|  |  |  |  |
| --- | --- | --- | --- |
| A picture containing drawing, stop, room  Description automatically generated | Machine Learning  Practical # 6 | | |
|  |  |  |  |
|  |  |  |  |
| **Name** | Ninad Karlekar | **Roll Number** | 2230A1012 |
| **A1012Subject/Course:** | Machine Learning | **Class** | M.Sc. IT – Sem III |
| **Topic** | Naive Bayes and Gaussian Classification | **Batch** |  |
|  |  |  |  |
| **Topic: 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 :-**    **Random Forest Classifier :-**    **Code and 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)          **Confusion Matrix Calculations:**  **Learnings:** | | | |
|  | | | |
| **Analysis of Confusion Matrix**  **Learnings:** | | | |