CS3006 Machine Learning Algorithms : CIA 1

CS3802 Machine Learning Algorithms Lab : Ex 5

21011102079

In [ ]:
```
import pandas as pd
```

In [ ]:
```
data  = pd.read_csv(r"D:/snu/academic/sem6/ML_Lab/Lab5/Telco-Customer-Churn.csv")
```

Data Pre-Processing

In [ ]:
```
data
```

Out[ ]:

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

7043 rows × 21 columns

In [ ]:
```
data.drop(['customerID'], axis=1, inplace=True)
```

In [ ]:
```
data.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   object
 19  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(17)
memory usage: 1.1+ MB
```

In [ ]:
```
null_values_count = data.isnull().sum()

print("Number of null values in each column:")
print(null_values_count)
```

```
Number of null values in each column:
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
```

the **TotalCharges** column contains null values and they might not be immediately visible in the output of data.info() and isnull() because it doesn't explicitly show the null values since it is of type Object and data present in is numeric.

In [ ]:
```
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors='coerce')
```

In [ ]:
```
null_values_count = data.isnull().sum()

print("Number of null values in each column:")
print(null_values_count)
```

```
Number of null values in each column:
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
```

In [ ]:
```
data.dropna(subset=['TotalCharges'], inplace=True)
```

In [ ]:
```
null_values_count = data.isnull().sum()

print("Number of null values in each column:")
print(null_values_count)
```

```
Number of null values in each column:
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
```

In [ ]:
```
data.head()
```

Out[ ]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetServic
0	Female	0	Yes	No	1	No	No phone service	DS
1	Male	0	No	No	34	Yes	No	DS
2	Male	0	No	No	2	Yes	No	DS
3	Male	0	No	No	45	No	No phone service	DS
4	Female	0	No	No	2	Yes	No	Fiber opti

In [ ]:
```
data.replace({'Yes': 1, 'No': 0, 'No phone service': 0, 'No internet service': 0}, inplace=
```

In [ ]:
```
data.head()
```

Out[ ]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetServic
0	Female	0	1	0	1	0	0	DS
1	Male	0	0	0	34	1	0	DS
2	Male	0	0	0	2	1	0	DS
3	Male	0	0	0	45	0	0	DS
4	Female	0	0	0	2	1	0	Fiber opti

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7032 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   gender            7032 non-null   object 
 1   SeniorCitizen     7032 non-null   int64  
 2   Partner           7032 non-null   int64  
 3   Dependents        7032 non-null   int64  
 4   tenure            7032 non-null   int64  
 5   PhoneService      7032 non-null   int64  
 6   MultipleLines     7032 non-null   int64  
 7   InternetService   7032 non-null   object 
 8   OnlineSecurity    7032 non-null   int64  
 9   OnlineBackup      7032 non-null   int64  
 10  DeviceProtection  7032 non-null   int64  
 11  TechSupport       7032 non-null   int64  
 12  StreamingTV       7032 non-null   int64  
 13  StreamingMovies   7032 non-null   int64  
 14  Contract          7032 non-null   object 
 15  PaperlessBilling  7032 non-null   int64  
 16  PaymentMethod     7032 non-null   object 
 17  MonthlyCharges    7032 non-null   float64
 18  TotalCharges      7032 non-null   float64
 19  Churn             7032 non-null   int64  
dtypes: float64(2), int64(14), object(4)
memory usage: 1.1+ MB
```

In [ ]:
```
object_columns = data.select_dtypes(include=['object']).columns

for column in object_columns:
    data = pd.get_dummies(data, columns=[column], drop_first=True, dtype=int)
```

In [ ]:
```
last_column = data.pop("Churn")
data["Churn"] = last_column
```

In [ ]:
```
pd.set_option('display.max_columns', None)
data.head()
```

Out[ ]:

	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	Onlinel
0	0	1	0	1	0	0	0	
1	0	0	0	34	1	0	1	
2	0	0	0	2	1	0	1	
3	0	0	0	45	0	0	1	
4	0	0	0	2	1	0	0	

Some machine learning algorithms are sensitive to the scale of the input features. Feature scaling ensures that all features contribute equally to the model. It standardizes or normalizes the range of independent variables or features of the dataset.

In [ ]:
```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
data[['tenure', 'MonthlyCharges', 'TotalCharges']] = scaler.fit_transform(data[['tenure', '
```

In [ ]:
```
data.head()
```

Out[ ]:

	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	Onli
0	0	1	0	-1.280248	0	0	0	
1	0	0	0	0.064303	1	0	1	
2	0	0	0	-1.239504	1	0	1	
3	0	0	0	0.512486	0	0	1	
4	0	0	0	-1.239504	1	0	0	

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7032 entries, 0 to 7042
Data columns (total 24 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   SeniorCitizen                          7032 non-null   int64
 1   Partner                                7032 non-null   int64
 2   Dependents                             7032 non-null   int64
 3   tenure                                 7032 non-null   float64
 4   PhoneService                           7032 non-null   int64
 5   MultipleLines                          7032 non-null   int64
 6   OnlineSecurity                         7032 non-null   int64
 7   OnlineBackup                           7032 non-null   int64
 8   DeviceProtection                       7032 non-null   int64
 9   TechSupport                            7032 non-null   int64
 10  StreamingTV                            7032 non-null   int64
 11  StreamingMovies                        7032 non-null   int64
 12  PaperlessBilling                       7032 non-null   int64
 13  MonthlyCharges                         7032 non-null   float64
 14  TotalCharges                           7032 non-null   float64
 15  gender_Male                            7032 non-null   int32
 16  InternetService_DSL                    7032 non-null   int32
 17  InternetService_Fiber optic            7032 non-null   int32
 18  Contract_One year                      7032 non-null   int32
 19  Contract_Two year                      7032 non-null   int32
 20  PaymentMethod_Credit card (automatic)  7032 non-null   int32
 21  PaymentMethod_Electronic check         7032 non-null   int32
 22  PaymentMethod_Mailed check             7032 non-null   int32
 23  Churn                                  7032 non-null   int64
dtypes: float64(3), int32(8), int64(13)
memory usage: 1.1 MB
```

Train Test Split

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
X = data.drop('Churn', axis=1)
Y = data['Churn']
```

In [ ]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,random_state = 42)
```

Import acuuracy_score

In [ ]:
```
from sklearn.metrics import accuracy_score
```

1. K-Nearest Neighbors (KNN)

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

In [ ]:
```
knn = KNeighborsClassifier(n_neighbors = 8)
knn.fit(X_train, Y_train)
knn_pred = knn.predict(X_test)
knn_accuracy = accuracy_score(Y_test, knn_pred)
print(f"KNN Accuracy: {knn_accuracy:.2f}")
```

```
KNN Accuracy: 0.78
```

2. Logistic Regression

In [ ]:
```
from sklearn.linear_model import LogisticRegression
```

In [ ]:
```
logreg = LogisticRegression()
logreg.fit(X_train, Y_train)
logreg_pred = logreg.predict(X_test)
logreg_accuracy = accuracy_score(Y_test, logreg_pred)
print(f"Logistic Regression Accuracy: {logreg_accuracy:.2f}")
```

```
Logistic Regression Accuracy: 0.80
```

3. Naive Bayes

In [ ]:
```
from sklearn.naive_bayes import GaussianNB
```

In [ ]:
```
nb = GaussianNB()
nb.fit(X_train, Y_train)
nb_pred = nb.predict(X_test)
nb_accuracy = accuracy_score(Y_test, nb_pred)
print(f"Naive Bayes Accuracy: {nb_accuracy:.2f}")
```

```
Naive Bayes Accuracy: 0.75
```

4. Decision Trees

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [ ]:
```
dt = DecisionTreeClassifier(max_depth = 4)
dt.fit(X_train, Y_train)
dt_pred = dt.predict(X_test)
dt_accuracy = accuracy_score(Y_test, dt_pred)
print(f"Decision Trees Accuracy: {dt_accuracy:.2f}")
```

```
Decision Trees Accuracy: 0.78
```

5. Support Vector Machine (SVM)

In [ ]:
```
from sklearn.svm import SVC
```

In [ ]:
```
svm = SVC()
svm.fit(X_train, Y_train)
svm_pred = svm.predict(X_test)
svm_accuracy = accuracy_score(Y_test, svm_pred)
print(f"SVM Accuracy: {svm_accuracy:.2f}")
```

```
SVM Accuracy: 0.80
```

Accuracy Score Comparison

In [ ]:
```
print(f"KNN                : {knn_accuracy:.2f}  Accuracy ")
print(f"Logistic Regression: {logreg_accuracy:.2f} Accuracy")
print(f"Naive Bayes        : {nb_accuracy:.2f} Accuracy")
print(f"Decision Trees     : {dt_accuracy:.2f} Accuracy")
print(f"SVM                : {svm_accuracy:.2f} Accuracy")
```

```
KNN                : 0.78  Accuracy
Logistic Regression: 0.80 Accuracy
Naive Bayes        : 0.75 Accuracy
Decision Trees     : 0.78 Accuracy
SVM                : 0.80 Accuracy
```

Both Logistic Regression and SVM models have the highest accuracy

But when feature scaling is not done for the given data the accuracy of SVM model is the lowest. This is because SVM is sensitive and influenced by the scale of the input features.

Logistic regression deals with binary classification and the given data requires binary classification. This algorithm has higher accuracy as it is optimised for binary classification than other given algorithms.

Logistic Regression assumes a linear relationship between the features and the log-odds of the target variable. Also logistic regression is a linear model in terms of its parameters, but its decision boundary is non-linear when considering the transformation of log odds to probabilities. This allows logistic regression to capture and model complex non-linear relationships between features and the target variable in the input space.

Logistic Regression is less sensitive to outliers and can perform well without extensive feature scaling. When the features of the given data were unscaled the accuracy of Decision Trees and SVM underperform. KNN is also be affected by outliers.

Models like Naive Bayes assume independence between features, which is not true in the given dataset as different features depend upon the each other. Decision Trees, if too deep, can overfit noisy or irrelevant features.

The similar performance across models may indicate that the given dataset does not have complex, non-linear relationships. Alternatively, it suggests that the models are effectively adapting to the dataset's characteristics, resulting in comparable accuracy. It also suggests that given dataset may not have distinctive patterns that strongly favor one modeling approach over another.