UAS MACHINE LEARNING

Disusun Oleh

Muhammad Sabiilul Hikam Azzuhrie – 231011400198

Kelas 05TPLE004

Program Studi Teknik Informatika

Fakultas Ilmu Komputer Universitas Pamulang

Jl. Raya Puspiptek No. 11, Serpong, Kota Tangerang Selatan, Banten 15316

Pendahuluan

Machine learning adalah pendekatan dalam kecerdasan buatan yang memungkinkan komputer membuat prediksi atau keputusan berdasarkan pola dari data. Dalam tugas ini, algoritma yang digunakan adalah Decision Tree karena logikanya mudah dipahami: model menentukan keputusan lewat serangkaian aturan berbentuk "jika–maka".

Studi kasus yang dipakai adalah dataset Titanic, dengan tujuan memprediksi status selamat atau tidak selamat untuk setiap penumpang. Dataset ini cocok untuk latihan karena berisi kombinasi fitur numerik dan kategorikal (misalnya Age, Fare, Sex, Embarked) serta memiliki beberapa nilai kosong yang perlu ditangani pada tahap preprocessing. Dengan begitu, proses pengerjaan mencakup alur machine learning yang lengkap: eksplorasi data, pembersihan data, pemodelan, evaluasi, dan visualisasi pohon.

Teori Singkat

1. Pengertian Decision Tree

Decision Tree adalah model supervised learning yang membentuk struktur pohon keputusan untuk melakukan klasifikasi atau regresi. Data dipisahkan bertahap berdasarkan fitur yang paling "membantu" membedakan kelas, hingga mencapai keputusan akhir di bagian daun (leaf).

2. Istilah penting pada Decision Tree

- **Node**: titik keputusan yang memuat aturan pemisahan (contoh: Age <= 29.5).
- **Root**: node awal, tempat seluruh data mulai diproses.
- **Leaf**: node akhir yang memberikan prediksi (kelas 0/1 pada kasus ini).
- **Splitting**: proses memilih fitur dan batas nilai untuk memecah data menjadi kelompok yang lebih homogen.
- **Pruning**: cara mengurangi kompleksitas pohon agar tidak terlalu "menghafal" data latih (mengurangi overfitting). Pruning bisa dilakukan dengan membatasi kedalaman pohon atau mengatur minimal sampel pada node/leaf.

3. Perbedaan antara Decision Tree, Random Forest dan Gradient Boosting

- **Decision Tree**: satu pohon, cepat dan mudah diinterpretasi, tapi cenderung mudah overfitting.
- **Random Forest**: kumpulan banyak pohon yang dilatih pada sampel data berbeda; hasil prediksi digabungkan, biasanya lebih stabil dan lebih akurat.
- **Gradient Boosting**: membangun model secara bertahap; model berikutnya fokus memperbaiki kesalahan model sebelumnya. Sering sangat kuat, namun butuh tuning parameter yang lebih hati-hati.

4. Kelebihan dan Kekurangan Tree-Based Methods

Metode berbasis pohon (tree-based methods) memiliki sejumlah kelebihan yang membuatnya populer dalam berbagai tugas klasifikasi maupun regresi. Salah satu keunggulan utamanya adalah kemudahan interpretasi, karena proses pengambilan keputusan dapat dijelaskan melalui aturan-aturan yang jelas (misalnya bentuk "jika–maka"), sehingga hasil model relatif mudah dipahami. Selain itu,

model berbasis tree juga mampu menangkap hubungan yang tidak linear serta interaksi antar fitur tanpa perlu asumsi bentuk hubungan tertentu seperti pada model linear. Keunggulan lainnya, metode ini umumnya tidak memerlukan scaling atau normalisasi fitur, karena pemisahan data dilakukan berdasarkan aturan threshold/partition, bukan jarak antar titik data.

Meskipun demikian, metode berbasis tree juga memiliki keterbatasan. Pada decision tree tunggal, risiko overfitting cukup tinggi, terutama bila pohon dibiarkan tumbuh terlalu dalam sehingga model cenderung "menghafal" data latih dan kurang baik dalam generalisasi pada data baru. Selain itu, model tree bersifat sensitif terhadap perubahan kecil pada data; sedikit perubahan pada sampel dapat menghasilkan struktur pohon yang berbeda, sehingga kestabilannya lebih rendah dibanding beberapa metode lain. Pada dataset yang lebih kompleks, performa decision tree tunggal sering kali kurang kompetitif dibanding metode ensemble seperti Random Forest atau Gradient Boosting yang menggabungkan banyak pohon untuk meningkatkan stabilitas dan akurasi.

Metodologi

Dataset dan target

Dataset yang digunakan adalah **Titanic**, dengan target **Survived**:

- 0: tidak selamat
- 1: selamat

Dengan fitur yang digunakan pada pemodelan yakni Pclass, Sex, Age, SibSp, Parch, Fare, Embarked

Alur pengerjaan

1. Memuat dataset dan melakukan eksplorasi singkat (ukuran data, statistik sederhana, missing value).
2. Preprocessing: menangani nilai kosong dan mengubah fitur kategorikal menjadi numerik (encoding).
3. Membagi data menjadi training set dan testing set.
4. Melatih model Decision Tree dan mencoba beberapa konfigurasi parameter.
5. Evaluasi model menggunakan metrik klasifikasi.
6. Visualisasi pohon keputusan untuk melihat aturan yang terbentuk.

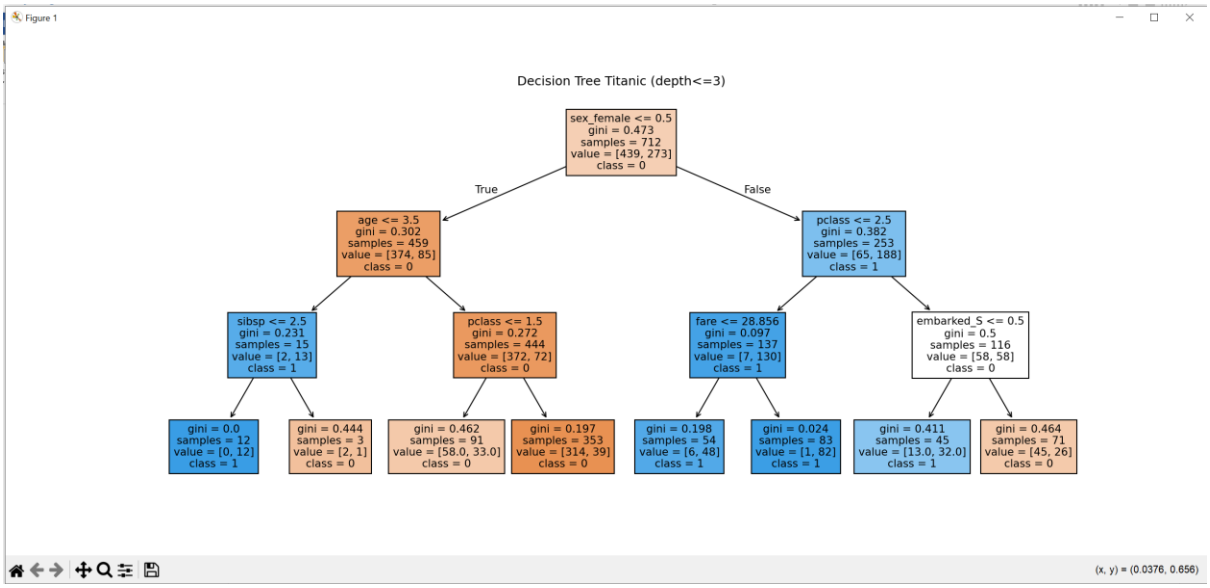
Implementasi dengan Python

```python
# ================ UAS Machine Learning ================
# Nama : M Sabiilul Hikam Azzuhrie
# NIM  : 231011400198
# Kelas : 05TPLE004
# ======================================================
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report

# Load data
df = pd.read_csv("titanic_seaborn.csv")

# Fitur & target
X = df[["pclass","sex","age","sibsp","parch","fare","embarked"]]
y = df["survived"]
```

```python
# Preprocess: missing value + encoding
num = ["pclass","age","sibsp","parch","fare"]
cat = ["sex","embarked"]
pre = ColumnTransformer([
    ("num", SimpleImputer(strategy="median"), num),
    ("cat", Pipeline([("imp", SimpleImputer(strategy="most_frequent")),
                      ("oh", OneHotEncoder(handle_unknown="ignore"))]), cat)
])

# Model
model = Pipeline([("pre", pre),
                  ("dt", DecisionTreeClassifier(criterion="gini", max_depth=3, random_state=42))])

# Split + train + evaluate
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
model.fit(X_train, y_train)
pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, pred))
print(classification_report(y_test, pred))

# Visualisasi tree (dibatasi biar kebaca)
dt = model.named_steps["dt"]
ohe = model.named_steps["pre"].named_transformers_["cat"].named_steps["oh"]
feat_names = num + list(ohe.get_feature_names_out(cat))

plt.figure(figsize=(16, 7))
plot_tree(dt, feature_names=feat_names, class_names=["0","1"], filled=True, max_depth=3)
plt.title("Decision Tree Titanic (depth<=3)")
plt.show()
```

Hasil dan Analisis

```
Sabil@Vivobook MINGW64 /d/uas_machine_learning_2026
$ py titanic_seaborn.py
Accuracy: 0.7932960893854749
              precision    recall  f1-score   support

           0       0.77      0.95      0.85       110
           1       0.86      0.55      0.67        69

    accuracy                           0.79       179
   macro avg       0.82      0.75      0.76       179
weighted avg       0.81      0.79      0.78       179
```



Analisis singkat

- Model lebih kuat mengenali kelas 0 (recall 0.95) dibanding kelas 1 (recall 0.55). Artinya, model masih sering melewatkan penumpang yang sebenarnya selamat (false negative kelas 1).
- Dari visualisasi tree, split awal adalah sex_female, jadi jenis kelamin adalah fitur paling dominan. Setelah itu model mempertimbangkan age, pclass, fare, dan embarked pada cabang-cabang berikutnya.
- Secara umum, membatasi kompleksitas (mis. max_depth=3) membantu mengurangi overfitting, tetapi bisa membuat model kurang menangkap pola selamat (kelas 1).

Kesimpulan

Model Decision Tree pada dataset Titanic (dengan imputasi missing value dan encoding kategorikal) menghasilkan akurasi sekitar 79,33% dengan criterion='gini' dan max_depth=3, namun performanya lebih baik untuk kelas tidak selamat (0) dibanding kelas selamat (1) karena recall kelas 1 masih rendah. Visualisasi pohon menunjukkan jenis kelamin sebagai faktor paling dominan, disusul

usia, kelas tiket, tarif, dan pelabuhan, sehingga model cukup interpretatif tetapi masih bisa ditingkatkan lewat tuning atau metode ensemble.

Link Repository GitHub

https://github.com/MOLAdev20/uas_machine_learning_2026

Penutup

Seluruh source code dan laporan ini disusun sebagai pemenuhan tugas UAS Machine Learning.