Practical No: 6

Decision Tree Classifier & Random Forest Classifier

AIM: Write a program to implement the Decision Tree Classifier & Random Forest Classifier with prediction, test score and confusion matrix.

Description:

Decision Tree Classifier:

Interpretability: Decision trees offer easy interpretability, aiding in understanding and explaining the logic behind classification decisions.

Overfitting: Decision trees can be prone to overfitting, especially if deep or complex, necessitating regularization techniques for optimal performance.

Random Forest Classifier:

Ensemble Learning: Random Forest is an ensemble method that combines multiple decision trees, enhancing model accuracy and stability.

Variance Reduction: Random Forest reduces variance by aggregating predictions from different trees, mitigating overfitting and improving generalization to new data.

Code with output

```
import pandas as pd
 import numpy as np
 import matplotlib.pyplot as plt
 import seaborn as sns
 from sklearn.preprocessing import LabelEncoder
 from sklearn.model_selection import train_test_split
 from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
 %matplotlib inline

df = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")

# Keeping emp position unaffected.
 df.head()
```

```
# Exploratory Data Analysis
 sns.countplot(x='Attrition', data=df)

from pandas.core.arrays import categorical

df.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis="columns",
inplace=True)

categorical_col = []

for column in df.columns:
    if df[column].dtype == object:
        categorical_col.append(column)

df['Attrition'] = df['Attrition'].astype("category").cat.codes

for column in categorical_col:
    df[column] = LabelEncoder().fit_transform(df[column])

X = df.drop('Attrition', axis=1)
 y = df['Attrition']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    if train:
        pred = clf.predict(X_train)
        clf_report = pd.DataFrame(classification_report(y_train, pred, output_dict=True))
        print("Train Result:\n===========================================")
        print(f"Accuracy Score: {accuracy_score(y_train, pred) * 100:.2f}%")
        print(" ")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print(" ")
        print(f"Confusion Matrix: \n{confusion_matrix(y_train, pred)}\n")
```

```
    elif not train:
        pred = clf.predict(X_test)
        clf_report = pd.DataFrame(classification_report(y_test, pred, output_dict=True)
        )
        print("Test Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_test, pred) * 100:.2f}%")
        print(" ")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print(" ")
        print(f"Confusion Matrix: \n{confusion_matrix(y_test, pred)}\n")
from sklearn.tree import DecisionTreeClassifier
 from sklearn.ensemble import RandomForestClassifier
 from pickle import TRUE

from sklearn.tree import DecisionTreeClassifier

tree_clf = DecisionTreeClassifier(random_state=42)
 tree_clf.fit(X_train, y_train)

print_score(tree_clf, X_train, y_train, X_test, y_test, train=True)
 print_score(tree_clf, X_train, y_train, X_test, y_test, train=False)

from sklearn.ensemble import RandomForestClassifier

rf_clf = RandomForestClassifier(random_state=42)
 rf_clf.fit(X_train, y_train)

print_score(rf_clf, X_train, y_train, X_test, y_test, train=True)
 print_score(rf_clf, X_train, y_train, X_test, y_test, train=False)
```

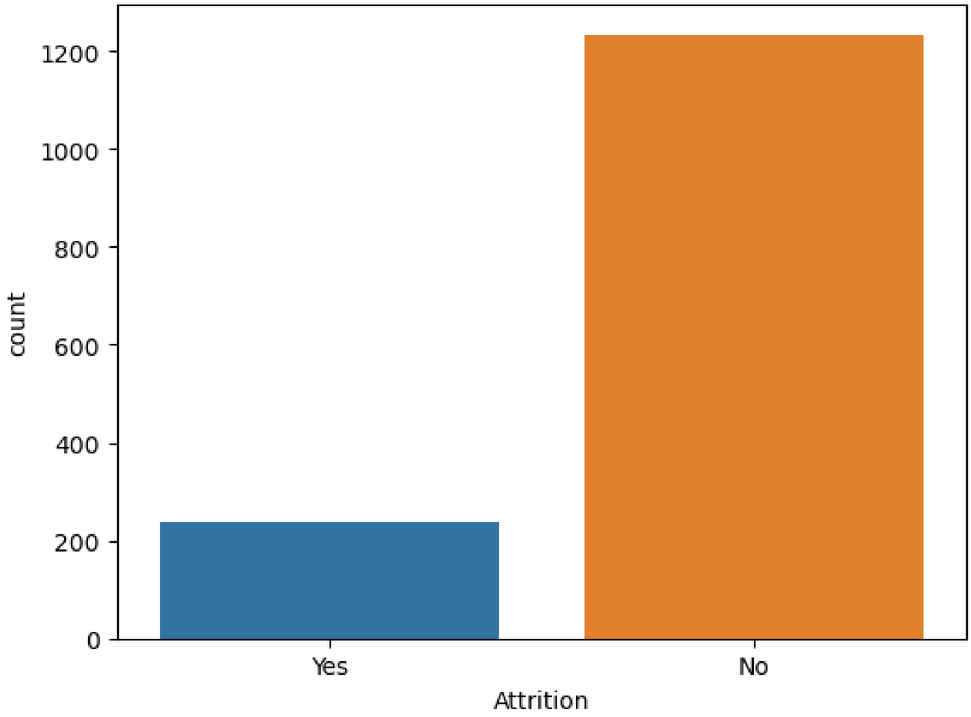
OUTPUT

```
In [3]: df.head()
```

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	Rela
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	

5 rows × 35 columns



```
Train Result:
=======================================
Accuracy Score: 100.00%

CLASSIFICATION REPORT:
               0      1  accuracy  macro avg  weighted avg
precision    1.0    1.0       1.0        1.0           1.0
recall       1.0    1.0       1.0        1.0           1.0
f1-score     1.0    1.0       1.0        1.0           1.0
support    853.0  176.0       1.0     1029.0        1029.0

Confusion Matrix:
[[853   0]
 [  0 176]]

Test Result:
=======================================
Accuracy Score: 77.78%

CLASSIFICATION REPORT:
                    0          1  accuracy   macro avg  weighted avg
precision    0.887363   0.259740  0.777778    0.573551      0.800549
recall       0.850000   0.327869  0.777778    0.588934      0.777778
f1-score     0.868280   0.289855  0.777778    0.579067      0.788271
support    380.000000  61.000000  0.777778  441.000000    441.000000

Confusion Matrix:
[[323  57]
 [ 41  20]]
```

```
Train Result:
=======================================
Accuracy Score: 100.00%

CLASSIFICATION REPORT:
               0      1  accuracy  macro avg  weighted avg
precision    1.0    1.0       1.0        1.0           1.0
recall       1.0    1.0       1.0        1.0           1.0
f1-score     1.0    1.0       1.0        1.0           1.0
support    853.0  176.0       1.0     1029.0        1029.0

Confusion Matrix:
[[853   0]
 [  0 176]]

Test Result:
=======================================
Accuracy Score: 86.17%

CLASSIFICATION REPORT:
                    0          1  accuracy   macro avg  weighted avg
precision    0.871795   0.500000  0.861678    0.685897      0.820367
recall       0.984211   0.098361  0.861678    0.541286      0.861678
f1-score     0.924598   0.164384  0.861678    0.544491      0.819444
support    380.000000  61.000000  0.861678  441.000000    441.000000

Confusion Matrix:
[[374   6]
 [ 55   6]]
```

Confusion Matrix Calculations:

Ninad Karlekar 22306A1012

PAGE No. DATE / /

Random forest confusion matrix

	Attrition	No attrition
Predicted +	374 (TP)	6 (FP)
predicted −	55 (FN)	6 (TN)

1 Accuracy $= \frac{TP + TN}{TP + FP + FN + TN} = \frac{380}{441}$

$= 86.16\ \%$

2 Precision $= \frac{TP}{TP+FP} = \frac{374}{374+6} = 98\ \%$

3 Recall $= \frac{TP}{TP+FN} = \frac{374}{374+55} = 87\%$

Decision tree confusion matrix

$$\begin{bmatrix} 323\ (TP) & 57\ (FP) \\ 41\ (FN) & 20\ (TN) \end{bmatrix}$$

Acc $= \frac{323+20}{323+57+41+20} = \frac{343}{441} = 77\%$

Precision $= \frac{323}{323+57} = \frac{323}{380} = 85\%$

Recall $= \frac{323}{323+41} = \frac{323}{364} = 88\%$

Analysis of Confusion Matrix

The model correctly identified 6 instances as positive.

It correctly identified 374 instances as negative.

However, it made 6 false positive predictions, indicating instances that were predicted as positive but were actually negative.

It also made 55 false negative predictions, indicating instances that were predicted as negative but were actually positive.