AML Lab Assignment-6:

Name: Gurvinder Kaur Matharu
PRN: 1032230432
Roll No.: PA14

AIM:

Perform ML performance analysis on a given dataset to find Accuracy, Error rate, precision, recall and confusion matrix for supervised learning algorithms

THEORY:

Performing a machine learning (ML) performance analysis involves evaluating models using various metrics to gauge their effectiveness in making predictions. Here are the metrics and their roles in assessing supervised learning algorithms:

Accuracy: It measures the proportion of correctly classified instances out of the total instances. High accuracy indicates a good overall performance but might not be sufficient for imbalanced datasets.

Precision: It quantifies the ratio of correctly predicted positive observations to the total predicted positives. It's essential when the cost of false positives is high.

Recall (Sensitivity): It calculates the ratio of correctly predicted positive observations to the all-actual positives in the dataset. It's crucial when the cost of false negatives is high.

F1 Score: The F1 score is the harmonic mean of precision and recall and provides a balanced evaluation metric, particularly useful in imbalanced datasets.The F1 score ranges between 0 and 1, where a higher score indicates better balance between precision and recall.

Confusion Matrix: This matrix summarizes the performance of a classification algorithm by presenting the counts of true positives, true negatives, false positives, and false negatives.

ROC Curve: Plots the trade-off between sensitivity (true positive rate) and specificity (true negative rate) for different threshold values. It illustrates how well the model distinguishes between classes.

AUC (Area Under the Curve): Measures the entire two-dimensional area under the ROC curve from (0,0) to (1,1). An AUC closer to 1 signifies better model performance, indicating a higher true positive rate and lower false positive rate across various thresholds.

CODE EXECUTION & OUTPUT:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score, roc_curve, auc
from sklearn.preprocessing import LabelEncoder
```

```
data = pd.read_csv('/content/Employee.csv')
```

```
data.head()
```

	Education	JoiningYear	City	PaymentTier	Age	Gender	EverBenched	ExperienceInCurrentDomain	LeaveOrNot
0	Bachelors	2017	Bangalore	3	34	Male	No	0	0
1	Bachelors	2013	Pune	1	28	Female	No	3	1
2	Bachelors	2014	New Delhi	3	38	Female	No	2	0

```
data.columns
```

```
Index(['Education', 'JoiningYear', 'City', 'PaymentTier', 'Age', 'Gender',
       'EverBenched', 'ExperienceInCurrentDomain', 'LeaveOrNot'],
      dtype='object')
```

```
data.describe()
```

	JoiningYear	PaymentTier	Age	ExperienceInCurrentDomain	LeaveOrNot
count	4653.000000	4653.000000	4653.000000	4653.000000	4653.000000
mean	2015.062970	2.698259	29.393295	2.905652	0.343864
std	1.863377	0.561435	4.826087	1.558240	0.475047
min	2012.000000	1.000000	22.000000	0.000000	0.000000
25%	2013.000000	3.000000	26.000000	2.000000	0.000000
50%	2015.000000	3.000000	28.000000	3.000000	0.000000
75%	2017.000000	3.000000	32.000000	4.000000	1.000000
max	2018.000000	3.000000	41.000000	7.000000	1.000000

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4653 entries, 0 to 4652
Data columns (total 9 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   Education                  4653 non-null   object
 1   JoiningYear                4653 non-null   int64 
 2   City                       4653 non-null   object
 3   PaymentTier                4653 non-null   int64 
 4   Age                        4653 non-null   int64 
 5   Gender                     4653 non-null   object
 6   EverBenched                4653 non-null   object
 7   ExperienceInCurrentDomain  4653 non-null   int64 
 8   LeaveOrNot                 4653 non-null   int64 
dtypes: int64(5), object(4)
memory usage: 327.3+ KB
```

```
data.shape
```

```
(4653, 9)
```

```
data.isnull().sum()
```

```
Education                    0
JoiningYear                  0
City                         0
PaymentTier                  0
Age                          0
Gender                       0
EverBenched                  0
ExperienceInCurrentDomain    0
LeaveOrNot                   0
dtype: int64
```

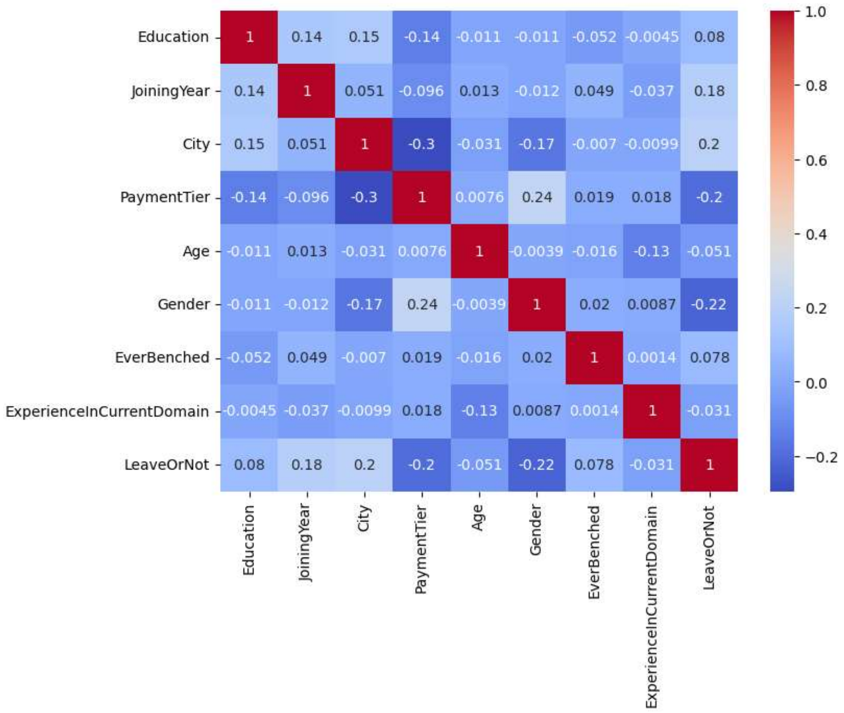
```
encoder = LabelEncoder()
```

```
data['City'] = encoder.fit_transform(data['City'])
data['Education'] = encoder.fit_transform(data['Education'])
data['Gender'] = encoder.fit_transform(data['Gender'])
data['EverBenched'] = encoder.fit_transform(data['EverBenched'])
```

```
data.head()
```

	Education	JoiningYear	City	PaymentTier	Age	Gender	EverBenched	Exp
0	0	2017	0	3	34	1	0	
1	0	2013	2	1	28	0	0	
2	0	2014	1	3	38	0	0	
3	1	2016	0	3	27	1	0	
4	1	2017	2	3	24	1	1	

```
corr = data.corr()
plt.figure(figsize=(8,6))
sns.heatmap(corr, cmap = 'coolwarm', annot=True)
plt.show()
```




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

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

```
MLA = [
    #KNN
    KNeighborsClassifier(n_neighbors=6, metric='euclidean'),
    #SVM
    SVC(kernel='linear'),
    #Trees
    DecisionTreeClassifier(),
    ]
```

```
MLA_columns = []
MLA_compare = pd.DataFrame(columns = MLA_columns)
```

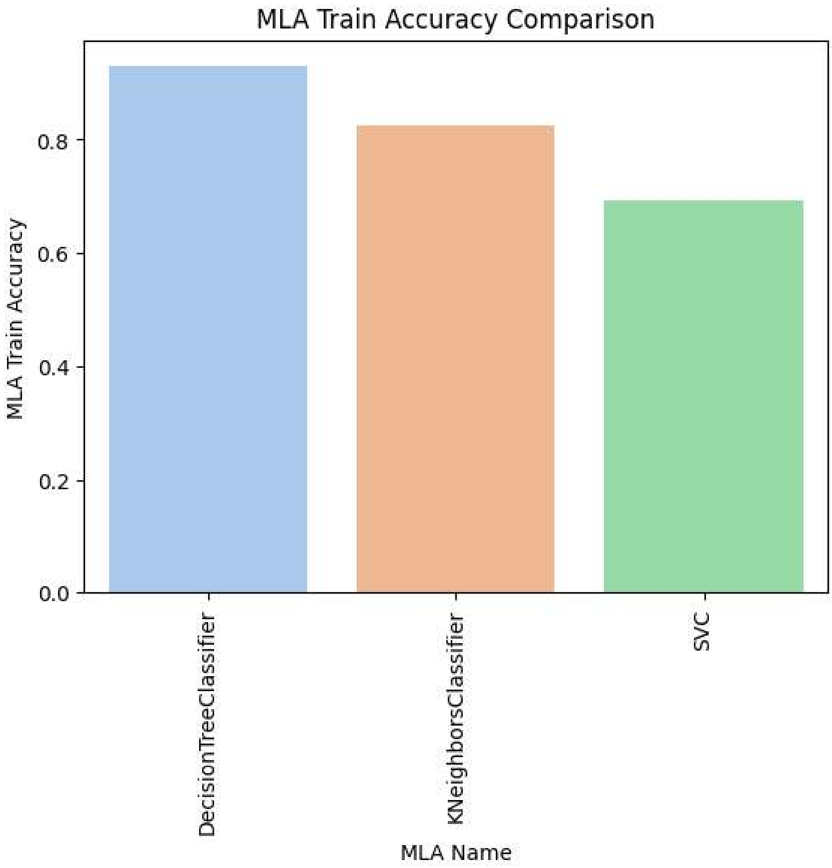
```
row_index = 0
for alg in MLA:
    predicted = alg.fit(X_train, y_train).predict(X_test)
    print(alg)
    print(classification_report(y_test,predicted))
    fp, tp, th = roc_curve(y_test, predicted)
    MLA_name = alg.__class__.__name__
    MLA_compare.loc[row_index,'MLA Name'] = MLA_name
    MLA_compare.loc[row_index, 'MLA Train Accuracy'] = round(alg.score(X_train, y_train), 4)
    MLA_compare.loc[row_index, 'MLA Test Accuracy'] = round(alg.score(X_test, y_test), 4)
    MLA_compare.loc[row_index, 'MLA AUC'] = auc(fp, tp)
    row_index+=1
```

```
KNeighborsClassifier(metric='euclidean', n_neighbors=6)
              precision    recall  f1-score   support

           0       0.80      0.96      0.87       775
           1       0.87      0.52      0.65       389

    accuracy                           0.81      1164
   macro avg       0.84      0.74      0.76      1164
weighted avg       0.82      0.81      0.80      1164

SVC(kernel='linear')
              precision    recall  f1-score   support

           0       0.74      0.89      0.81       775
           1       0.63      0.37      0.46       389

    accuracy                           0.72      1164
   macro avg       0.69      0.63      0.64      1164
weighted avg       0.70      0.72      0.69      1164

DecisionTreeClassifier()
              precision    recall  f1-score   support

           0       0.87      0.89      0.88       775
           1       0.77      0.72      0.75       389

    accuracy                           0.84      1164
   macro avg       0.82      0.81      0.81      1164
weighted avg       0.83      0.84      0.83      1164
```

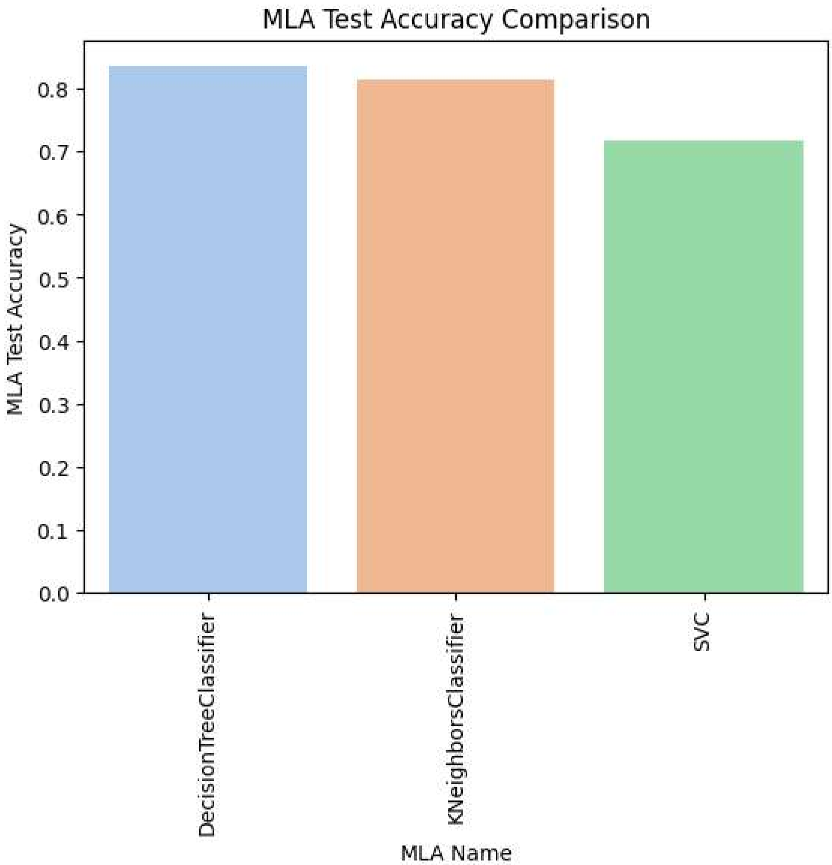
```
MLA_compare.sort_values(by = ['MLA Test Accuracy'], ascending = False, inplace = True)
MLA_compare
```

	MLA Name	MLA Train Accuracy	MLA Test Accuracy	MLA AUC
2	DecisionTreeClassifier	0.9301	0.8351	0.807629
0	KNeighborsClassifier	0.8240	0.8136	0.739645
1	SVC	0.6916	0.7174	0.629616

```
plt.figsize=(6,4)
sns.barplot(x="MLA Name", y="MLA Train Accuracy",data=MLA_compare,palette='pastel')
plt.xticks(rotation=90)
plt.title('MLA Train Accuracy Comparison')
plt.show()
```

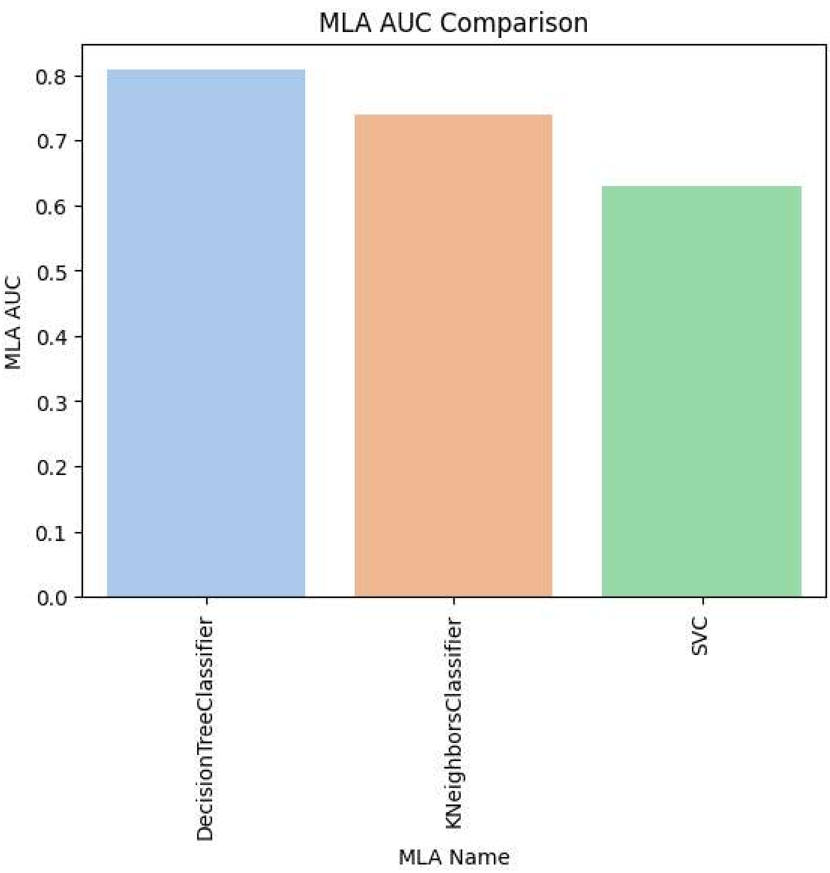



```
plt.figsize=(6,4)
sns.barplot(x="MLA Name", y="MLA Test Accuracy",data=MLA_compare,palette='pastel')
plt.xticks(rotation=90)
plt.title('MLA Test Accuracy Comparison')
plt.show()
```

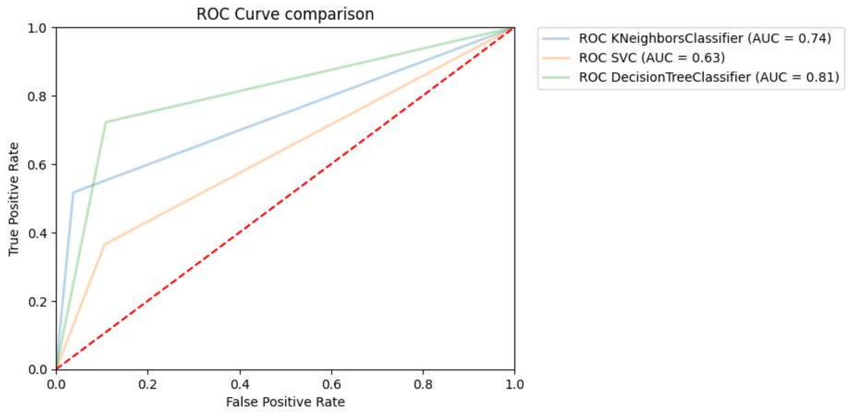



```
plt.figsize=(6,4)
sns.barplot(x="MLA Name", y="MLA AUC",data=MLA_compare,palette='pastel')
```

```
plt.xticks(rotation=90)
plt.title('MLA AUC Comparison')
plt.show()
```




```
index = 1
for alg in MLA:
    predicted = alg.fit(X_train, y_train).predict(X_test)
    fp, tp, th = roc_curve(y_test, predicted)
    roc_auc_mla = auc(fp, tp)
    MLA_name = alg.__class__.__name__
    plt.plot(fp, tp, lw=2, alpha=0.3, label='ROC %s (AUC = %0.2f)'  % (MLA_name, roc_auc_mla))
    index+=1
plt.title('ROC Curve comparison')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```


ROC Curve comparison
ROC KNeighborsClassifier (AUC = 0.74)
ROC SVC (AUC = 0.63)
ROC DecisionTreeClassifier (AUC = 0.81)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0