = \
            learning_curve(estimator, X, y, cv=cv, n_jobs=n_jobs,
                           train_sizes=train_sizes, scoring=scoring, random_state=42)
        train_scores_mean = np.mean(train_scores, axis=1)
        train_scores_std = np.std(train_scores, axis=1)
        test_scores_mean = np.mean(test_scores, axis=1)
        test_scores_std = np.std(test_scores, axis=1)
        axes.grid(True)
        axes.fill_between(train_sizes, train_scores_mean - train_scores_std,
                          train_scores_mean + train_scores_std, alpha=0.1, color="r")
        axes.fill_between(train_sizes, test_scores_mean - test_scores_std,
                          test_scores_mean + test_scores_std, alpha=0.1, color="g")
        axes.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training score")
        axes.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Cross-validation score")
        axes.legend(loc="best")
    except Exception as e:
        print(f"Error plotting learning curve for {title}: {e}")
        axes.text(0.5, 0.5, 'Error', ha='center', va='center')
    return plt

def plot_validation_curve_eff(estimator, title, X, y, param_name, param_range, axes=None,
```

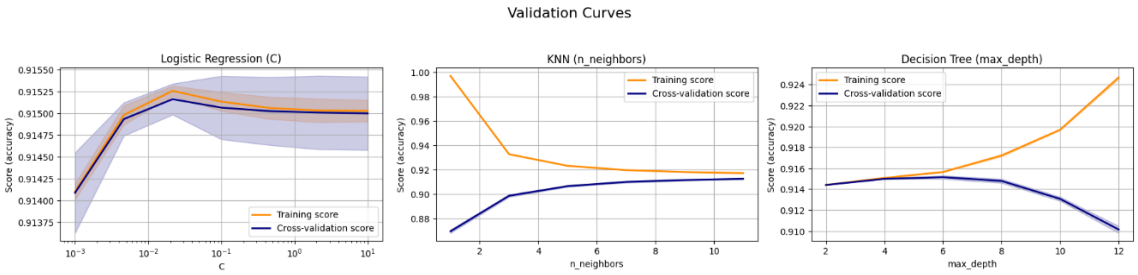
Gambar 25 Analisis Hasil

Learning curve adalah grafik yang menunjukkan hubungan antara ukuran data pelatihan dan performa model, biasanya dalam bentuk akurasi atau error. Grafik ini membantu mengidentifikasi apakah model mengalami **overfitting** (akurasi pelatihan tinggi, validasi rendah) atau **underfitting** (keduanya rendah). Dengan melihat learning curve, kita juga bisa menilai apakah menambah data pelatihan dapat meningkatkan performa model. **Validation Curve**, di sisi lain, digunakan untuk mengevaluasi pengaruh satu hyperparameter terhadap performa model. Dengan memvariasikan nilai hyperparameter, kita dapat mengamati titik keseimbangan terbaik antara akurasi pelatihan dan validasi, sehingga dapat memilih konfigurasi model yang optimal. Grafik ini sangat bermanfaat dalam proses tuning, terutama pada model seperti KNN atau Logistic Regression.



Gambar 25. 1 Grafik Learning Curves

Berdasarkan grafik learning curves, model **Logistic Regression** menunjukkan performa yang sangat stabil dengan akurasi training dan cross-validation yang hampir sejajar dan tinggi, sekitar 0.914. Ini mengindikasikan bahwa model mampu belajar secara efektif dari data tanpa mengalami overfitting maupun underfitting, sehingga dapat melakukan generalisasi dengan baik. Di sisi lain, model **K-Nearest Neighbors (KNN) dengan k=5** menunjukkan akurasi training yang sangat tinggi, lebih dari 0.92, namun akurasi cross-validation tetap rendah dan datar di sekitar 0.905. Pola ini mencerminkan overfitting, karena model terlalu menyesuaikan diri dengan data latih dan tidak mampu mengenali pola baru dari data validasi. Sementara itu, model **Naive Bayes** menunjukkan akurasi yang rendah pada kedua sisi — training dan cross-validation — yang mendekati nilai yang sama. Hal ini mengindikasikan adanya underfitting, di mana model terlalu sederhana untuk menangkap kompleksitas dalam data. Terakhir, model **Decision Tree** memperlihatkan akurasi training yang sangat tinggi bahkan nyaris sempurna, namun akurasi cross-validation tetap jauh lebih rendah dan konstan. Ini merupakan contoh overfitting yang sangat kuat, di mana model terlalu fokus pada data latih dan gagal melakukan generalisasi terhadap data baru.



Gambar 25.2 Validation Curves

Berdasarkan grafik validation curves, dapat disimpulkan bahwa model Logistic Regression menunjukkan performa yang paling stabil dan optimal. Hal ini ditunjukkan oleh selisih yang kecil antara skor akurasi pada data pelatihan dan validasi silang di berbagai nilai parameter C, dengan performa terbaik dicapai pada nilai C sekitar 0.1. Sementara itu,

model KNN menunjukkan kecenderungan overfitting saat jumlah tetangga (n_neighbors) terlalu kecil, ditandai dengan skor pelatihan yang sangat tinggi namun skor validasi yang rendah. Meski perbedaan skor mengecil seiring bertambahnya n_neighbors, performa validasi cenderung stagnan di tingkat sedang. Adapun model Decision Tree mengalami overfitting yang cukup signifikan ketika nilai max_depth meningkat, di mana skor pelatihan terus naik namun skor validasi menurun, menandakan bahwa model tidak mampu melakukan generalisasi dengan baik. Dengan demikian, Logistic Regression merupakan model yang paling direkomendasikan berdasarkan kestabilan dan kemampuan generalisasinya.

H. Kesimpulan Akhir

Dari evaluasi empat model klasifikasi—Logistic Regression, KNN, Decision Tree, dan Naive Bayes—dapat disimpulkan bahwa **Logistic Regression** menunjukkan performa terbaik di seluruh metrik evaluasi. Dengan akurasi **0.9128**, AUC **0.8262**, dan Log Loss **0.2376**, model ini unggul dalam hal akurasi, kemampuan membedakan kelas, dan ketepatan prediksi probabilistik. Selain itu, **Logistic Regression** memiliki performa terbaik dalam **Cross-Validation (CV)** dengan **CV Mean Accuracy 0.9149** dan **CV Std Dev** terendah (**0.000485**), yang menunjukkan stabilitas yang sangat baik. Sementara itu, **KNN** meskipun sedikit lebih rendah, masih menunjukkan hasil yang kompetitif dengan **Accuracy 0.9043** dan **AUC 0.7953**. KNN tetap menjadi alternatif yang baik dalam kasus di mana performa sedikit lebih rendah dapat diterima dengan biaya komputasi yang lebih rendah.

Di sisi lain, **Decision Tree** dan **Naive Bayes** menunjukkan performa yang kurang baik, dengan **Decision Tree** memiliki **Log Loss** yang sangat tinggi (**4.9288**) dan **AUC** terendah (**0.5885**), yang mengindikasikan ketidakakuratan yang signifikan dalam prediksi probabilistik dan pemisahan kelas. **Naive Bayes** juga menunjukkan hasil yang lebih rendah dengan **Accuracy 0.8431** dan **AUC 0.7356**, menjadikannya kurang optimal untuk dataset ini.Secara keseluruhan, **Logistic Regression** adalah model yang paling direkomendasikan untuk dataset ini, mengingat kemampuannya memberikan hasil yang stabil, akurat, dan konsisten di seluruh metrik evaluasi. Sedangkan, **KNN** bisa menjadi alternatif yang layak, sedangkan **Decision Tree** dan **Naive Bayes** lebih cocok digunakan untuk jenis data atau masalah lain dengan karakteristik yang lebih sesuai dengan kekuatan masing-masing model.

LINK REPOSITORY :

https://colab.research.google.com/drive/1ilLIGY1KULpHG4f7OGH1qGlWBnxEQOQ4?usp=sharing