EX5 - CIA 1

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

1. Perform the necessary pre-processings.
2. Apply all the classification algorithms (KNN, Logisitc 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
5. zip the code files and the justification file and attach the zipped folder in the submission page

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
```

```
df = pd.read_csv("/content/Telco-Customer-Churn.csv")
```

```
df
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneServ
0	7590-VHVEG	Female	0	Yes	No	1	
1	5575-GNVDE	Male	0	No	No	34	
2	3668-QPYBK	Male	0	No	No	2	
3	7795-CFOCW	Male	0	No	No	45	
4	9237-HQITU	Female	0	No	No	2	
...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0	Yes	Yes	24	
7039	2234-XADUH	Female	0	Yes	Yes	72	
7040	4801-JZAZL	Female	0	Yes	Yes	11	
7041	8361-LTMKD	Male	1	Yes	No	4	
7042	3186-AJIEK	Male	0	No	No	66	

7043 rows × 21 columns

```
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
```

```
df.isnull().sum()
```

```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

1. Data Preprocessing

```
df = df.drop(["customerID"], axis = 1)
```

```
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"], errors="coerce")
```

```
df.isnull().sum()
```

```
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
```

```
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges       11
Churn               0
dtype: int64
```

```
df = df.fillna(df["TotalCharges"].mean())
```

```
df.isnull().sum()
```

```
gender             0
SeniorCitizen      0
Partner            0
Dependents         0
tenure             0
PhoneService       0
MultipleLines      0
InternetService    0
OnlineSecurity     0
OnlineBackup       0
DeviceProtection   0
TechSupport        0
StreamingTV        0
StreamingMovies    0
Contract           0
PaperlessBilling   0
PaymentMethod      0
MonthlyCharges     0
TotalCharges       0
Churn              0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   gender            7043 non-null   object
 1   SeniorCitizen     7043 non-null   int64
 2   Partner           7043 non-null   object
 3   Dependents        7043 non-null   object
 4   tenure            7043 non-null   int64
 5   PhoneService      7043 non-null   object
 6   MultipleLines     7043 non-null   object
 7   InternetService   7043 non-null   object
 8   OnlineSecurity    7043 non-null   object
 9   OnlineBackup      7043 non-null   object
 10  DeviceProtection  7043 non-null   object
 11  TechSupport       7043 non-null   object
 12  StreamingTV       7043 non-null   object
 13  StreamingMovies   7043 non-null   object
 14  Contract          7043 non-null   object
 15  PaperlessBilling  7043 non-null   object
```

```
 16  PaymentMethod     7043 non-null   object
 17  MonthlyCharges    7043 non-null   float64
 18  TotalCharges      7043 non-null   float64
 19  Churn             7043 non-null   object
dtypes: float64(2), int64(2), object(16)
memory usage: 1.1+ MB
```

```
df["PaymentMethod"].unique()
```

```
array(['Electronic check', 'Mailed check', 'Bank transfer (automatic)',
       'Credit card (automatic)'], dtype=object)
```

```
df.replace({'No phone service': "No", 'No internet service': "No"}, inplace=True)
```

```
df["OnlineSecurity"].unique()
```

```
array(['No', 'Yes'], dtype=object)
```

```
df["Churn"].unique()
```

```
array(['No', 'Yes'], dtype=object)
```

```
object_cols = df.select_dtypes(include=['object']).columns
le = LabelEncoder()
for col in object_cols:
  df[col] = le.fit_transform(df[col])
df
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Multipl
0	0	0	1	0	1	0	
1	1	0	0	0	34	1	
2	1	0	0	0	2	1	
3	1	0	0	0	45	0	
4	0	0	0	0	2	1	
...	...	...	...	...	...	...	
7038	1	0	1	1	24	1	
7039	0	0	1	1	72	1	
7040	0	0	1	1	11	0	
7041	1	1	1	0	4	1	
7042	1	0	0	0	66	1	

7043 rows × 20 columns

```
df1 = pd.get_dummies(df)
df1.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLi
0	0	0	1	0	1	0	
1	1	0	0	0	34	1	
2	1	0	0	0	2	1	
3	1	0	0	0	45	0	
4	0	0	0	0	2	1	

```
X = df1.drop(columns = ["Churn"])
y = df1["Churn"].values
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_s
```

Model Prediction

KNN

```
knn = KNeighborsClassifier(n_neighbors = 10)
knn.fit(X_train,y_train)
y_pred = knn.predict(X_test)
knn_acc = knn.score(X_test,y_test)
print("KNN accuracy:",knn_acc)
```

```
KNN accuracy: 0.7884524372929484
```

Logistic Regression

```
lr = LogisticRegression()
lr.fit(X_train,y_train)
lr_acc = lr.score(X_test,y_test)
print("Logistic Regression accuracy:",lr_acc)
```

```
Logistic Regression accuracy is : 0.807382867960246
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
  n_iter_i = _check_optimize_result(
```

SVM

```
svc = SVC(kernel="linear", random_state = 42)
svc.fit(X_train,y_train)
svc_acc = svc.score(X_test,y_test)
print("SVM accuracy:", svc_acc)
```

```
SVM accuracy: 0.792238523426408
```

Naive Bayes

```
nb = GaussianNB()
nb.fit(X_train,y_train)
nb_acc = nb.score(X_test,y_test)
print("Naive Bayes accuracy:",nb_acc)
```

```
Naive Bayes accuracy: 0.7586370089919545
```

Decision Tree

```
dt = DecisionTreeClassifier()
dt.fit(X_train,y_train)
dt_acc = dt.score(X_test,y_test)
print("Decision Tree accuracy:",dt_acc)
```

```
Decision Tree accuracy: 0.7259820160908661
```

Summary

- Logistic Regression has the highest accuracy, followed by SVM.
- Logistic Regression excels in binary classification, so it has higher accuracy for this dataset.
- Logistic Regression needs independent variables to be linearly related to the log-odds of the outcome, providing non-linear decision boundaries in probability space. It is also less sensitive to outliers and can perform well without extensive feature scaling.
- Decision Trees and SVM underperform when features are unscaled
- KNN is affected by outliers.
- Naive Bayes assumes independence between features, which contradicts the interdependence in the given dataset.