DW&M - Lab Task 8

Name - Debangan Ghosh

Roll - AP22110010984

AIM -

Write a Python Program to implement Decision tree based classification by downloading some benchmark dataset.

[Consider 70% of the training data and 30% as test data.]

📝 Topics Used

- Dataset Selection: **Titanic Dataset** (Kaggle)
- Data Preprocessing: Handling **missing** values, encoding categorical variables
- Feature Engineering: Selecting key **features** for training
- Train-Test Split: 70% training, 30% testing
- Decision Tree Classifier: Model training and hyperparameter tuning
- Model Evaluation: Accuracy, classification report, and feature importance visualization
- Performance Optimization: Using **GridSearchCV** to find the best max_depth

---

🔄 Step-by-Step Implementation

1️⃣ Data Loading & Exploration

1. Imported the dataset from a GitHub source.
2. Explored data types, missing values, and overall structure.

2️⃣ Data Preprocessing

1. Handled missing values:

❖ Filled Age with the median value.

❖ Filled Embarked with the mode (most frequent value).

❖ Dropped the Cabin column (too many missing values).

2. Encoded categorical variables like Sex and Embarked using Label Encoding.
3. Selected relevant features (Pclass, Sex, Age, SibSp, Parch, Fare, Embarked).

3️⃣ Train-Test Split

Split the dataset into 70% training and 30% testing.

4️⃣ Decision Tree Model Training

1. Used GridSearchCV to find the optimal max_depth.
2. Trained the Decision Tree Classifier with the best depth.

5️⃣ Model Evaluation

1. Predicted survival outcomes on test data.
2. Measured accuracy and generated a classification report.
3. Visualized feature importance to understand key factors affecting survival.

Code -

```
# Step 1: Import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

# Step 2: Load the dataset
url =
"https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic
.csv"
df = pd.read_csv(url)

# Step 3: Basic Data Exploration
print(df.info())
print(df.head())

# Step 4: Handling Missing Values
df['Age'].fillna(df['Age'].median(), inplace=True)  # Fill missing ages
with median
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)  # Fill
missing Embarked with mode
df.drop(columns=['Cabin'], inplace=True)  # Drop Cabin (too many missing
values)

# Step 5: Encode Categorical Variables
label_encoder = LabelEncoder()
df['Sex'] = label_encoder.fit_transform(df['Sex'])  # Male: 1, Female: 0
df['Embarked'] = label_encoder.fit_transform(df['Embarked'])  # Convert
Embarked to numeric

# Step 6: Feature Selection (Choosing important columns)
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']
X = df[features]  # Independent variables
```

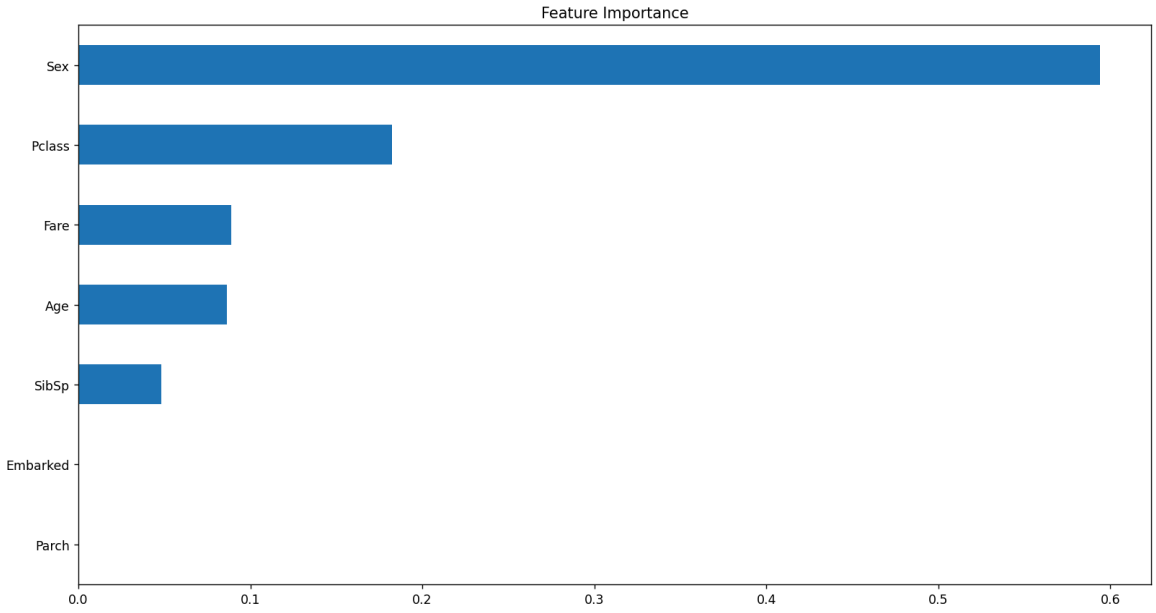
```
y = df['Survived']  # Target variable

# Step 7: Train-Test Split (70% Train, 30% Test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Step 8: Optimize max_depth using GridSearchCV
param_grid = {'max_depth': range(1, 15)}  # Testing max_depth from 1 to 14
grid_search = GridSearchCV(DecisionTreeClassifier(criterion="gini",
random_state=42),
                           param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

# Best max_depth
best_max_depth = grid_search.best_params_['max_depth']
print(f"Optimal max_depth: {best_max_depth}")

# Step 9: Train Decision Tree with optimized max_depth
dt_classifier = DecisionTreeClassifier(criterion="gini",
max_depth=best_max_depth, random_state=42)
dt_classifier.fit(X_train, y_train)

# Step 10: Model Prediction
y_pred = dt_classifier.predict(X_test)

# Step 11: Model Evaluation
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", classification_report(y_test, y_pred))

# Step 12: Visualizing Feature Importance
feature_importances = pd.Series(dt_classifier.feature_importances_,
index=features)
feature_importances.sort_values().plot(kind='barh', title="Feature
Importance")
plt.show()
```

Output -

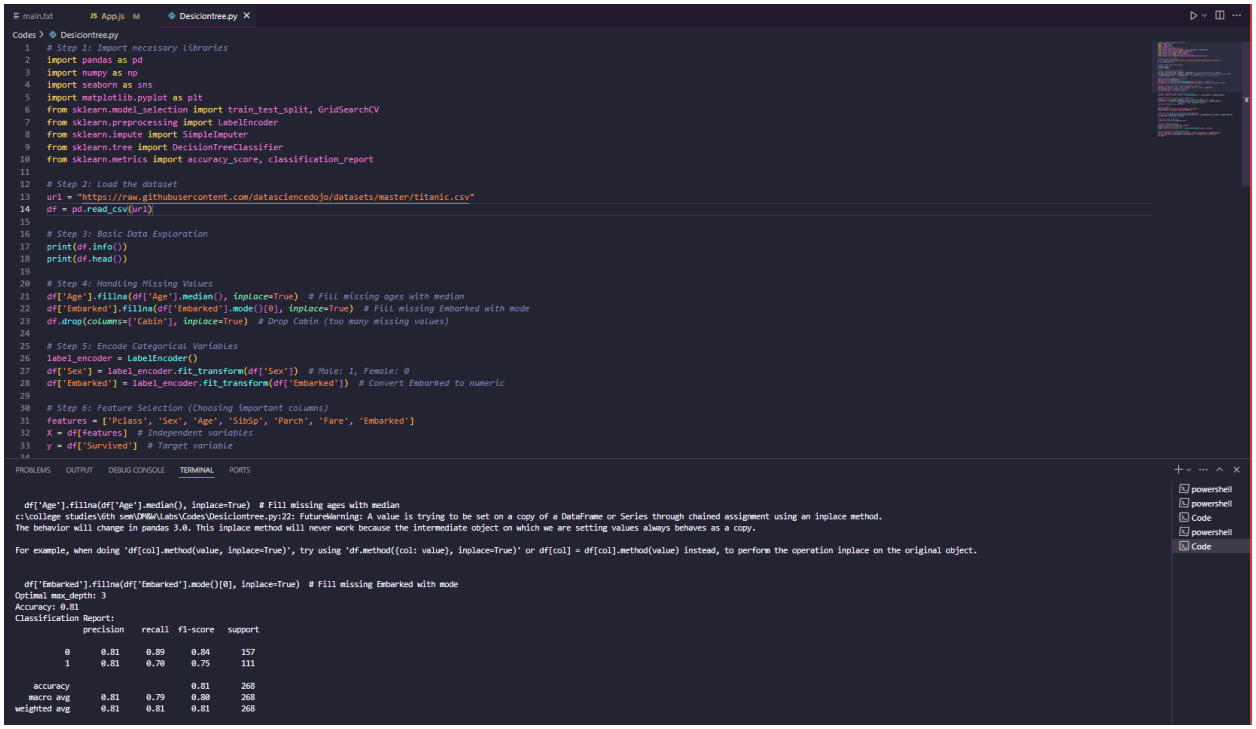
```
Optimal max_depth: 3
Accuracy: 0.81
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.89      0.84       157
           1       0.81      0.70      0.75       111

    accuracy                           0.81       268
   macro avg       0.81      0.79      0.80       268
weighted avg       0.81      0.81      0.81       268
```

Feature Visualization -

(**Feature visualization** is used to understand which features contribute the most to the Decision Tree's predictions.)



Code Image (VSCode) -



main.txt | App.js M | Desicontree.py

Codes > Desicontree.py

```python
# Step 1: Import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

# Step 2: Load the dataset
url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"
df = pd.read_csv(url)

# Step 3: Basic Data Exploration
print(df.info())
print(df.head())

# Step 4: Handling Missing Values
df['Age'].fillna(df['Age'].median(), inplace=True)  # Fill missing ages with median
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)  # Fill missing Embarked with mode
df.drop(columns=['Cabin'], inplace=True)  # Drop Cabin (too many missing values)

# Step 5: Encode Categorical Variables
label_encoder = LabelEncoder()
df['Sex'] = label_encoder.fit_transform(df['Sex'])  # Male: 1, Female: 0
df['Embarked'] = label_encoder.fit_transform(df['Embarked'])  # Convert Embarked to numeric

# Step 6: Feature Selection (Choosing important columns)
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']
X = df[features]  # Independent variables
y = df['Survived']  # Target variable
```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

powershell
powershell
Code
powershell
Code

```
  df['Age'].fillna(df['Age'].median(), inplace=True)  # Fill missing ages with median
c:\college studies\6th sem\DMW\Labs\Codes\Desicontree.py:22: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)  # Fill missing Embarked with mode
Optimal max_depth: 3
Accuracy: 0.81
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.89      0.84       157
           1       0.81      0.70      0.75       111

    accuracy                           0.81       268
   macro avg       0.81      0.79      0.80       268
weighted avg       0.81      0.81      0.81       268
```