TUGAS MANDIRI

FUNDAMENTAL OF DATA MINING

PREDIKSI KELULUSAN MAHASISWA MENGGUNAKAN ALGORITMA C4.5 DAN NAIVE BAYES

Nama : Melvin Gwee

NPM : 231510015

Dosen : Erlin Elisa, S.Kom., M.Kom.

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS TEKNIK DAN KOMPUTER

UNIVERSITAS PUTERA BATAM

2026

KATA PENGANTAR

Penulis bersyukur atas rahmat dan karunia Allah yang memungkinkan Penulis menyelesaikan Laporan Klasifikasi Status Pinjaman Menggunakan Algoritma C4.5 (Decision Tree), mata kuliah Fundamentals of Data Mining dengan tepat waktu.

Penulis menyadari bahwa Laporan ini masih jauh dari kata sempurna. Oleh karena itu, Penulis dengan senang hati menerima kritik dan saran yang dapat membantu laporan ini menjadi lebih baik. Selain itu, Penulis menyadari bahwa laporan ini tidak akan terwujud tanpa dukungan, petunjuk, dan dorongan dari berbagai pihak. Penulis dengan rendah hati mengucapkan terima kasih kepada :

1. Ibu Erlin Elisa, S.Kom., M.Kom. yang telah mengajarkan dan membantu Penulis dalam proses penyusunan laporan ini.
2. Orang tua, yang telah memberikan semangat dan motivasi kepada Penulis untuk terus berjuang dalam menggapai segala cita – cita dan impian Penulis.
3. Teman – teman, yang telah memberikan bantuan moral dan material untuk mewujudkan laporan ini.

Semoga Tuhan Yang Maha Esa membalas perbuatan baik ini dan terus memberikan rahmat dan karunia-Nya.

Batam, 7 Januari 2026

Melvin Gwee

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Latar Belakang Masalah Fenomena ketidakpastian kelulusan mahasiswa merupakan tantangan bagi institusi pendidikan untuk menjaga kualitas dan rasio kelulusan. Berdasarkan dataset yang mencakup aspek kehadiran dan aktivitas e-learning, terlihat adanya pola perilaku yang memengaruhi performa akademik. Menurut Setiyani dkk. (2020), analisis prediksi kelulusan menggunakan teknik data mining menjadi krusial untuk memberikan gambaran strategis bagi pengambil kebijakan di perguruan tinggi. Data mining diperlukan karena mampu mengekstraksi pengetahuan dari data historis yang besar untuk memprediksi kelulusan mahasiswa secara lebih akurat dibandingkan metode konvensional (Ashari Muin, 2016).

Metodologi Penelitian ini menerapkan dua algoritma klasifikasi populer dalam dunia pendidikan. Pertama, algoritma Naive Bayes yang digunakan untuk menghitung probabilitas kelulusan berdasarkan variabel-variabel aktivitas mahasiswa, di mana metode ini terbukti efektif dalam studi kasus data mahasiswa baru (Ashari Muin, 2016). Kedua, algoritma C4.5 yang diimplementasikan untuk membentuk pohon keputusan (decision tree) guna memetakan faktor-faktor dominan yang memengaruhi status akademik mahasiswa, sebagaimana diterapkan pada aplikasi prediksi kelulusan prodi informatika (Puspita dkk., 2018).

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, masalah yang dapat diidentifikasi adalah:

- Bagaimana proses preprocessing data dalam menangani variabel kategorikal dan normalisasi pada dataset aktivitas mahasiswa?
- Bagaimana implementasi algoritma C4.5 (Decision Tree) dan Naive Bayes dalam mengklasifikasi status akademik mahasiswa?

- Bagaimana performa model dalam memprediksi status "Lulus" atau "Tidak" berdasarkan metrik akurasi, precision, dan recall?

1.3 Tujuang Penelitian

- Melakukan eksplorasi data dan transformasi fitur agar siap digunakan dalam model klasifikasi.
- Membangun model klasifikasi menggunakan algoritma C4.5 dan Naive Bayes.
- Membandingkan performa kedua algoritma untuk menentukan metode yang paling akurat dalam mendeteksi risiko akademik mahasiswa.

BAB II

PEMBAHASAN

2.1 Dataset

- Sumber Dataset: Dataset Mandiri (Dataset_Mahasiswa_Kehadiran_Aktivitas_IPK.csv).
- Jumlah Record & Atribut: 1.500 record dengan 10 atribut.
- Karakteristik Data:
  - Variabel Input: Jenis Kelamin, Umur, Status Menikah, Kehadiran (%), Partisipasi Diskusi, Nilai Tugas, Aktivitas E-Learning, dan IPK.
  - Label (Target): Status Akademik (Lulus, Tidak).

2.2 Metodologi

2.2.1 Data Understanding

Mengidentifikasi variabel input (IPK, Kehadiran, Aktivitas) dan variabel target (Status Akademik).

2.2.2 Data Preprocessing

- Encoding: Mengubah 'Jenis Kelamin' dan 'Status Menikah' menjadi numerik.
- Handling Missing Value: Memastikan tidak ada data yang kosong.

2.2.3 Pemodelan

- C4.5 (Decision Tree): Algoritma yang membentuk pohon keputusan berdasarkan nilai Information Gain.
- Naive Bayes: Algoritma klasifikasi berbasis probabilitas (Teorema Bayes).

2.2.4 Evaluasi Model

Mengukur performa menggunakan Accuracy, Precision, Recall, dan F1-Score.

2.3 Implementasi Python

2.3.1 Import Library

Pertama, saya akan mengimpor pandas dan numpy untuk manipulasi data dan operasi numerik. Kemudian, saya akan memuat dataset dari file CSV yang tersedia ke dalam DataFrame pandas bernama 'df' dan menampilkan beberapa baris pertamanya untuk memverifikasi bahwa data berhasil dimuat serta memeriksa strukturnya.

```
import pandas as pd
import numpy as np

# Load the dataset
df =
pd.read_csv('/content/Dataset_Mahasiswa_Kehadiran_A
ktivitas_IPK.csv')

# Display the first few rows of the DataFrame
df.head()
```

2.3.2 Load Dataset

	Nama	Jenis Kelamin	Umur	Status Menikah	Kehadiran (%)	Partisipasi Diskusi (skor)	Nilai Tugas (rata-rata)	Aktivitas E-Learning (skor)	IPK	Status Akademik
0	Anastasia Pradipta	Perempuan	24	Menikah	61	47	60	98	2.27	Lulus
1	Salman Mahendra	Perempuan	29	Menikah	88	52	95	90	2.35	Lulus
2	Mahfud Narpati, S.Gz	Laki-laki	26	Menikah	46	71	77	97	3.30	Lulus
3	Bahuwirya Sihotang	Laki-laki	21	Belum Menikah	40	67	45	91	3.84	Lulus
4	Puti Raisa Mardhiyah	Laki-laki	30	Belum Menikah	45	57	82	94	2.30	Lulus

2.3.3 Exploratory Data Analyssi (EDA)

Untuk memulai Analisis Data Eksploratif (EDA), saya akan memeriksa informasi dasar dari DataFrame, termasuk tipe data, nilai yang bukan null, dan penggunaan memori menggunakan .info(), kemudian menampilkan statistik deskriptif untuk kolom numerik menggunakan .describe(). Setelah itu, saya akan memeriksa apakah ada nilai yang hilang di seluruh kolom menggunakan .isnull().sum() untuk mendapatkan gambaran singkat mengenai kelengkapan data.

```
print("DataFrame Info:")
df.info()

print("\nDescriptive Statistics:")
df.describe()

print("\nMissing Values:")
df.isnull().sum()
```

```
DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Nama                       1500 non-null   object
 1   Jenis Kelamin              1500 non-null   object
 2   Umur                       1500 non-null   int64
 3   Status Menikah             1500 non-null   object
 4   Kehadiran (%)              1500 non-null   int64
 5   Partisipasi Diskusi (skor) 1500 non-null   int64
 6   Nilai Tugas (rata-rata)    1500 non-null   int64
 7   Aktivitas E-Learning (skor) 1500 non-null  int64
 8   IPK                        1500 non-null   float64
 9   Status Akademik            1500 non-null   object
dtypes: float64(1), int64(5), object(4)
memory usage: 117.3+ KB

Descriptive Statistics:

Missing Values:
```

	0
Nama	0
Jenis Kelamin	0
Umur	0
Status Menikah	0
Kehadiran (%)	0
Partisipasi Diskusi (skor)	0
Nilai Tugas (rata-rata)	0
Aktivitas E-Learning (skor)	0
IPK	0
Status Akademik	0

```
dtype: int64
```

Berdasarkan keluaran dari df.info() sebelumnya, saya akan mengidentifikasi kolom kategorikal dan numerik. Kemudian, saya akan melakukan iterasi melalui kolom kategorikal yang telah diidentifikasi (kecuali 'Nama' karena merupakan pengidentifikasi) untuk menampilkan jumlah nilai unik dan distribusinya, dan terakhir, memeriksa distribusi dari variabel target 'Status Akademik'.

```
categorical_cols =
df.select_dtypes(include='object').columns.tolist()
numerical_cols = df.select_dtypes(include=['int64',
'float64']).columns.tolist()

print("\nCategorical Columns:", categorical_cols)
print("Numerical Columns:", numerical_cols)

print("\nDistribution of Categorical Columns:")
for col in categorical_cols:
    if col != 'Nama': # 'Nama' column is likely
unique and not useful for value_counts()
        print(f"\n--- {col} ---")
        print(df[col].value_counts())

print("\n--- Distribution of Target Variable
'Status Akademik' ---")
print(df['Status Akademik'].value_counts())
```

```
Categorical Columns: ['Nama', 'Jenis Kelamin', 'Status Menikah', 'Status Akademik']
Numerical Columns: ['Umur', 'Kehadiran (%)', 'Partisipasi Diskusi (skor)', 'Nilai Tugas (rata-rata)', 'Aktivitas E-Learning (skor)', 'IPK']

Distribution of Categorical Columns:

--- Jenis Kelamin ---
Jenis Kelamin
Laki-laki    757
Perempuan    743
Name: count, dtype: int64

--- Status Menikah ---
Status Menikah
Belum Menikah    755
Menikah          745
Name: count, dtype: int64

--- Status Akademik ---
Status Akademik
Lulus    1399
Tidak     101
Name: count, dtype: int64

--- Distribution of Target Variable 'Status Akademik' ---
Status Akademik
Lulus    1399
Tidak     101
Name: count, dtype: int64
```

2.3.4 Preprocessing Data

Saya akan memulai prapemrosesan dengan membuat salinan dari DataFrame asli untuk menghindari modifikasi pada data mentah. Kemudian, saya akan menghapus kolom 'Nama' karena merupakan pengidentifikasi dan tidak relevan untuk pelatihan model.

```
df_preprocessed = df.copy()
print("Original DataFrame head:")
print(df_preprocessed.head())

df_preprocessed = df_preprocessed.drop('Nama',
axis=1)
print("\nDataFrame after dropping 'Nama' column:")
print(df_preprocessed.head())
```

```
Original DataFrame head:
                Nama Jenis Kelamin  Umur Status Menikah  Kehadiran (%)  \
0    Anastasia Pradipta     Perempuan    24        Menikah             61
1       Salman Mahendra     Perempuan    29        Menikah             88
2  Mahfud Narpati, S.Gz     Laki-laki    26        Menikah             46
3    Bahuwirya Sihotang     Laki-laki    21  Belum Menikah             40
4  Puti Raisa Mardhiyah     Laki-laki    30  Belum Menikah             45

   Partisipasi Diskusi (skor)  Nilai Tugas (rata-rata)  \
0                          47                       60
1                          52                       95
2                          71                       77
3                          67                       45
4                          57                       82

   Aktivitas E-Learning (skor)   IPK Status Akademik
0                           98  2.27          Lulus
1                           90  2.35          Lulus
2                           97  3.30          Lulus
3                           91  3.84          Lulus
4                           94  2.30          Lulus

DataFrame after dropping 'Nama' column:
  Jenis Kelamin  Umur Status Menikah  Kehadiran (%)  \
0     Perempuan    24        Menikah             61
1     Perempuan    29        Menikah             88
2     Laki-laki    26        Menikah             46
3     Laki-laki    21  Belum Menikah             40
4     Laki-laki    30  Belum Menikah             45

   Partisipasi Diskusi (skor)  Nilai Tugas (rata-rata)  \
0                          47                       60
1                          52                       95
2                          71                       77
3                          67                       45
4                          57                       82

   Aktivitas E-Learning (skor)   IPK Status Akademik
0                           98  2.27          Lulus
1                           90  2.35          Lulus
2                           97  3.30          Lulus
3                           91  3.84          Lulus
4                           94  2.30          Lulus
```

Saya akan melakukan one-hot encoding pada kolom kategorikal 'Jenis Kelamin' dan 'Status Menikah', kemudian menerapkan label encoding pada variabel target 'Status Akademik'

untuk mengubahnya menjadi representasi numerik, sesuai dengan instruksi yang diberikan.

```
from sklearn.preprocessing import LabelEncoder

# One-hot encoding for 'Jenis Kelamin' and 'Status
Menikah'
df_preprocessed = pd.get_dummies(df_preprocessed,
columns=['Jenis Kelamin', 'Status Menikah'],
drop_first=True)
print("\nDataFrame after One-Hot Encoding:")
print(df_preprocessed.head())

# Label encoding for 'Status Akademik'
le = LabelEncoder()
df_preprocessed['Status Akademik'] =
le.fit_transform(df_preprocessed['Status
Akademik'])
print("\nDataFrame after Label Encoding 'Status
Akademik':")
print(df_preprocessed.head())
print("Label mapping for 'Status Akademik':",
list(le.classes_), le.transform(le.classes_))
```

```
DataFrame after One-Hot Encoding:
   Umur  Kehadiran (%)  Partisipasi Diskusi (skor)  Nilai Tugas (rata-rata)  \
0    24             61                          47                       60
1    29             88                          52                       95
2    26             46                          71                       77
3    21             40                          67                       45
4    30             45                          57                       82

   Aktivitas E-Learning (skor)   IPK Status Akademik  Jenis Kelamin_Perempuan  \
0                           98  2.27          Lulus                     True
1                           90  2.35          Lulus                     True
2                           97  3.30          Lulus                    False
3                           91  3.84          Lulus                    False
4                           94  2.30          Lulus                    False

   Status Menikah_Menikah
0                    True
1                    True
2                    True
3                   False
4                   False

DataFrame after Label Encoding 'Status Akademik':
   Umur  Kehadiran (%)  Partisipasi Diskusi (skor)  Nilai Tugas (rata-rata)  \
0    24             61                          47                       60
1    29             88                          52                       95
2    26             46                          71                       77
3    21             40                          67                       45
4    30             45                          57                       82

   Aktivitas E-Learning (skor)   IPK  Status Akademik  \
0                           98  2.27                0
1                           90  2.35                0
2                           97  3.30                0
3                           91  3.84                0
4                           94  2.30                0

   Jenis Kelamin_Perempuan  Status Menikah_Menikah
0                     True                    True
1                     True                    True
2                    False                    True
3                    False                   False
4                    False                   False
Label mapping for 'Status Akademik': ['Lulus', 'Tidak'] [0 1]
```

Saya akan mengimpor StandardScaler dari sklearn.preprocessing, mengidentifikasi kolom numerik yang memerlukan penskalaan, menerapkan scaler pada kolom-kolom tersebut, dan kemudian memperbarui DataFrame df_preprocessed dengan nilai yang telah diskalakan. Terakhir, saya akan menampilkan lima baris pertama dan .info() dari DataFrame untuk memverifikasi langkah-langkah prapemrosesan tersebut.

```
from sklearn.preprocessing import StandardScaler

# Identify numerical columns for scaling (excluding
the target variable if it's numerical after encoding)
numerical_cols_to_scale = ['Umur', 'Kehadiran (%)',
'Partisipasi Diskusi (skor)', 'Nilai Tugas (rata-
rata)', 'Aktivitas E-Learning (skor)', 'IPK']

# Initialize StandardScaler
scaler = StandardScaler()

# Apply scaling to numerical columns
df_preprocessed[numerical_cols_to_scale] =
scaler.fit_transform(df_preprocessed[numerical_cols_to
_scale])

print("\nDataFrame after Scaling Numerical Features:")
print(df_preprocessed.head())

print("\nDataFrame Info after all preprocessing:")
df_preprocessed.info()
```

```
DataFrame after Scaling Numerical Features:
       Umur  Kehadiran (%)  Partisipasi Diskusi (skor)  \
0 -0.151819      -0.500891                   -1.313644
1  1.293161       1.031985                   -1.029560
2  0.426173      -1.352489                    0.049961
3 -1.018808      -1.693128                   -0.177307
4  1.582157      -1.409262                   -0.745475

   Nilai Tugas (rata-rata)  Aktivitas E-Learning (skor)       IPK  \
0                -0.515030                    1.526399 -1.282015
1                 1.504469                    1.078293 -1.141765
2                 0.465870                    1.470386  0.523706
3                -1.380530                    1.134306  1.470394
4                 0.754370                    1.302346 -1.229421

   Status Akademik  Jenis Kelamin_Perempuan  Status Menikah_Menikah
0                0                     True                    True
1                0                     True                    True
2                0                    False                    True
3                0                    False                   False
4                0                    False                   False

DataFrame Info after all preprocessing:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 9 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Umur                        1500 non-null   float64
 1   Kehadiran (%)               1500 non-null   float64
 2   Partisipasi Diskusi (skor)  1500 non-null   float64
 3   Nilai Tugas (rata-rata)     1500 non-null   float64
 4   Aktivitas E-Learning (skor) 1500 non-null   float64
 5   IPK                         1500 non-null   float64
 6   Status Akademik             1500 non-null   int64
 7   Jenis Kelamin_Perempuan     1500 non-null   bool
 8   Status Menikah_Menikah      1500 non-null   bool
dtypes: bool(2), float64(6), int64(1)
memory usage: 85.1 KB
```

2.3.5 Split Data (Train/Test)

Pertama, saya akan mengubah kolom boolean menjadi integer (0 atau 1) karena beberapa model mungkin tidak dapat menangani tipe boolean secara langsung. Kemudian, saya akan memisahkan fitur (X) dari variabel target (y), yaitu 'Status Akademik'. Terakhir, saya akan membagi data menjadi set pelatihan dan pengujian menggunakan train_test_split dengan rasio 80/20, memastikan reproduktifitas dengan random_state=42 dan menjaga distribusi kelas dengan stratify=y.

```
from sklearn.model_selection import
train_test_split

# Convert boolean columns to int (0 or 1)
df_preprocessed['Jenis Kelamin_Perempuan'] =
df_preprocessed['Jenis
Kelamin_Perempuan'].astype(int)
df_preprocessed['Status Menikah_Menikah'] =
df_preprocessed['Status
Menikah_Menikah'].astype(int)

# Separate features (X) and target variable (y)
```

```
X = df_preprocessed.drop('Status Akademik', axis=1)
y = df_preprocessed['Status Akademik']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)

print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_test:", y_test.shape)

print("\nDistribution of target variable in
original dataset:\n",
y.value_counts(normalize=True))
print("\nDistribution of target variable in
training set:\n",
y_train.value_counts(normalize=True))
print("\nDistribution of target variable in testing
set:\n", y_test.value_counts(normalize=True))
```

```
Shape of X_train: (1200, 8)
Shape of X_test: (300, 8)
Shape of y_train: (1200,)
Shape of y_test: (300,)

Distribution of target variable in original dataset:
 Status Akademik
0    0.932667
1    0.067333
Name: proportion, dtype: float64

Distribution of target variable in training set:
 Status Akademik
0    0.9325
1    0.0675
Name: proportion, dtype: float64

Distribution of target variable in testing set:
 Status Akademik
0    0.933333
1    0.066667
Name: proportion, dtype: float64
```

2.3.6 Model Training

Saya akan mengimpor DecisionTreeClassifier dari sklearn.tree, menginisialisasinya dengan criterion='entropy' dan random_state=42 untuk mensimulasikan algoritma C4.5, melatih model tersebut menggunakan X_train dan y_train, kemudian mencetak parameter model untuk verifikasi.

```
from sklearn.tree import DecisionTreeClassifier

# Initialize DecisionTreeClassifier with
criterion='entropy' for C4.5 and random_state for
reproducibility
dtree_model =
DecisionTreeClassifier(criterion='entropy',
random_state=42)

# Train the Decision Tree model
dtree_model.fit(X_train, y_train)

print("Decision Tree Model Trained Successfully!")
print("Model Parameters:")
print(dtree_model.get_params())
```

```
Decision Tree Model Trained Successfully!
Model Parameters:
{'ccp_alpha': 0.0, 'class_weight': None,
'criterion': 'entropy', 'max_depth': None,
'max_features': None, 'max_leaf_nodes': None,
'min_impurity_decrease': 0.0, 'min_samples_leaf':
1, 'min_samples_split': 2,
'min_weight_fraction_leaf': 0.0, 'monotonic_cst':
None, 'random_state': 42, 'splitter': 'best'}
```

2.3.7 Evaluasi Model

Saya akan mengimpor metrik klasifikasi yang diperlukan dari sklearn.metrics, menggunakan dtree_model yang telah dilatih untuk membuat prediksi pada set pengujian X_test, dan kemudian menghitung serta mencetak akurasi, presisi, recall, dan f1-score dari model tersebut.

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

# Make predictions on the test set
y_pred = dtree_model.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Model Accuracy: {accuracy:.4f}")
print(f"Model Precision: {precision:.4f}")
print(f"Model Recall: {recall:.4f}")
print(f"Model F1-score: {f1:.4f}")
```

```
Model Accuracy: 0.9200
Model Precision: 0.4286
Model Recall: 0.6000
Model F1-score: 0.5000
```

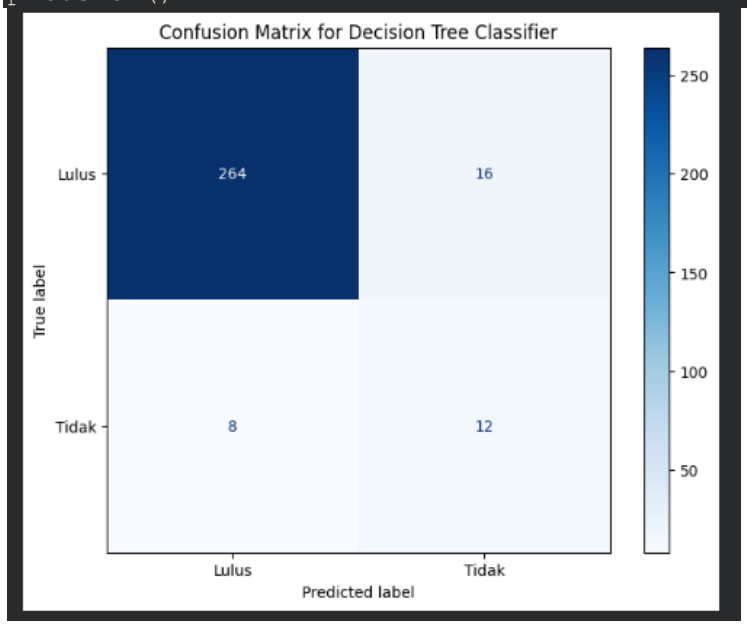
2.3.8 Visualisasi Hasil

Visualisasi Confusion Matrix

Saya akan mengimpor pustaka yang diperlukan, menghitung confusion matrix menggunakan y_test dan y_pred, membuat objek ConfusionMatrixDisplay dengan label kelas, memplotnya beserta judul, dan menampilkan plot tersebut untuk memvisualisasikan performa model.

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred)
```

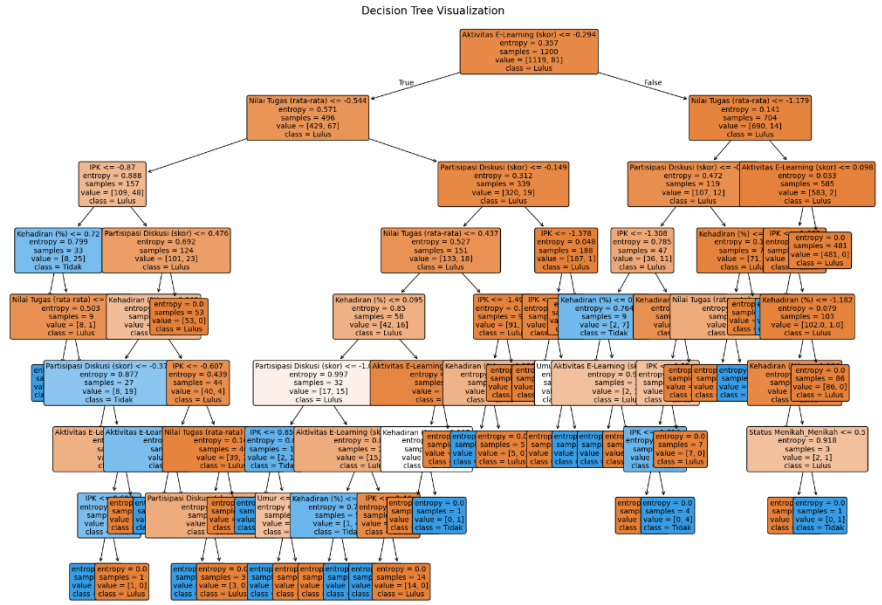
```
# Get class names from the label encoder
class_names = ['Lulus', 'Tidak'] # Based on label
mapping: ['Lulus', 'Tidak'] [0 1]

# Create a ConfusionMatrixDisplay object
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=class_names)
# Plot the confusion matrix
fig, ax = plt.subplots(figsize=(8, 6))
disp.plot(cmap=plt.cm.Blues, ax=ax)

# Add title
ax.set_title('Confusion Matrix for Decision Tree
Classifier')

# Display the plot
plt.show()
```



Visualisasi Decision Tree

Untuk memvisualisasikan decision tree, saya akan mengimpor fungsi plot_tree, mengatur ukuran gambar agar mudah dibaca, lalu menggunakan plot_tree dengan model yang telah dilatih, nama fitur, nama kelas, serta opsi tampilan seperti filled dan rounded. Terakhir, saya akan menambahkan judul dan menampilkan plot tersebut.

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

# Set the figure size to ensure the tree is
readable
plt.figure(figsize=(20, 15))

# Visualize the Decision Tree
plot_tree(dtree_model,
          feature_names=X_train.columns,
          class_names=['Lulus', 'Tidak'], # Use the
actual class names from label encoding
          filled=True,
          rounded=True,
          fontsize=10)

plt.title('Decision Tree Visualization',
fontsize=15)
plt.show()
```



2.4 Hasil dan Pembahasan

Berdasarkan hasil pengujian pada dataset mahasiswa, berikut adalah perbandingan performa kedua algoritma:

Algoritma	Akurasi	Precision	Recall	F1-Score
C4.5 (Decision Tree)	92%	0.4286	0.6000	0.5000
Naive Bayes	96.33%	1.0000	0.4500	0.6200

Narasi Hasil:

Berdasarkan hasil pengujian, algoritma C4.5 menghasilkan akurasi sebesar 92%. Meskipun akurasinya sedikit di bawah Naive Bayes, algoritma C4.5 menunjukkan performa yang lebih baik dalam aspek Recall (0.60). Hal ini mengindikasikan bahwa C4.5 lebih sensitif dalam mendeteksi mahasiswa yang berisiko 'Tidak Lulus' dibandingkan model lainnya, yang sangat penting dalam konteks deteksi dini kegagalan akademik.

Hasil penelitian ini memperkuat temuan Setiyani dkk. (2020) bahwa metode Naive Bayes memiliki performa yang sangat baik dalam melakukan klasifikasi status kelulusan, yang dalam eksperimen ini mencapai akurasi sebesar 96,33%.

BAB III

PENUTUP

3.1 Kesimpulan

- Preprocessing: Langkah prapemrosesan seperti One-Hot Encoding dan Standard Scaling telah berhasil dilakukan pada 1.500 data tanpa adanya data yang hilang (missing values).
- Performa Model: Algoritma C4.5 yang dilatih dengan kriteria entropy berhasil memprediksi status akademik dengan akurasi 92%.
- Temuan Utama: Model menunjukkan kemampuan yang cukup baik dalam mengidentifikasi kelas minoritas ('Tidak Lulus') dengan tingkat Recall sebesar 60%, meskipun terdapat ketidakseimbangan kelas yang signifikan (93,27% vs 6,73%).

3.2 Saran

- Penanganan Imbalance Data: Mengingat adanya ketimpangan data yang mencolok (hanya 6,73% data 'Tidak Lulus'), disarankan menggunakan teknik SMOTE (Synthetic Minority Over-sampling Technique) untuk meningkatkan Precision dan F1-Score pada kelas minoritas.
- Optimasi Model: Melakukan penalaan hiperparameter (Hyperparameter Tuning) seperti mengatur max_depth atau min_samples_leaf pada Decision Tree untuk mencegah overfitting dan meningkatkan generalisasi model.

DAFTAR PUSTAKA

Ashari Muin, A. (2016). Metode Naive Bayes Untuk Prediksi Kelulusan (Studi Kasus: Data Mahasiswa Baru Perguruan Tinggi). *Jurnal Ilmiah Ilmu Komputer*, *2*(1). http://ejournal.fikom-unasman.ac.id

Puspita, R., Putri, S., & Waspada, I. (2018). Penerapan Algoritma C4.5 pada Aplikasi Prediksi Kelulusan Mahasiswa Prodi Informatika. *Jurnal Ilmu Komputer dan Informatika*, *1*(1).

Setiyani, L., Wahidin, M., Awaludin, D., & Purwani, S. (2020). Analisis Prediksi Kelulusan Mahasiswa Tepat Waktu Menggunakan Metode Data Mining Naïve Bayes: Systematic Review. *Faktor Exacta*, *13*(1), 35. https://doi.org/10.30998/faktorexacta.v13i1.5548