

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:

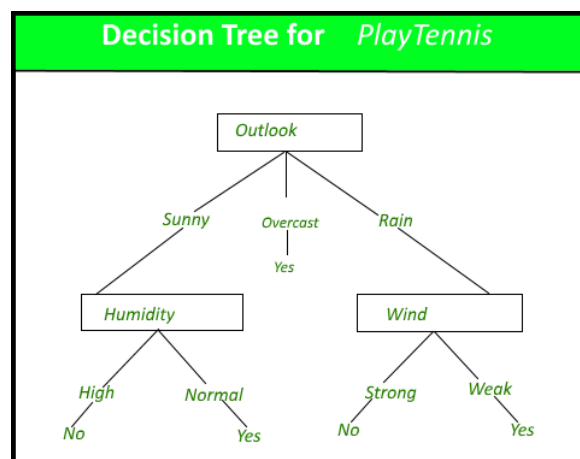


Aim: Apply Decision Tree Algorithm on Adult Census Income Dataset and analyze the performance of the model.

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.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

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, Trinidad & Tobago, Peru, Hong, Holand-Netherlands.

Code:

Conclusion:

1. Discuss about the how categorical attributes have been dealt with during data pre-processing.

Handling categorical & numerical features during data preparation is crucial for creating precise models. There are various preprocessing methods, including:

- a. Label Encoding, which converts qualities like education, occupation, marital status, relationship, and sex into numerical values. Decision tree algorithm can use these attributes since label encoding gives each category a distinct number.
- b. Missing values: For categorical values with missing values, the median imputation method was used, or the mode method for ordinal values.
- c. One Hot Encoder: One Hot Encoder is used to convert categorical values into binary vectors and each category is represented as a binary column for characteristics with nominal categories like native country and occupation.

2. Discuss the hyper-parameter tuning done based on the decision tree obtained.

The Decision Tree model's performance is optimized by the hyper-parameter tweaking, which makes sure that the model is not under- or overfitting the data.

To investigate various combinations of factors, such as the maximum tree depth and splitting criteria, a systematic grid search method was used. Finding the appropriate hyper-parameter to maximize the model's performance is the goal of this method.



3. Comment on the accuracy, confusion matrix, precision, recall and F1 score obtained.

The ratio of results that were accurately predicted to all instances was used to determine the model's accuracy. It gives an overview of the model's performance. The accuracy of this model is about 85%, meaning that 84% of the results were as expected.

Confusion Matrix: The confusion matrix offers a thorough analysis of the performance of the model by displaying true positives, true negatives, false positives, and false negatives. For this model, the confusion matrix is as follows: TP=1031, TN=6564, FP=303, FN=1151.

The harmonic mean of recall and precision is known as the F1 score. It offers a fair evaluation of a model's performance. This model's f1 score falls between recall & precision.

```
import pandas as pd

data = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
test_ids = test["PassengerId"]


def clean(data):
    data = data.drop(["Ticket", "PassengerId", "Name", "Cabin"], axis=1)

    cols = ["SibSp", "Parch", "Fare", "Age"]
    for col in cols:
        data[col].fillna(data[col].median(), inplace=True)

    data.Embarked.fillna("U", inplace=True)
    return data

data = clean(data)
test = clean(test)
```

data.head(10)



	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S
5	0	3	male	28.0	0	0	8.4583	Q
6	0	1	male	54.0	0	0	51.8625	S
7	0	3	male	2.0	3	1	21.0750	S
8	1	3	female	27.0	0	2	11.1333	S
9	1	2	female	14.0	1	0	30.0708	C

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
columns = ["Sex", "Embarked"]

for col in columns:
    data[col] = le.fit_transform(data[col])
    test[col] = le.transform(test[col])
    print(le.classes_)

data.head(5)
```

```
['female' 'male']
['C' 'Q' 'S' 'U']

Survived Pclass Sex Age SibSp Parch Fare Embarked
0 0 3 1 22.0 1 0 7.2500 2
1 1 1 0 38.0 1 0 71.2833 0
2 1 3 0 26.0 0 0 7.9250 2
3 1 1 0 35.0 1 0 53.1000 2

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

y = data["Survived"]
X = data.drop("Survived", axis=1)

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

clf = LogisticRegression(random_state=0, max_iter=1000).fit(X_train, y_train)

predictions = clf.predict(X_val)
from sklearn.metrics import accuracy_score
accuracy_score(y_val, predictions)

0.8100558659217877
```