MACHINE LEARNING LAB

EXERCISE 5

Aim :

Use the teleco-customer-churn dataset for the following:

1. Perform the necessary pre-processings.
2. Apply all the classification algorithms (KNN, Logistic Regression, Naive Bayes, Decision Trees, SVM) on this dataset and print the accuracies.
3. Find which algorithm gave the best accuracy.
4. Provide a justification as to why that algorithm provided the best accuracy

Algorithm :

1. We load and preprocess the data by removing the unnecessary features, hot encoding.

1. We then split the data into X and Y and then scale the data

1. One by one, we apply the classification algorithms, KNN, Logistic Regression, Naive Bayes, Decision Trees and SVM on the dataset

1. We also print the performance metrics using each of the algorithms.

1. We then try to analyse why a particular algorithm would have given the highest accuracy and justify.

Code and Output :

In [72]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [73]:
```
df=pd.read_csv(r"C:\Users\TEJU\Downloads\Telco-Customer-Churn.csv")
```

In [74]:
```
df.head()
```

Out[74]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Int
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Int
2	3668-QPYBK	Male	0	No	No	2	Yes	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	
4	9237-HQITU	Female	0	No	No	2	Yes	No	

5 rows × 21 columns

In [75]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

I notice that a particular numerical column TotalCharges is given as an object type instead of int/float. So we change that first

In [76]:
```
df['TotalCharges'].replace(" ",0,inplace=True)
df['TotalCharges']=df['TotalCharges'].astype('float64')
```

We drop the customerID column as it is not required

In [77]:
```
df=df.drop('customerID',axis=1)
```

We give meaning to the 0's and 1's in the senior citizen column by mapping to No's and Yes's

In [78]:
```
df['SeniorCitizen']=df['SeniorCitizen'].map({0:'No',1:'Yes'})
```

We separate the columns having numbers and columns having words separately

In [79]:
```
num_features=df.select_dtypes(include='number')
cat_features=df.select_dtypes(exclude='number')
```

Let us now perform hot encoding on the cat features and then calculate correlation matrix

In [80]:
```
cat_features_encoded = pd.get_dummies(data=cat_features, dtype=int)
churn_corr = cat_features_encoded.corr()['Churn_Yes'].drop(['Churn_Yes', 'Churn_No']
```

In [81]:
```
df_final=pd.get_dummies(data=df,drop_first=True,dtype=int)
```

We drop the features which are unrelated to the target variable churn

In [82]:
```
df_final = df_final.drop(['gender_Male', 'PhoneService_Yes',
                         'MultipleLines_No phone service',
                         'MultipleLines_Yes'], axis=1)
```

In [83]:
```
df_final.head()
```

Out[83]:

	tenure	MonthlyCharges	TotalCharges	SeniorCitizen_Yes	Partner_Yes	Dependents_Yes	InternetSe
0	1	29.85	29.85	0	1	0	
1	34	56.95	1889.50	0	0	0	
2	2	53.85	108.15	0	0	0	
3	45	42.30	1840.75	0	0	0	
4	2	70.70	151.65	0	0	0	

5 rows × 27 columns

In [84]:
```
df_final['Churn_Yes'].value_counts()
```

Out[84]:
```
0    5174
1    1869
Name: Churn_Yes, dtype: int64
```

We notice that there is an imbalance of 0's and 1's because of which the model will tend to give 0 as the answer because of its high number, we can use some techniques to adjust that by adding random values to increase number of 1's like SMOTE but refraining from doing the same for this lab

Let us now split into X and Y to start creating our ML model

In [85]:
```
X=df_final.drop('Churn_Yes',axis=1)
y=df_final['Churn_Yes']
```

In [87]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_stat
```

Let us scale the data now

In [88]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

KNN Classifier

In [89]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [107…
```
knn_model = KNeighborsClassifier(n_jobs=-1,n_neighbors=5)
knn_model.fit(X_train,y_train)
```

Out[107…
```
KNeighborsClassifier(n_jobs=-1)
```

In [108…
```
y_pred= knn_model.predict(X_test)
```

In [109…
```
from sklearn.metrics import confusion_matrix, accuracy_score,classification_report
```

In [110…
```
confusion_matrix(y_test,y_pred)
```

Out[110…
```
array([[872, 161],
       [184, 192]], dtype=int64)
```

In [111…
```
accuracy_score(y_test,y_pred)
```

Out[111…
```
0.7551454932576295
```

In [162…
```
class_report=classification_report(y_test,y_pred)
print(class_report)
```

```
              precision    recall  f1-score   support

           0       0.83      0.88      0.85      1033
           1       0.60      0.52      0.56       376

    accuracy                           0.78      1409
   macro avg       0.72      0.70      0.71      1409
weighted avg       0.77      0.78      0.78      1409
```

=> KNN Classifier - Accuracy of 75.5%

Logistic Regression

In [113…
```
from sklearn.linear_model import LogisticRegression
reg = LogisticRegression()
reg.fit(X_train,y_train)
```
Out[113…
```
LogisticRegression()
```
In [114…
```
y_pred = reg.predict(X_test)
```
In [115…
```
confusion_matrix(y_test,y_pred)
```
Out[115…
```
array([[917, 116],
       [171, 205]], dtype=int64)
```
In [116…
```
accuracy_score(y_test,y_pred)
```
Out[116…
```
0.7963094393186657
```
In [117…
```
class_report=classification_report(y_test,y_pred)
print(class_report)
```
```
              precision    recall  f1-score   support

           0       0.84      0.89      0.86      1033
           1       0.64      0.55      0.59       376

    accuracy                           0.80      1409
   macro avg       0.74      0.72      0.73      1409
weighted avg       0.79      0.80      0.79      1409
```

=> Logistic Regression - Accuracy of 79.6%

Naive Bayes

In [118…
```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train,y_train)
```
Out[118…
```
GaussianNB()
```
In [119…
```
y_pred = classifier.predict(X_test)
```
In [120…
```
confusion_matrix(y_test,y_pred)
```
Out[120…
```
array([[597, 436],
       [ 52, 324]], dtype=int64)
```
In [121…
```
accuracy_score(y_test,y_pred)
```
Out[121…
```
0.6536550745209369
```

In [123…

```
class_report=classification_report(y_test,y_pred)
print(class_report)
```

```
              precision    recall  f1-score   support

           0       0.92      0.58      0.71      1033
           1       0.43      0.86      0.57       376

    accuracy                           0.65      1409
   macro avg       0.67      0.72      0.64      1409
weighted avg       0.79      0.65      0.67      1409
```

=> Naive Bayes - Accuracy of 65.3%

Decision Trees

In [124…

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(criterion='entropy')
tree.fit(X_train,y_train)
```

Out[124…

```
DecisionTreeClassifier(criterion='entropy')
```

In [125…

```
y_pred = tree.predict(X_test)
```

In [126…

```
confusion_matrix(y_test,y_pred)
```

Out[126…

```
array([[859, 174],
       [190, 186]], dtype=int64)
```

In [127…

```
accuracy_score(y_test,y_pred)
```

Out[127…

```
0.7416607523066004
```

In [129…

```
class_report=classification_report(y_test,y_pred)
print(class_report)
```

```
              precision    recall  f1-score   support

           0       0.82      0.83      0.83      1033
           1       0.52      0.49      0.51       376

    accuracy                           0.74      1409
   macro avg       0.67      0.66      0.67      1409
weighted avg       0.74      0.74      0.74      1409
```

In [130…

```
tree2=DecisionTreeClassifier()
tree2.fit(X_train,y_train)
```

Out[130…

```
DecisionTreeClassifier()
```

In [131…

```
y_pred=tree2.predict(X_test)
```

In [132…
```
confusion_matrix(y_test,y_pred)
```

Out[132…
```
array([[840, 193],
       [188, 188]], dtype=int64)
```

In [133…
```
accuracy_score(y_test,y_pred)
```

Out[133…
```
0.7295954577714692
```

In [134…
```
class_report = classification_report(y_test, y_pred)
print(class_report)
```

```
              precision    recall  f1-score   support

           0       0.82      0.81      0.82      1033
           1       0.49      0.50      0.50       376

    accuracy                           0.73      1409
   macro avg       0.66      0.66      0.66      1409
weighted avg       0.73      0.73      0.73      1409
```

I tried with both criterions of decision tree - Gini Impurity and entropy.

Entropy gave an accuracy higher than Gini

=> Decision Tree - Accuracy of 74.1%

Support Vector Machines

In [156…
```
from sklearn.svm import SVC
clf = SVC(kernel='linear')
```

In [157…
```
clf.fit(X_train,y_train)
```

Out[157…
```
SVC(kernel='linear')
```

In [158…
```
y_pred=clf.predict(X_test)
```

In [159…
```
confusion_matrix(y_test,y_pred)
```

Out[159…
```
array([[905, 128],
       [180, 196]], dtype=int64)
```

In [160…
```
accuracy_score(y_test,y_pred)
```

Out[160…
```
0.7814052519517388
```

In [161…
```
class_report = classification_report(y_test, y_pred)
print(class_report)
```

```
              precision    recall  f1-score   support

           0       0.83      0.88      0.85      1033
           1       0.60      0.52      0.56       376

    accuracy                           0.78      1409
   macro avg       0.72      0.70      0.71      1409
weighted avg       0.77      0.78      0.78      1409
```

Tried different kernels like linear, poly, rbf and sigmoid

Got the highest accuracy for linear

=> SVM - Accuracy of 78.1%

Justification file :

Algorithm with best accuracy :

All five algorithms provided accuracy around the same range but the algorithm with the highest accuracy is Logistic Regression with an accuracy of ***79.6%***.

The reason why accuracy isn't very high is because of the nature of the dataset where in the target variable 'churn' had high number of No's and less number of Yes's (0's and 1's) because of which there is an imbalance.

The reason why logistic regression has the highest accuracy is as follows :

- Let us notice the F1 score closely. F1 score is the measure of the harmonic mean of precision and recall. Commonly used as an evaluation metric in binary and multi-class classification, the F1 score integrates precision and recall into a single metric to gain a better understanding of model performance. It is 0.86 and 0.59 which signifies that the model can effectively identify positive cases while minimising false positives and false negatives.
- Precision is also higher than the ones achieved in other algorithms with 0.84 and 0.64
- Dataset is fairly linear allowing better capturing in this model

Result :

Therefore, we were successfully able to apply all the classification algorithms (KNN, Logisitc Regression, Naive Bayes, Decision Trees, SVM) on this dataset and print the accuracies. Additionally, we were able to justify reason for highest accuracy algorithm.