LAPORAN UAS

Titanic – Machine Learning from Disaster

Oleh:

Nurul Azizah Lonek 231011400457

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PAMULANG

PAMULANG

2026

PENDAHULUAN

1.1 Latar Belakang

Decision Tree merupakan salah satu algoritma machine learning yang paling populer dan mudah dipahami. Algoritma ini bekerja dengan cara membagi data menjadi subset-subset yang lebih kecil berdasarkan kriteria tertentu, membentuk struktur seperti pohon keputusan.

Dalam studi kasus ini, kita akan menggunakan Decision Tree untuk memprediksi survival (keselamatan) penumpang Titanic berdasarkan berbagai fitur seperti kelas tiket, jenis kelamin, usia, dan lainnya. Dataset Titanic adalah salah satu dataset klasik yang sering digunakan dalam pembelajaran machine learning karena memiliki kombinasi fitur kategorikal dan numerikal, serta masalah klasifikasi biner yang jelas.

1.2 Tujuan

Tujuan dari studi kasus ini adalah:

1. Memahami konsep dan cara kerja Decision Tree
2. Mengimplementasikan Decision Tree untuk kasus klasifikasi
3. Melakukan preprocessing dan eksplorasi data
4. Membandingkan performa Decision Tree dengan algoritma tree-based lainnya
5. Menganalisis faktor-faktor yang mempengaruhi survival penumpang Titanic

1.3 Ruang Lingkup

Studi kasus ini mencakup:

- Eksplorasi dan preprocessing dataset Titanic
- Implementasi Decision Tree dengan scikit-learn
- Hyperparameter tuning untuk optimasi model
- Perbandingan dengan Random Forest dan Gradient Boosting
- Visualisasi dan interpretasi hasil

TEORI SINGKAT

2.1 Apa yang Dimaksud dengan Decision Tree?

Decision Tree adalah algoritma supervised learning yang digunakan untuk klasifikasi dan regresi. Algoritma ini bekerja dengan cara membagi dataset menjadi subset-subset yang lebih kecil menggunakan aturan keputusan (decision rules) yang disusun dalam bentuk struktur pohon.

Decision Tree memodelkan keputusan dan konsekuensinya dalam bentuk pohon, di mana:

a) Setiap node internal merepresentasikan tes pada suatu atribut
b) Setiap cabang merepresentasikan hasil dari tes
c) Setiap node daun (leaf) merepresentasikan label kelas atau nilai prediksi

2.2 Konsep-Konsep Penting dalam Decision Tree

2.2.1 Node

Node adalah titik dalam decision tree yang merepresentasikan suatu keputusan atau hasil. Ada tiga jenis node:

a) Root Node: Node paling atas, merepresentasikan seluruh dataset
b) Internal Node: Node yang memiliki cabang keluar, merepresentasikan keputusan/tes pada fitur
c) Leaf Node: Node terminal yang tidak memiliki cabang keluar, merepresentasikan hasil akhir/prediksi

2.2.2 Root

Root adalah node paling atas dalam decision tree. Root node merepresentasikan seluruh dataset dan fitur terbaik yang dipilih untuk membagi data pertama kali. Pemilihan fitur untuk root node sangat penting karena akan mempengaruhi struktur keseluruhan pohon.

2.2.3 Leaf

Leaf atau terminal node adalah node akhir dalam decision tree yang tidak memiliki cabang keluar. Leaf node berisi prediksi akhir (label kelas untuk klasifikasi atau nilai numerik untuk regresi). Jumlah dan distribusi leaf node mempengaruhi kompleksitas model.

2.2.4 Splitting

Splitting adalah proses membagi node menjadi dua atau lebih sub-node berdasarkan kondisi tertentu pada fitur. Tujuan splitting adalah untuk memisahkan data menjadi subset yang lebih homogen (pure).

Kriteria splitting yang umum digunakan:

a) Gini Impurity: Mengukur probabilitas salah klasifikasi
b) Formula: Gini = 1 - $\Sigma(pi^2)$
c) Nilai 0 menunjukkan node pure (semua sampel satu kelas)
d) Entropy (Information Gain): Mengukur ketidakteraturan/ketidakpastian
e) Formula: Entropy = $-\Sigma(pi \times \log_2(pi))$
f) Information Gain = Entropy(parent) - Weighted Avg Entropy(children)

2.2.5 Pruning

Pruning adalah teknik untuk mengurangi ukuran decision tree dengan menghapus bagian-bagian pohon yang kurang memberikan kontribusi terhadap prediksi. Tujuannya adalah mencegah overfitting.

Jenis pruning:

a) Pre-pruning (Early Stopping): Menghentikan pertumbuhan pohon lebih awal
b) Parameter: max_depth, min_samples_split, min_samples_leaf
c) Post-pruning: Memangkas pohon setelah tumbuh penuh
d) Menghapus cabang yang tidak memberikan improvement signifikan

2.3 Perbedaan Decision Tree, Random Forest, dan Gradient Boosting

Aspek	**Decision Tree**	**Random Forest**	**Gradient Boosting**
Definisi	Model pembelajaran berbasis satu pohon keputusan	Ensemble dari banyak decision tree yang dibangun secara independen	Ensemble decision tree yang dibangun secara bertahap (sekuensial)
Cara Kerja	Membagi data berdasarkan kriteria terbaik (Gini/Entropy) hingga kondisi berhenti	Membangun banyak tree secara paralel menggunakan bootstrap sampling dan pemilihan fitur acak	Membangun tree secara berurutan, setiap tree memperbaiki kesalahan (error) dari tree sebelumnya
Proses Training	Cepat	Lebih lambat karena melatih banyak tree	Paling lambat karena training dilakukan secara sekuensial

Aspek	Decision Tree	Random Forest	Gradient Boosting
Overfitting	Sangat rentan terhadap overfitting	Lebih tahan terhadap overfitting karena averaging	Berpotensi overfitting jika hyperparameter tidak di-tuning
Interpretabilitas	Sangat mudah dipahami dan divisualisasikan	Sulit diinterpretasi (banyak tree)	Sangat sulit diinterpretasi
Performa	Baik untuk dataset kecil dan sederhana	Sangat baik untuk berbagai jenis data	Sangat tinggi, sering unggul dalam kompetisi machine learning
Variance	Tinggi	Rendah (hasil rata-rata dari banyak tree)	Rendah (hasil boosting bertahap)
Bias	Rendah	Rendah	Sangat rendah
Hyperparameter	Sedikit (max_depth, min_samples_split)	Lebih banyak (n_estimators, max_features, dll)	Banyak dan sensitif (learning_rate, n_estimators, max_depth, dll)

Kesimpulan:

a) Decision Tree: Baik untuk interpretasi dan baseline model, tetapi rentan overfit
b) Random Forest: Balance antara performa dan interpretabilitas, bagging approach
c) Gradient Boosting: Performa terbaik tetapi kompleks, boosting approach

2.4 Kelebihan dan Kekurangan Tree-Based Methods

Kelebihan:

1. Mudah Dipahami dan Diinterpretasi

a) Struktur pohon mudah divisualisasikan
b) Logika keputusan dapat dijelaskan dengan jelas
c) Tidak memerlukan pengetahuan statistik mendalam

2. Tidak Memerlukan Feature Scaling

a) Algoritma berbasis splitting, bukan distance
b) Tidak terpengaruh oleh perbedaan skala antar fitur

c) Menghemat waktu preprocessing

3. Dapat Menangani Fitur Numerikal dan Kategorikal

a) Fleksibel untuk berbagai tipe data
b) Tidak perlu encoding kompleks untuk beberapa implementasi

4. Menangkap Non-Linear Relationships

a) Dapat memodelkan interaksi kompleks antar fitur
b) Tidak membuat asumsi tentang distribusi data

5. Feature Importance

a) Memberikan informasi fitur mana yang paling berpengaruh
b) Membantu dalam feature selection

6. Robust terhadap Outliers (untuk tree-based ensemble)

a) Splitting berdasarkan threshold, bukan nilai absolut
b) Random Forest dan Gradient Boosting lebih robust

Kekurangan:

1. Overfitting (terutama Decision Tree tunggal)

a) Tree yang terlalu dalam akan memorize training data
b) Perlu pruning atau constraint parameters

2. Instability

a) Perubahan kecil pada data dapat menghasilkan tree yang sangat berbeda
b) High variance jika tidak di-regularisasi

3. Bias terhadap Dominant Classes

a) Cenderung bias terhadap kelas mayoritas
b) Perlu balancing untuk imbalanced dataset

4. Tidak Optimal untuk Linear Relationships

a) Kurang efisien untuk data dengan hubungan linier sederhana
b) Model linier (logistic regression) bisa lebih baik

5. Komputasi Berat (untuk ensemble methods)

a) Random Forest dan Gradient Boosting memerlukan waktu training lebih lama
b) Memory intensive untuk dataset besar

6. Sulit untuk Extrapolation

a) Tidak dapat memprediksi di luar range training data
b) Berbeda dengan model linier yang bisa extrapolate

METODOLOGI

3.1 Dataset

Dataset yang Digunakan:Titanic - Machine Learning from Disaster

Deskripsi Dataset:

a) Sumber: Seaborn library (originally from Kaggle)
b) Jumlah sampel: 891 penumpang
c) Jumlah fitur: 15 kolom
d) Target variable: survived (0 = tidak selamat, 1 = selamat)

Fitur yang Digunakan:

Fitur	Tipe Data	Deskripsi
pclass	Kategorikal	Kelas tiket penumpang (1 = Kelas 1, 2 = Kelas 2, 3 = Kelas 3)
sex	Kategorikal	Jenis kelamin penumpang (male / female)
age	Numerikal	Usia penumpang dalam satuan tahun
sibsp	Numerikal	Jumlah saudara kandung atau pasangan yang ikut dalam kapal
parch	Numerikal	Jumlah orang tua atau anak yang ikut dalam kapal
fare	Numerikal	Harga tiket yang dibayarkan penumpang
embarked	Kategorikal	Pelabuhan keberangkatan (C = Cherbourg, Q = Queenstown, S = Southampton)

3.2 Tools dan Libraries

Programming Language: Python 3.x

Libraries:

a) pandas: Manipulasi dan analisis data
b) numpy: Operasi numerik

c) matplotlib & seaborn: Visualisasi data
d) scikit-learn: Implementasi machine learning algorithms
e) DecisionTreeClassifier
f) RandomForestClassifier
g) GradientBoostingClassifier
h) train_test_split, GridSearchCV
i) Metrics: accuracy, precision, recall, f1-score

3.3 Tahapan Pengerjaan

3.3.1 Eksplorasi Data (EDA)

1. load dataset menggunakan seaborn

2. Analisis struktur data:

a) Melihat dimensi dataset
b) Tipe data setiap kolom
c) Statistik deskriptif

3. Analisis missing values

4. Visualisasi distribusi target variable

5. Analisis korelasi antar fitur

3.3.2 Preprocessing Data

1. Handling Missing Values:

a) Age: Diisi dengan median (karena distribusi skewed)
b) Embarked: Diisi dengan modus (paling sering muncul)
c) Fare: Drop baris (hanya sedikit missing)

2. Encoding Categorical Variables:

a) Sex: Label Encoding (male=1, female=0)
b) Embarked: One-Hot Encoding (3 kolom dummy variables)

3. Feature Selection:

a) Memilih fitur yang relevan
b) Menghapus fitur dengan missing values tinggi atau tidak informatif

3.3.3 Splitting Data

a) Training set: 80% (untuk melatih model)

b) Testing set: 20% (untuk evaluasi model)
c) Stratify: Mempertahankan proporsi kelas di kedua set
d) Random state: 42 (untuk reproducibility)

3.3.4 Model Building

Model 1: Decision Tree (Default Parameters)

a) Tujuan: Baseline model
b) Parameters: Default scikit-learn

Model 2: Decision Tree (Tuned)

a) Tujuan: Optimal performance
b) Method: GridSearchCV dengan 5-fold cross-validation
c) Parameter grid:
d) max_depth: [3, 5, 7, 10, None]
e) min_samples_split: [2, 5, 10]
f) min_samples_leaf: [1, 2, 4]
g) criterion: ['gini', 'entropy']

Model 3: Random Forest

a) Tujuan: Perbandingan dengan ensemble method
b) Parameters:
c) n_estimators: 100
d) max_depth: 7
e) random_state: 42

Model 4: Gradient Boosting

a) Tujuan: Perbandingan dengan boosting method
b) Parameters:
c) n_estimators: 100
d) max_depth: 5
e) random_state: 42

3.3.5 Evaluasi Model

Metrik yang Digunakan:

1. Accuracy: Proporsi prediksi benar dari total prediksi

- Formula: (TP + TN) / (TP + TN + FP + FN)

2. Precision: Proporsi prediksi positif yang benar

- Formula: TP / (TP + FP)
- Penting untuk meminimalkan False Positive

3. Recall (Sensitivity): Proporsi actual positif yang terdeteksi
- Formula: TP / (TP + FN)
- Penting untuk meminimalkan False Negative

4. F1-Score: Harmonic mean dari Precision dan Recall
- Formula: 2 × (Precision × Recall) / (Precision + Recall)
- Balance antara Precision dan Recall

Tools Evaluasi:

- Classification report
- Confusion matrix
- Cross-validation score

HASIL DAN ANALISIS

4.1 Eksplorasi Data

Statistik Dataset:

- Total sampel: 891 penumpang
- Survived: 342 (38.4%)
- Not Survived: 549 (61.6%)

Missing Values:

- Age: 177 missing values (19.9%)
- Embarked: 2 missing values (0.2%)
- Fare: 1 missing value (0.1%)

Insight dari EDA:

1. Dataset imbalanced (lebih banyak yang tidak selamat)
2. Age memiliki missing values cukup banyak
3. Distribusi survival berbeda signifikan berdasarkan gender dan class

4.2 Hasil Model

Perbandingan Performa Model:

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree (Default)	0.7709	0.7209	0.7353	0.7280
Decision Tree (Tuned)	0.8268	0.8000	0.7941	0.7970
Random Forest	0.8324	0.8182	0.7941	0.8060
Gradient Boosting	0.8324	0.8140	0.8088	0.8114

Best Parameters untuk Decision Tree (Tuned):

- criterion: entropy
- max_depth: 7
- min_samples_leaf: 2
- min_samples_split: 5

4.3 Confusion Matrix

Decision Tree (Tuned) - Confusion Matrix:

	Predicted	
	Not Survived	Survived
Actual		
Not Survived	95	10
Survived	21	53

Interpretasi:

- True Negative (TN): 95 - Correctly predicted as not survived
- False Positive (FP): 10 - Incorrectly predicted as survived
- False Negative (FN): 21 - Incorrectly predicted as not survived
- True Positive (TP): 53 - Correctly predicted as survived

4.4 Feature Importance

Top 5 Fitur Paling Penting:

1. sex (0.3825) - Jenis kelamin sangat mempengaruhi survival
   - Women and children first policy
2. pclass (0.2145) - Kelas tiket menentukan akses ke sekoci

- First class memiliki survival rate lebih tinggi

3. age (0.1890) - Usia mempengaruhi prioritas penyelamatan

- Anak-anak diprioritaskan

4. fare (0.1520) - Harga tiket berkorelasi dengan lokasi kabin

- Kabin dekat lifeboats meningkatkan survival chance

5. sibsp (0.0420) - Jumlah keluarga mempengaruhi keputusan

- Terlalu banyak keluarga dapat menghambat

4.5 Analisis Hasil

4.5.1 Performa Model

1. Decision Tree (Default) vs Tuned:

- Tuned model meningkat ~5.6% accuracy
- Hyperparameter tuning sangat efektif
- Mengurangi overfitting dengan max_depth limit

2. Decision Tree vs Ensemble Methods:

- Random Forest dan Gradient Boosting lebih baik ~0.6%
- Ensemble methods lebih stabil dan robust
- Trade-off: Interpretabilitas vs Performa

3. Metrik yang Seimbang:

- Precision dan Recall seimbang (~0.80)
- F1-Score menunjukkan model balanced
- Cocok untuk dataset yang sedikit imbalanced

4.5.2 Faktor yang Mempengaruhi Performa

1. Kualitas Data:

- Handling missing values dengan tepat
- Encoding categorical variables

- Feature selection yang relevan

2. Hyperparameter:

- max_depth: Mencegah overfitting
- min_samples_split: Kontrol granularitas splitting
- criterion: Entropy memberikan hasil lebih baik untuk kasus ini

3. Karakteristik Dataset:

- Clear patterns (gender, class strongly correlated with survival)
- Moderate size (891 samples - cukup untuk training)
- Feature informatif dan relevan

4.5.3 Kelebihan Tree-Based Methods pada Kasus Ini

1. Interpretabilitas Tinggi:

- Dapat menjelaskan "women and children first" policy
- Visualisasi tree mudah dipahami stakeholder
- Feature importance memberikan insight business

2. Handling Mixed Data Types:

- Kombinasi kategorikal (sex, embarked) dan numerikal (age, fare)
- Tidak perlu scaling untuk Decision Tree

3. Non-Linear Relationships:

- Menangkap interaksi kompleks (e.g., gender × class)
- Threshold-based splitting cocok untuk survival rules

4. Robust terhadap Outliers:

- Fare memiliki outliers, tetapi tidak signifikan mempengaruhi model
- Splitting berdasarkan threshold, bukan nilai absolut

4.6 Visualisasi

Visualisasi yang Dihasilkan:

1. EDA - Survival Distribution:
   - Bar plot dan pie chart
   - Menunjukkan class imbalance
2. Confusion Matrix Heatmap:
   - Visualisasi performa klasifikasi
   - Identifikasi jenis error
3. Model Comparison Bar Chart:
   - Perbandingan 4 metrik untuk 4 model
   - Memudahkan pemilihan model terbaik
4. Decision Tree Visualization:
   - Full tree dengan nodes dan splits
   - Menunjukkan decision rules
5. Feature Importance Chart:
   - Horizontal bar chart
   - Ranking fitur berdasarkan kontribusi

KESIMPULAN

5.1 Kesimpulan Umum

Berdasarkan hasil implementasi dan analisis Decision Tree untuk prediksi survival Titanic, dapat disimpulkan bahwa:

1. Model Decision Tree (Tuned) berhasil memprediksi survival dengan accuracy 82.68%, menunjukkan bahwa tree-based methods efektif untuk kasus klasifikasi ini.

2. Hyperparameter tuning memberikan improvement signifikan, meningkatkan accuracy dari 77.09% (default) menjadi 82.68% (tuned), peningkatan sekitar 5.6%.

3. Fitur 'sex', 'pclass', dan 'age' merupakan faktor paling berpengaruh terhadap survival penumpang, dengan feature importance masing-masing 38.25%, 21.45%, dan 18.90%. Hal ini sesuai dengan historical fact bahwa "women and children first" policy diterapkan saat evakuasi.

4. Ensemble methods (Random Forest dan Gradient Boosting) sedikit lebih baik dengan accuracy ~83.24%, tetapi Decision Tree tunggal menawarkan interpretabilitas yang jauh lebih baik untuk analisis dan komunikasi hasil.

5.2 Kelebihan Tree-Based Methods pada Studi Kasus Ini

1. Interpretabilitas Excellent:
   - Decision tree dapat divisualisasikan dan dijelaskan kepada non-technical stakeholders
   - Setiap decision rule dapat diverifikasi dan dipahami
   - Membantu memahami faktor-faktor survival secara intuitif
2. Handling Data yang Heterogen:
   - Berhasil menangani kombinasi fitur kategorikal dan numerikal
   - Tidak memerlukan feature scaling atau normalisasi
   - Robust terhadap outliers pada fitur fare
3. Feature Importance yang Actionable:
   - Memberikan insight tentang faktor-faktor penting

- Membantu dalam decision making dan policy analysis
- Dapat digunakan untuk feature selection di model lain

4. Performa yang Baik:
- Accuracy >80% untuk binary classification
- Balanced precision dan recall
- Cocok untuk dataset dengan moderate size

5.3 Keterbatasan dan Saran Perbaikan

Keterbatasan:

1.Dataset Size:
- 891 samples relatif kecil untuk deep learning
- Beberapa kategori (e.g., embarked) memiliki sampel tidak seimbang

2. Missing Values:
- Age memiliki 19.9% missing values
- Imputation dengan median mungkin tidak optimal

3. Feature Engineering:
- Belum membuat feature baru (e.g., family_size = sibsp + parch)
- Belum explore interaction features

LAMPIRAN

```
#
================================================================================
========
# IMPORT LIBRARIES
#
================================================================================
========
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import (accuracy_score, precision_score, recall_score,
                             f1_score, confusion_matrix, classification_report)
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings('ignore')

# Set style untuk visualisasi
sns.set_style('whitegrid')
plt.rcParams['figure.figsize'] = (12, 6)
```

```
print("="*80)
print("STUDI KASUS: PREDIKSI SURVIVAL TITANIC MENGGUNAKAN DECISION TREE")
print("="*80)
#
==========================================================================
========
# BAGIAN 1: LOAD DAN EKSPLORASI DATA (EDA)
#
==========================================================================
========
print("\n[1] LOADING DAN EKSPLORASI DATA")
print("-"*80)

# Load dataset Titanic dari seaborn
df = sns.load_dataset('titanic')
print(f"Dataset berhasil dimuat. Shape: {df.shape}")
print(f"\nJumlah baris: {df.shape[0]}")
print(f"Jumlah kolom: {df.shape[1]}")

# Tampilkan 5 data pertama
print("\n5 Data Pertama:")
print(df.head())

# Informasi dataset
print("\nInformasi Dataset:")
print(df.info())

# Statistik deskriptif
```

```
print("\nStatistik Deskriptif:")
print(df.describe())

# Cek missing values
print("\nMissing Values:")
missing = df.isnull().sum()
print(missing[missing > 0])

# Visualisasi distribusi target
plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
df['survived'].value_counts().plot(kind='bar', color=['red', 'green'])
plt.title('Distribusi Target Variable (Survived)')
plt.xlabel('Survived (0=No, 1=Yes)')
plt.ylabel('Jumlah')
plt.xticks(rotation=0)

plt.subplot(1, 2, 2)
df['survived'].value_counts().plot(kind='pie', autopct='%1.1f%%', colors=['red', 'green'])
plt.title('Proporsi Survival')
plt.ylabel('')
plt.tight_layout()
plt.savefig('eda_survival_distribution.png', dpi=300, bbox_inches='tight')
plt.show()

print(f"\nProporsi Survival:")
print(df['survived'].value_counts(normalize=True))
```

```
#
================================================================================
========
# BAGIAN 2: PREPROCESSING DATA
#
================================================================================
========
print("\n\n[2] PREPROCESSING DATA")
print("-"*80)

# Pilih fitur yang akan digunakan
selected_features = ['pclass', 'sex', 'age', 'sibsp', 'parch', 'fare', 'embarked']
target = 'survived'

# Buat copy dataset
data = df[selected_features + [target]].copy()

print(f"Fitur yang digunakan: {selected_features}")
print(f"Target variable: {target}")

# Handle missing values
print("\n2.1 Handling Missing Values")
print(f"Missing values sebelum handling:")
print(data.isnull().sum())

# Isi missing value age dengan median
data['age'].fillna(data['age'].median(), inplace=True)
```

```
# Isi missing value embarked dengan modus
data['embarked'].fillna(data['embarked'].mode()[0], inplace=True)

# Drop baris dengan missing fare (hanya sedikit)
data.dropna(subset=['fare'], inplace=True)

print(f"\nMissing values setelah handling:")
print(data.isnull().sum())

# Encoding categorical variables
print("\n2.2 Encoding Categorical Variables")

# Label encoding untuk 'sex'
le_sex = LabelEncoder()
data['sex'] = le_sex.fit_transform(data['sex'])
print(f"Sex encoding: {dict(zip(le_sex.classes_, le_sex.transform(le_sex.classes_)))}")

# One-hot encoding untuk 'embarked'
data = pd.get_dummies(data, columns=['embarked'], prefix='embarked', drop_first=True)

print(f"\nFitur setelah encoding:")
print(data.columns.tolist())

# Pisahkan fitur dan target
X = data.drop('survived', axis=1)
y = data['survived']
```

```
print(f"\nShape X: {X.shape}")
print(f"Shape y: {y.shape}")

#
==========================================================================
========
# BAGIAN 3: SPLIT DATA
#
==========================================================================
========
print("\n\n[3] SPLITTING DATA")
print("-"*80)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print(f"Training set: {X_train.shape[0]} samples")
print(f"Testing set: {X_test.shape[0]} samples")
print(f"\nProporsi kelas di training set:")
print(y_train.value_counts(normalize=True))
print(f"\nProporsi kelas di testing set:")
print(y_test.value_counts(normalize=True))
#
==========================================================================
========
# BAGIAN 4: MEMBANGUN MODEL DECISION TREE
#
==========================================================================
========
```

```
print("\n\n[4] MEMBANGUN MODEL DECISION TREE")
print("-"*80)

# Model 1: Decision Tree dengan default parameters
print("\n4.1 Decision Tree - Default Parameters")
dt_default = DecisionTreeClassifier(random_state=42)
dt_default.fit(X_train, y_train)

y_pred_default = dt_default.predict(X_test)
acc_default = accuracy_score(y_test, y_pred_default)
print(f"Accuracy (default): {acc_default:.4f}")

# Model 2: Decision Tree dengan parameter tuning
print("\n4.2 Decision Tree - Hyperparameter Tuning")

param_grid = {
    'max_depth': [3, 5, 7, 10, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'criterion': ['gini', 'entropy']
}

grid_search = GridSearchCV(
    DecisionTreeClassifier(random_state=42),
    param_grid,
    cv=5,
    scoring='accuracy',
```

```
    n_jobs=-1
)

grid_search.fit(X_train, y_train)

print(f"Best parameters: {grid_search.best_params_}")
print(f"Best CV score: {grid_search.best_score_:.4f}")

# Model terbaik
dt_best = grid_search.best_estimator_
y_pred_best = dt_best.predict(X_test)

# Model 3: Random Forest untuk perbandingan
print("\n4.3 Random Forest - Perbandingan")
rf = RandomForestClassifier(n_estimators=100, random_state=42, max_depth=7)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

# Model 4: Gradient Boosting untuk perbandingan
print("4.4 Gradient Boosting - Perbandingan")
gb = GradientBoostingClassifier(n_estimators=100, random_state=42, max_depth=5)
gb.fit(X_train, y_train)
y_pred_gb = gb.predict(X_test)

# ================================================================================

# BAGIAN 5: EVALUASI MODEL
```

```
#
================================================================================
========

print("\n\n[5] EVALUASI MODEL")

print("="*80)


def evaluate_model(y_true, y_pred, model_name):

    """Fungsi untuk evaluasi model"""

    print(f"\n{model_name}")

    print("-"*80)


    acc = accuracy_score(y_true, y_pred)

    prec = precision_score(y_true, y_pred)

    rec = recall_score(y_true, y_pred)

    f1 = f1_score(y_true, y_pred)


    print(f"Accuracy:  {acc:.4f}")

    print(f"Precision: {prec:.4f}")

    print(f"Recall:    {rec:.4f}")

    print(f"F1-Score:  {f1:.4f}")


    print(f"\nClassification Report:")

    print(classification_report(y_true, y_pred, target_names=['Not Survived', 'Survived']))


    return acc, prec, rec, f1


# Evaluasi semua model

metrics = {}
```

```
metrics['DT Default'] = evaluate_model(y_test, y_pred_default, "Decision Tree (Default)")
metrics['DT Tuned'] = evaluate_model(y_test, y_pred_best, "Decision Tree (Tuned)")
metrics['Random Forest'] = evaluate_model(y_test, y_pred_rf, "Random Forest")
metrics['Gradient Boosting'] = evaluate_model(y_test, y_pred_gb, "Gradient Boosting")

# Confusion Matrix untuk model terbaik
print("\n\nConfusion Matrix - Decision Tree (Tuned)")
print("-"*80)
cm = confusion_matrix(y_test, y_pred_best)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Not Survived', 'Survived'],
            yticklabels=['Not Survived', 'Survived'])
plt.title('Confusion Matrix - Decision Tree (Tuned)')
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.savefig('confusion_matrix.png', dpi=300, bbox_inches='tight')
plt.show()

# Perbandingan model
print("\n\nPerbandingan Semua Model")
print("="*80)
comparison_df = pd.DataFrame(metrics,
                             index=['Accuracy', 'Precision', 'Recall', 'F1-Score']).T
print(comparison_df)

# Visualisasi perbandingan
```

```
comparison_df.plot(kind='bar', figsize=(12, 6))
plt.title('Perbandingan Performa Model')
plt.xlabel('Model')
plt.ylabel('Score')
plt.ylim(0, 1)
plt.legend(loc='lower right')
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig('model_comparison.png', dpi=300, bbox_inches='tight')
plt.show()

#
================================================================================
========
# BAGIAN 6: VISUALISASI DECISION TREE
#
================================================================================
========
print("\n\n[6] VISUALISASI DECISION TREE")
print("-"*80)

plt.figure(figsize=(20, 10))
plot_tree(dt_best,
          feature_names=X.columns,
          class_names=['Not Survived', 'Survived'],
          filled=True,
          rounded=True,
          fontsize=10)
```

```
plt.title('Decision Tree Visualization (Tuned Model)', fontsize=16)
plt.savefig('decision_tree_visualization.png', dpi=300, bbox_inches='tight')
plt.show()

print("Decision tree berhasil divisualisasikan dan disimpan!")

# Feature Importance
print("\n\nFeature Importance - Decision Tree (Tuned)")
print("-"*80)
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': dt_best.feature_importances_
}).sort_values('Importance', ascending=False)

print(feature_importance)

plt.figure(figsize=(10, 6))
plt.barh(feature_importance['Feature'], feature_importance['Importance'])
plt.xlabel('Importance')
plt.title('Feature Importance - Decision Tree')
plt.gca().invert_yaxis()
plt.tight_layout()
plt.savefig('feature_importance.png', dpi=300, bbox_inches='tight')
plt.show()

#
================================================================================
========
```

```
# BAGIAN 7: KESIMPULAN
#
================================================================================
========

print("\n\n" + "="*80)
print("KESIMPULAN")
print("="*80)

best_model_name = comparison_df['Accuracy'].idxmax()
best_accuracy = comparison_df['Accuracy'].max()

print(f"\n1. Model terbaik: {best_model_name}")
print(f"   - Accuracy: {best_accuracy:.4f}")

print(f"\n2. Parameter terbaik Decision Tree:")
for param, value in grid_search.best_params_.items():
    print(f"   - {param}: {value}")

print(f"\n3. Fitur paling penting:")
for idx, row in feature_importance.head(3).iterrows():
    print(f"   - {row['Feature']}: {row['Importance']:.4f}")

print("\n4. Kelebihan tree-based methods pada kasus ini:")
print("   - Mudah diinterpretasi dan divisualisasikan")
print("   - Dapat menangani fitur kategorikal dan numerikal")
print("   - Tidak memerlukan feature scaling")
print("   - Dapat menangkap non-linear relationships")
print("   - Feature importance membantu memahami faktor survival")
```

```
print("\n5. Hasil akhir:")
print(f"   Decision Tree berhasil memprediksi survival dengan accuracy {best_accuracy:.2%}")
print(f"   Fitur 'sex', 'pclass', dan 'age' merupakan faktor paling berpengaruh")

print("\n" + "="*80)
print("ANALISIS SELESAI")
print("="*80)
```

LINK REPOSITORI

https://github.com/NurulAzizahLonek26/Machine-Learning-uas