D209: Task 1 Classification Analysis

Import libraries and packages

In [23]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import roc_curve
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

Load and check data set

In [2]:
```
df = pd.read_csv('/Users/ebeth/Desktop/Churn Data/churn_clean.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	L
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	e885b299883d4f9fb18e39c75155d990	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.375
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	f2de8bef964785f41a2959829830fb8a	West Branch	MI	Ogemaw	48661	44.32893	-84.240
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	f1784cfa9f6d92ae816197eb175d3c71	Yamhill	OR	Yamhill	97148	45.35589	-123.246
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	dc8a365077241bb5cd5ccd305136b05e	Del Mar	CA	San Diego	92014	32.96687	-117.247
4	5	K662701	68ab861fd-0220-4e51-a587-8a90407ee574	aabb64a116e83fdc4befc1fbab1663f9	Needville	TX	Fort Bend	77461	29.38012	-95.806

5 rows × 50 columns

In [23]:
```
df.describe()
```

Out[21]:

	CaseOrder	Zip	Lat	Lng	Population	Children	Age	Income	Outage_s
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.00000	10000.0000	10000.000000	10000.000000	10
mean	5000.50000	49153.319600	38.757567	-90.782536	9756.562400	2.0877	53.078400	39806.926771	
std	2886.89568	27532.196108	5.437389	15.156142	14432.698671	2.1472	20.698882	28199.916702	
min	1.00000	601.000000	17.966120	-171.688150	0.00000	0.0000	18.000000	348.670000	
25%	2500.75000	26292.500000	35.341828	-97.082813	738.00000	0.0000	35.000000	19224.717500	
50%	5000.50000	48869.500000	39.395800	-87.918800	2910.50000	1.0000	53.000000	33170.605000	
75%	7500.25000	71866.500000	42.106908	-80.088745	13168.00000	3.0000	71.000000	53246.170000	
max	10000.00000	99929.000000	70.640660	-65.667850	111850.00000	10.0000	89.000000	258900.700000	

8 rows × 23 columns

Data Preparation

Check for null values

In [4]:
```
df.isnull().values.any()
```

Out[4]:
```
False
```

Check for duplicates

In [5]:
```
df.duplicated().values.any()
```

Out[5]:
```
False
```

Drop unused columns

In [6]:
```
df2 = df.drop(['CaseOrder','Customer_id','Interaction', 'UID','Lat','Lng','TimeZone','Job','City','County', 'St
```

Check for data types

In [7]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 30 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Population             10000 non-null  int64  
 1   Area                   10000 non-null  object 
 2   Children               10000 non-null  int64  
 3   Age                    10000 non-null  int64  
 4   Income                 10000 non-null  float64
 5   Marital                10000 non-null  object 
 6   Gender                 10000 non-null  object 
 7   Churn                  10000 non-null  object 
 8   Outage_sec_perweek     10000 non-null  float64
 9   Email                  10000 non-null  int64  
 10  Contacts               10000 non-null  int64  
 11  Yearly_equip_failure   10000 non-null  int64  
 12  Techie                 10000 non-null  object 
 13  Contract               10000 non-null  object 
 14  Port_modem             10000 non-null  object 
 15  Tablet                 10000 non-null  object 
 16  InternetService        10000 non-null  object 
 17  Phone                  10000 non-null  object 
 18  Multiple               10000 non-null  object 
 19  OnlineSecurity         10000 non-null  object 
 20  OnlineBackup           10000 non-null  object 
 21  DeviceProtection       10000 non-null  object 
 22  TechSupport            10000 non-null  object 
 23  StreamingTV            10000 non-null  object 
 24  StreamingMovies        10000 non-null  object 
 25  PaperlessBilling       10000 non-null  object 
 26  PaymentMethod          10000 non-null  object 
 27  Tenure                 10000 non-null  float64
 28  MonthlyCharge          10000 non-null  float64
 29  Bandwidth_GB_Year      10000 non-null  float64
dtypes: float64(5), int64(6), object(19)
memory usage: 2.3+ MB
```

Create Dummy variables for all categorical columns and drop unneeded columns. (code used from: https://towardsdatascience.com/the-dummys-guide-to-creating-dummy-variables-f21faddb1d40)

In [8]:
```
dummy1 = pd.get_dummies(df2.Area, prefix = 'Area', drop_first = True)
dummy2 = pd.get_dummies(df2.Marital, prefix = 'Marital', drop_first = True)
dummy3 = pd.get_dummies(df2.Gender, prefix = 'Gender', drop_first = True)
dummy4 = pd.get_dummies(df2.Churn, prefix = 'Churn', drop_first = True)
dummy5 = pd.get_dummies(df2.Techie, prefix = 'Techie', drop_first = True)
dummy6 = pd.get_dummies(df2.Contract, prefix = 'Contract', drop_first = True)
dummy7 = pd.get_dummies(df2.Port_modem, prefix = 'Port_modem', drop_first = True)
dummy8 = pd.get_dummies(df2.Tablet, prefix = 'Tablet', drop_first = True)
dummy9 = pd.get_dummies(df2.InternetService, prefix = 'InternetService', drop_first = True)
dummy10 = pd.get_dummies(df2.Phone, prefix = 'Phone', drop_first = True)
dummy11 = pd.get_dummies(df2.Multiple, prefix = 'Multiple', drop_first = True)
dummy12 = pd.get_dummies(df2.OnlineSecurity, prefix = 'OnlineSecurity', drop_first = True)
dummy13 = pd.get_dummies(df2.OnlineBackup, prefix = 'OnlineBackup', drop_first = True)
dummy14 = pd.get_dummies(df2.DeviceProtection, prefix = 'DeviceProtection', drop_first = True)
dummy15 = pd.get_dummies(df2.TechSupport, prefix = 'TechSupport', drop_first = True)
dummy16 = pd.get_dummies(df2.StreamingTV, prefix = 'StreamingTV', drop_first = True)
dummy17 = pd.get_dummies(df2.StreamingMovies, prefix = 'StreamingMovies', drop_first = True)
dummy18 = pd.get_dummies(df2.PaperlessBilling, prefix = 'PaperlessBilling', drop_first = True)
dummy19 = pd.get_dummies(df2.PaymentMethod, prefix = 'PaymentMethod', drop_first = True)

df2 = df2.drop(columns = 'Area').merge(dummy1, left_index = True, right_index = True)
df2 = df2.drop(columns = 'Marital').merge(dummy2, left_index = True, right_index = True)
df2 = df2.drop(columns = 'Gender').merge(dummy3, left_index = True, right_index = True)
df2 = df2.drop(columns = 'Churn').merge(dummy4, left_index = True, right_index = True)
df2 = df2.drop(columns = 'Techie').merge(dummy5, left_index = True, right_index = True)
df2 = df2.drop(columns = 'Contract').merge(dummy6, left_index = True, right_index = True)
df2 = df2.drop(columns = 'Port_modem').merge(dummy7, left_index = True, right_index = True)
df2 = df2.drop(columns = 'Tablet').merge(dummy8, left_index = True, right_index = True)
df2 = df2.drop(columns = 'InternetService').merge(dummy9, left_index = True, right_index = True)
df2 = df2.drop(columns = 'Phone').merge(dummy10, left_index = True, right_index = True)
df2 = df2.drop(columns = 'Multiple').merge(dummy11, left_index = True, right_index = True)
df2 = df2.drop(columns = 'OnlineSecurity').merge(dummy12, left_index = True, right_index = True)
df2 = df2.drop(columns = 'OnlineBackup').merge(dummy13, left_index = True, right_index = True)
df2 = df2.drop(columns = 'DeviceProtection').merge(dummy14, left_index = True, right_index = True)
df2 = df2.drop(columns = 'TechSupport').merge(dummy15, left_index = True, right_index = True)
df2 = df2.drop(columns = 'StreamingTV').merge(dummy16, left_index = True, right_index = True)
df2 = df2.drop(columns = 'StreamingMovies').merge(dummy17, left_index = True, right_index = True)
df2 = df2.drop(columns = 'PaperlessBilling').merge(dummy18, left_index = True, right_index = True)
df2 = df2.drop(columns = 'PaymentMethod').merge(dummy19, left_index = True, right_index = True)
```

In [9]:
```
df2.head()
```

Out[9]:

	Population	Children	Age	Income	Outage_sec_perweek	Email	Contacts	Yearly_equip_failure	Tenure	MonthlyCharge	...	O
0	38	0	68	28561.99	7.978323	10	0	1	6.795513	172.455519	...	
1	10446	1	27	21704.77	11.699080	12	0	1	1.156681	242.632554	...	
2	3735	4	50	9609.57	10.752800	9	0	1	15.754144	159.947583	...	
3	13863	1	48	18925.23	14.913540	15	2	0	17.087227	119.956840	...	
4	11352	0	83	40074.19	8.147417	16	2	1	1.670972	149.948316	...	

5 rows × 39 columns

Create clean copy of the data

In [10]:
```
df2.to_csv('classification_prepared_churn.csv')
```

Standardize the data

In [11]:
```
pipeline = make_pipeline(StandardScaler(), KNeighborsClassifier() )
```

K Nearest Neighbors with default settings

Split the data into train and test

In [12]:
```
y = df2['Churn_Yes']
X = df2.loc[:, df2.columns != 'Churn_Yes']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 13)
```

In [19]:
```
X_train.to_csv('X_train.csv')
X_test.to_csv('X_test.csv')
y_train.to_csv('y_train.csv')
y_test.to_csv('y_test.csv')
```

Fit the data to the model and predict.

In [13]:
```
knn_scaled = pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)
```

Calculate the confusion matrix

In [15]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[2020  190]
 [ 321  469]]
```

In [24]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.91      0.89      2210
           1       0.71      0.59      0.65       790

    accuracy                           0.83      3000
   macro avg       0.79      0.75      0.77      3000
weighted avg       0.82      0.83      0.82      3000
```

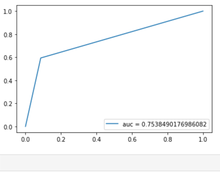
Calculate the accuracy score

In [14]:
```
knn_scaled.score(X_test, y_test)
```

Out[14]:
```
0.8296666666666667
```

Plot ROC and calculate the auc

In [16]:
```
pred_prob = pipeline.predict(X_test)
fpr, tpr, _ = metrics.roc_curve(y_test, pred_prob)
auc = metrics.roc_auc_score(y_test, pred_prob)
plt.plot(fpr, tpr, label = 'auc = '+str(auc))
plt.legend(loc = 4)
plt.show()
```



In [ ]:
```

```