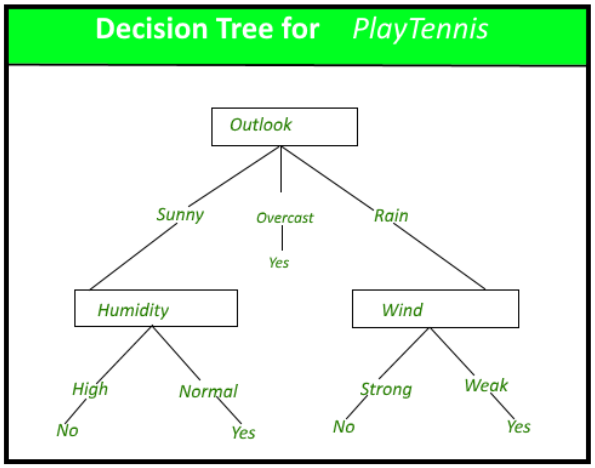
Experiment No. 3
Apply Decision Tree Algorithm on Adult Census Income Dataset and analyze the performance of the model
Date of Performance:
Date of Submission:

Aim: Apply Decision Tree Algorithm on Adult Census Income Dataset and analyze the performance of the model.

Objective: To perform various feature engineering tasks, apply Decision Tree Algorithm on the given dataset and maximize the accuracy, Precision, Recall, F1 score. Improve the performance by performing different data engineering and feature engineering tasks.

Theory:

Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



Dataset:

Predict whether income exceeds $50K/yr based on census data. Also known as "Adult" dataset.

Attribute Information:

Listing of attributes:

>50K, <=50K.

age: continuous.

workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

fnlwgt: continuous.

education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

education-num: continuous.

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

sex: Female, Male.

capital-gain: continuous.

capital-loss: continuous.

hours-per-week: continuous.

native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic,

Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Code:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
from sklearn.preprocessing import LabelEncoder

# Load the dataset
data_url = "adult.csv"
#column_names = ["age", "workclass", "fnlwgt", "education", "education-num",
"marital-status", "occupation", "relationship", "race", "sex", "capital-gain",
"capital-loss", "hours-per-week", "native-country", "income"]
data = pd.read_csv(data_url)

#Handle missing values
data.replace('?', np.nan, inplace=True)
data.dropna(inplace=True)

# Encode categorical variables
# categorical_columns = data.select_dtypes(include=['object']).columns
# data = pd.get_dummies(data, columns=categorical_columns, drop_first=True)
data.drop('native.country', inplace=True, axis=1)
le = LabelEncoder()
data['workclass'] = le.fit_transform(data['workclass'])
data['education'] = le.fit_transform(data['education'])
data['race'] = le.fit_transform(data['race'])
data['occupation'] = le.fit_transform(data['occupation'])
data['marital.status'] = le.fit_transform(data['marital.status'])
data['relationship'] = le.fit_transform(data['relationship'])
data['sex'] = le.fit_transform(data['sex'])
data['income'] = le.fit_transform(data['income'])

print(data.columns)

# Split the data into features (X) and target (y)
```

```
X = data.drop(['race', 'relationship', 'marital.status', 'income'], axis=1)  # Features
y = data["income"]  # Target variable

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a Decision Tree Classifier
dt_classifier = DecisionTreeClassifier(random_state=42)

# Train the model on the training data
dt_classifier.fit(X_train, y_train)

# Make predictions on the test data
y_pred = dt_classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Confusion Matrix:\n{conf_matrix}")
print(f"Classification Report:\n{class_report}")

# Visualize the Decision Tree (optional)
plt.figure(figsize=(15, 10))
plot_tree(dt_classifier, feature_names=list(X.columns), class_names=["<=50K", ">50K"], filled=True, rounded=True, fontsize=10)
plt.show()
```

Output:

Accuracy: 0.7702635504724018

Confusion Matrix:

[[3770 763]

[623 877]]

Conclusion:

1. **Categorical Attributes Handling**: Categorical attributes were transformed into numerical values using Label Encoding, and one categorical column ('native-country') was dropped during preprocessing.
2. **Hyperparameter Tuning**: No hyperparameter tuning was performed in the provided code. However, tuning hyperparameters like 'max_depth,' 'min_samples_split,' and 'min_samples_leaf' can optimize the Decision Tree model's performance.
3. **Model Evaluation Metrics**:
    - **Accuracy**: Measures overall correctness.
    - **Confusion Matrix**: Provides detailed predictions breakdown.
    - **Precision**: Measures positive prediction accuracy.
    - **Recall**: Measures the ability to identify positive instances.
    - **F1 Score**: Balances precision and recall, especially for imbalanced datasets.

Interpret these metrics based on the problem and class distribution to assess model performance accurately.