D209 – Assessment NVM2 TASK 1: CLASSIFICATION ANALYSIS

Part I: Research Question

A. Describe the purpose of this data mining report by doing the following:

1. Propose **one** question relevant to a real-world organizational situation that you will answer using **one** of the following classification methods:
    - *k*-nearest neighbor (KNN)
    - Naive Bayes

Predicting which customers are more likely to churn. Which features provided in the churn dataset are more relevant to the analysis? I will be using k-nearest neighbor method.

2. Define **one** goal of the data analysis. Ensure that your goal is reasonable within the scope of the scenario and is represented in the available data.

The company's stakeholder will have access to know which features are more determinant in predict the churn and with that information they will be able to create/modify marketing campaigns to make sure more customers are retained.

Part II: Method Justification

B. Explain the reasons for your chosen classification method from part A1 by doing the following:

1. Explain how the classification method you chose analyzes the selected data set. Include expected outcomes.

This algorithm stores all available cases, classifying new ones by what the majority of k-nearest neighbors are classified. The algorithm will try to find the most similar point in the training data. We will choose "k" number of points. The majority classification of the k nearest points will determine how the new point will be classified. Then the algorithm will use the test data to validate the outcome.

Expected Outcomes: Just like performed in D208 using Logistic Regression to predict churn, I am going to split my dataset in 30% test and 70% train. KNN will classify our test data according to the closest neighbor classification (Yes/No for Churn or 1/0).

2. Summarize **one** assumption of the chosen classification method.

KNN classifies the new data points based on the similarity measure of the earlier stored data points.[1] This method assumes that the closest neighbors are similar enough to be classified like its neighbors.

3. List the packages or libraries you have chosen for Python or R, and justify how *each* item on the list supports the analysis.

I have been working with Python since D206 and will continue learning this language. I have been working with the free version of Pycharm (Community). The packages and libraries are:

-) Panda: package to read our dataset, present some statistics of the features, clean and modify data
-) Numpy: performs a wide variety of mathematical operations on arrays
-) Matplotlib: important package in any data science project. It helps with data visualization.
-) Scikit-learn: packages that will split, test, train and make predictions on our dataset
-) Seaborn: this package will be used for more detailed graphs and matrices

Part III: Data Preparation

C. Perform data preparation for the chosen data set by doing the following:

1. Describe **one** data preprocessing goal relevant to the classification method from part A1.

As explained in D208, mathematical models cannot be successfully accomplished with categorical data so before attempting a model we will have to create dummy variables to convert categorical into numerical data. For categorical data with only Yes and No, I will be modifying the feature to 1 and 0.

2. Identify the initial data set variables that you will use to perform the analysis for the classification question from part A1 and classify *each* variable as continuous or categorical.

The initial dataset is my entire dataset presented in "churn_raw_data.csv". Variables that have "int64" or "float64" are continuous and variables with "object" are categorical. The only exceptions are the variables listed below from 30 to 37. These are the customer survey variables and they are not continuous, they are discrete ordinal variables.

```
# Loading the Original Churn Dataset
churn_df2 = pd.read_csv('/Users/bia/Desktop/churn_raw_data.csv')

#Variables Types
churn_df2.info()
```

```
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Area                  10000 non-null  object
 1   Children              10000 non-null  float64
 2   Age                   10000 non-null  float64
 3   Employment            10000 non-null  object
 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
 30  CS Responses          10000 non-null  int64
 31  CS Fixes              10000 non-null  int64
 32  CS Replacements       10000 non-null  int64
 33  CS Reliability        10000 non-null  int64
 34  CS Options            10000 non-null  int64
 35  CS Respectfulness     10000 non-null  int64
 36  CS Courteous          10000 non-null  int64
 37  CS Listening          10000 non-null  int64
dtypes: float64(7), int64(11), object(20)
```

Picture 1: Dataset Info

3. Explain *each* of the steps used to prepare the data for the analysis. Identify the code segment for *each* step.

1-) Read the dataset using Panda (read_csv) and naming it "churn_df2

```
# Loading the Original Churn Dataset
churn_df2 = pd.read_csv('/Users/bia/Desktop/churn_raw_data.csv')
```

2-) Analyze basic stats of the dataset using ".describe()"

```
#Basic Stats
churn_desc = churn_df2.describe()
print(churn_desc)
```

```
memory usage: 4.0+ MB
None
       Unnamed: 0   CaseOrder  ...        item7         item8
count  10000.00000  10000.00000  ...  10000.000000  10000.000000
mean    5000.50000   5000.50000  ...      3.509500      3.495600
std     2886.89568   2886.89568  ...      1.028502      1.028633
min        1.00000      1.00000  ...      1.000000      1.000000
25%     2500.75000   2500.75000  ...      3.000000      3.000000
50%     5000.50000   5000.50000  ...      4.000000      3.000000
75%     7500.25000   7500.25000  ...      4.000000      4.000000
max    10000.00000  10000.00000  ...      7.000000      8.000000
```

Picture 2: Basic Stats of the dataset

3-) I dropped some features that would not help to predict the churn rate and also I replaced some non descriptive labels in Customer Survey columns (8 last columns).

```
#Dropping Columns Job, Timezone, Education and customer_ID
churn_df = churn_df2.drop(columns=['Job', 'Timezone', 'Customer_id',
'Education'])
#Dropping the 1st column "unnamed"
churn_df2 = churn_df2.iloc[:,1:]

#Dropping more non important columns to predict churn
churn_df2 = churn_df2.drop(columns=['CaseOrder', 'Interaction', 'City',
'State', 'County', 'Zip', 'Lat', 'Lng', 'Population'])

#Renaming the last 8 survey columns for a more descriptive value
churn_df2.rename(columns = {'item1':'CS Responses', 'item2':'CS Fixes',
'item3':'CS Replacements',
                            'item4':'CS Reliability', 'item5':'CS Options',
'item6':'CS Respectfulness',
                            'item7':'CS Courteous', 'item8':'CS
Listening'},inplace=True)
```

4-) Check for missing data.

```
#Finding missing values in my dataset
churn_df2.isnull().any(axis=1)
null_values = churn_df2.isna().any()
print(null_values)
#How many rows of data are we missing?
data_null_sum = churn_df2.isnull().sum()
print(data_null_sum)
```

```
[8 rows x 24 columns]
Area                     False
Children                  True
Age                       True
Employment               False
Income                    True
Marital                  False
Gender                   False
Churn                    False
Outage_sec_perweek       False
Email                    False
Contacts                 False
Yearly_equip_failure     False
Techie                    True
Contract                 False
Port_modem               False
Tablet                   False
InternetService          False
Phone                     True
Multiple                 False
OnlineSecurity           False
OnlineBackup             False
DeviceProtection         False
TechSupport               True
StreamingTV              False
StreamingMovies          False
PaperlessBilling         False
PaymentMethod            False
Tenure                    True
MonthlyCharge            False
Bandwidth_GB_Year         True
CS Responses             False
CS Fixes                 False
CS Replacements          False
CS Reliability           False
CS Options               False
CS Respectfulness        False
CS Courteous             False
CS Listening             False
dtype: bool
TODO   Problems   Terminal   Python Packa
```

Picture 3: Columns with "TRUE" have missing data

5-) The data presented missing values so for numerical features I replaced missing values for the median and for categorical for the mode.

```
#Filling the missing data with the median of each variable
#We saw that the columns Children, Age, Income, Tenure and Bandwidth_GB_Year
have missing values
na_cols = churn_df2.isna().any()
na_cols = na_cols[na_cols == True].reset_index()
na_cols = na_cols["index"].tolist()
for col in churn_df2.columns[1:]:
     if col in na_cols:
        if churn_df2[col].dtype != 'object':
              churn_df2[col] =
churn_df2[col].fillna(churn_df2[col].median()).round(0)
```

```
#Phone and Techie Columns are categorical with missing values as well
print(churn_df2['Phone'].unique())
print(churn_df2['Techie'].unique())
#Since We have "YES" "NO" and "NAN" we will need to replace the nan values
for something
df_stats_phone = churn_df2['Phone'].describe()
print(df_stats_phone)

df_stats_phone = churn_df2['Techie'].describe()
print(df_stats_phone)

#Since these are categorical columns, I am going to replace the "NAN" values
for whatever shows more
#Phone --> "YES"
#Techie --> "NO"

churn_df2 = churn_df2.fillna(churn_df2.mode().iloc[0])

# #Making sure all values were replaced by the median (num) and mode (cat),
so we check against missing data again
missing_data_clean = churn_df2.isna().any()
print(missing_data_clean)
```

```
Name: Techie, dtype: object
Area                    False
Children                False
Age                     False
Employment              False
Income                  False
Marital                 False
Gender                  False
Churn                   False
Outage_sec_perweek      False
Email                   False
Contacts                False
Yearly_equip_failure    False
Techie                  False
Contract                False
Port_modem              False
Tablet                  False
InternetService         False
Phone                   False
Multiple                False
OnlineSecurity          False
OnlineBackup            False
DeviceProtection        False
TechSupport             False
StreamingTV             False
StreamingMovies         False
PaperlessBilling        False
PaymentMethod           False
Tenure                  False
MonthlyCharge           False
Bandwidth_GB_Year       False
CS Responses            False
CS Fixes                False
CS Replacements         False
CS Reliability          False
CS Options              False
CS Respectfulness       False
CS Courteous            False
CS Listening            False
dtype: bool
```

Picture 4: Missing data successfully replaced

6-) Extract the clean data set:

```
# #Extracting the clean dataset
churn_df2.to_csv('churn_clean.csv')
churn_df2= pd.read_csv('churn_clean.csv')
```

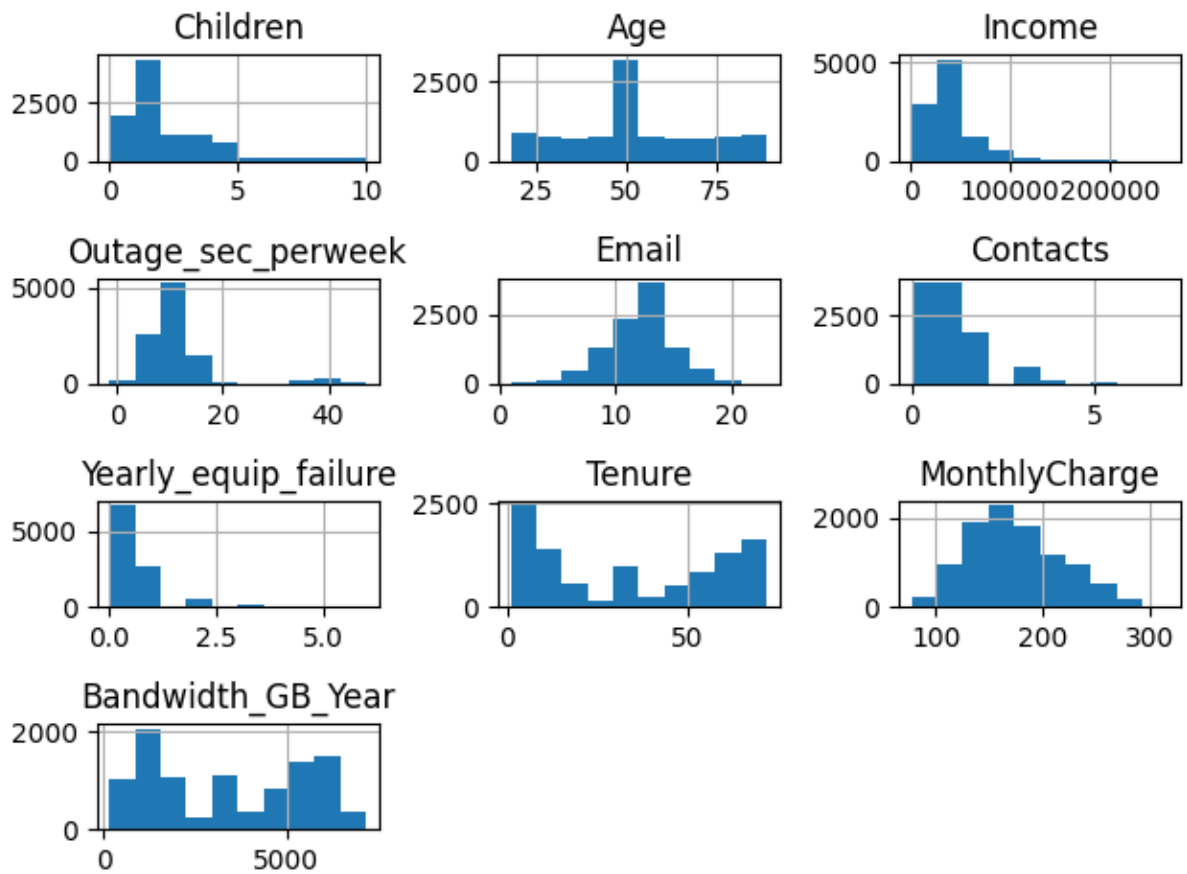
7-) Examine numerical data for outliers and plot a histogram

```
# Checking for outliers in the continuous variables
numerical_churn_df2 = churn_df2[['Tenure','MonthlyCharge',
'Bandwidth_GB_Year']]

# Checking for outliers at 25%, 50%, 75%, 90%, 95% and 99%
```

```
print(numerical_churn_df2.describe(percentiles=[.25, .5, .75, .90, .95,
.99]))

#Graphs of Numerical Features
churn_df2[['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email',
'Contacts', 'Yearly_equip_failure', 'Tenure',
           'MonthlyCharge', 'Bandwidth_GB_Year']].hist()
plt.savefig('churn_pyplot.jpg')
plt.tight_layout()
plt.show()
```

	Tenure	MonthlyCharge	Bandwidth_GB_Year
count	10000.000000	10000.000000	10000.000000
mean	34.640500	174.076305	3397.122700
std	25.188194	43.335473	2072.726654
min	1.000000	77.505230	156.000000
25%	9.000000	141.071078	1312.000000
50%	36.000000	169.915400	3382.000000
75%	60.000000	203.777441	5466.000000
90%	67.000000	238.683060	6094.100000
95%	70.000000	253.824616	6343.050000
99%	72.000000	275.859482	6717.010000
max	72.000000	315.878600	7159.000000

Picture 5: Percentile Analysis of Num Features



Picture 6: Histograms

8-) Change all categorical data either replacing Yes and No for 1 and 0 or creating dummy variables for features with 3 or more levels.

```
#Changing Variables from NO/YES to 0/1
churn_df2.Churn.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.Phone.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.PaperlessBilling.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.Techie.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.Port_modem.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.Tablet.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.Multiple.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.OnlineSecurity.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.OnlineBackup.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.DeviceProtection.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.TechSupport.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.StreamingTV.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.StreamingMovies.replace({"Yes":1, "No":0}, inplace = True)

#Creating a dummy variable for some of the categorical variables with 3 or
more levels
```

```
dummy1 = pd.get_dummies(churn_df2[['Marital', 'Contract', 'PaymentMethod',
'Gender', 'InternetService', 'Area',
                                    'Employment']])
#Adding the results to the master dataframe
churn_df2 = pd.concat([churn_df2, dummy1], axis=1)

#We have created dummies for the below variables, so we can drop them
churn_df2 = churn_df2.drop(['Marital', 'Contract', 'PaymentMethod', 'Gender',
'InternetService', 'Area','Employment'], 1)
```

9-) Extract the desired dataset (prepared_churn_data.csv) to start modelling using KNN method.

```
#Extract "Prepared"" dataset
churn_df2.to_csv('prepared_churn_data.csv')
churn_df2 = pd.read_csv('prepared_churn_data.csv')
df = churn_df2.columns
print('The dataset columns are ', df)
```

4. Provide a copy of the cleaned data set.

Part IV: Analysis

D. Perform the data analysis and report on the results by doing the following:
   1. Split the data into training and test data sets and provide the file(s).

```
#KNN Model
#Y variable is the target: Churn
y = churn_df2.Churn.values
#Removing Churn from remaining data
X = churn_df2.drop('Churn', axis = 1)

SEED = 1
#Train - Test - Split: 70%-30%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .3,
random_state = SEED)

# Instantiate KNN model
knn = KNeighborsClassifier(n_neighbors = 6)

#Fit the data
knn.fit(X_train, y_train)
#Predict outcomes from test set
y_pred = knn.predict(X_test)

print('Initial accuracy score KNN model: ', accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

#Calculation of the Confusion Matrix
confusion_matrix_initial = confusion_matrix(y_test, y_pred)
print('Confusion Matrix is: ',confusion_matrix_initial)
```

2. Describe the analysis technique you used to appropriately analyze the data. Include screenshots of the intermediate calculations you performed.

I started the analysis using the holdout method. I split my test and train data into 30%-70% respectively. I chose a random number of k-neighbors = 6. The accuracy with these settings is **0.73**. It's not great but also not bad.

```
Initial accuracy score KNN model:  0.73
              precision    recall  f1-score   support

           0       0.77      0.90      0.83      2174
           1       0.52      0.30      0.38       826

    accuracy                           0.73      3000
   macro avg       0.64      0.60      0.60      3000
weighted avg       0.70      0.73      0.70      3000

Confusion Matrix is:  [[1946  228]
 [ 582  244]]

Process finished with exit code 0
```

Picture 7: Initial Accuracy Score KNN Model

To check if I am able to improve the accuracy, I am now going to create a pipeline and normalize the data. A pipeline is a way to automate the machine learning workflow by allowing preprocessing of the data and instantiation of the estimator to occur in a single piece of code [6].

```
#Create pipeline object
#Normalize Data
steps = [('scaler', StandardScaler()),('knn', KNeighborsClassifier())]

#Split Dataframe
pipeline = Pipeline(steps)

#Scale dataframe with pipeline object
X_train_scaled, X_test_scaled, y_train_scaled, y_test_scaled =
train_test_split(X, y, test_size = 0.3, random_state = SEED)
knn_scaled = pipeline.fit(X_train_scaled, y_train_scaled)
#Predict from scaled dataframe
y_pred_scaled = pipeline.predict(X_test_scaled)

#Print new accuracy
print('New accuracy score of scaled KNN model:
{:0.3f}'.format(accuracy_score(y_test_scaled, y_pred_scaled)))
```

```
#Classification Report after scaling
print(classification_report(y_test_scaled, y_pred_scaled))

#Calculation of the Confusion Matrix
confusion_matrix = confusion_matrix(y_test_scaled, y_pred_scaled)
print('Confusion Matrix Scaled Model is: ',confusion_matrix)
```

```
Initial accuracy score KNN model:  0.73
              precision    recall  f1-score   support

           0       0.77      0.90      0.83      2174
           1       0.52      0.30      0.38       826

    accuracy                           0.73      3000
   macro avg       0.64      0.60      0.60      3000
weighted avg       0.70      0.73      0.70      3000

Confusion Matrix is:  [[1946  228]
 [ 582  244]]
New accuracy score of scaled KNN model: 0.818
              precision    recall  f1-score   support

           0       0.85      0.92      0.88      2174
           1       0.72      0.56      0.63       826

    accuracy                           0.82      3000
   macro avg       0.78      0.74      0.75      3000
weighted avg       0.81      0.82      0.81      3000

Confusion Matrix Scaled Model is:  [[1990  184]
 [ 363  463]]
AUC Score is  0.7563484143442979

Process finished with exit code 0
```

Picture 8: Initial and Scaled Models

We notice that the scaled model is more accurate. It's **0.82** which is considered a good model.

3. Provide the code used to perform the classification analysis from part D2.
   Code is along the documentation.

Part V: Data Summary and Implications

E. Summarize your data analysis by doing the following:
   1. Explain the accuracy and the area under the curve (AUC) of your classification model.

The accuracy has improved from 0.73 to 0.82 when the model is scaled.

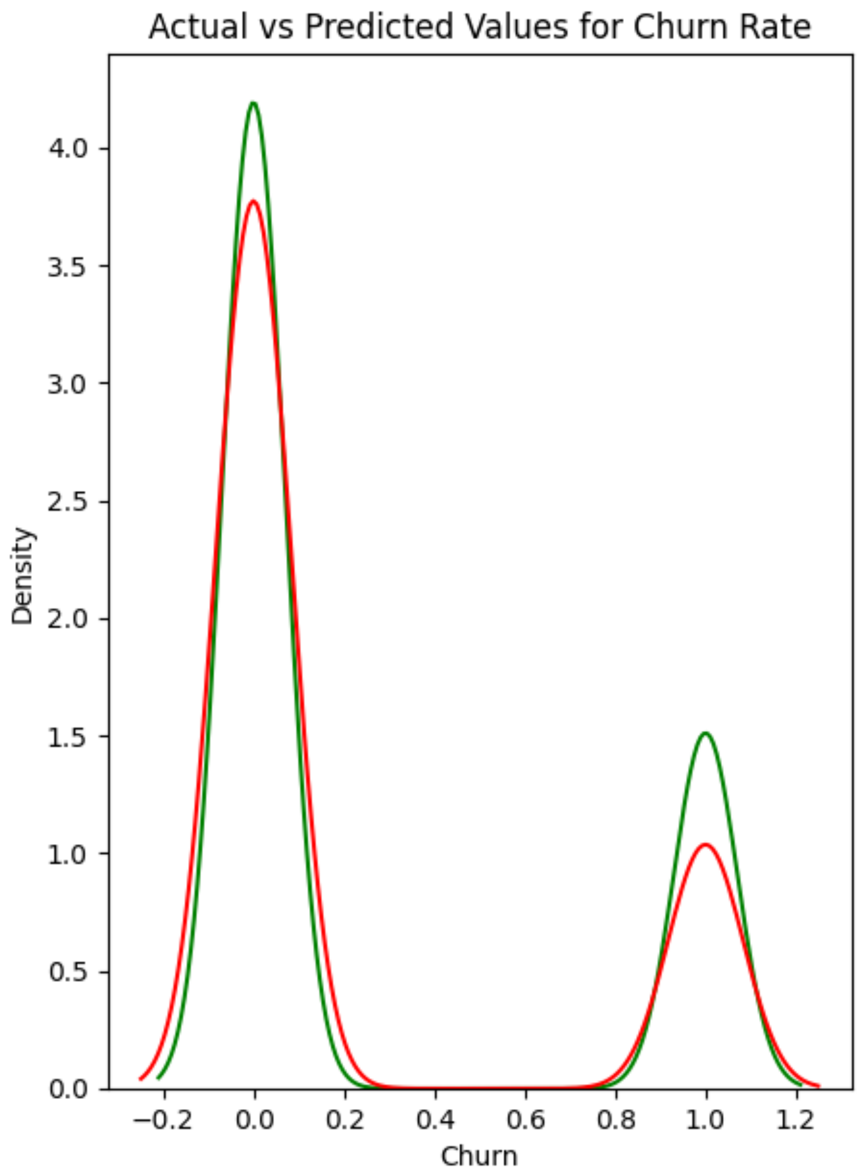
```
#Calculating AUC Score
pred_prob = knn.predict_proba(X_test)

#ROC Curve
fpr1, tpr1, thresh1 = roc_curve(y_test, pred_prob[:,1], pos_label=1)

#ROC curve for tpr = fpr
random_probs = [0 for i in range(len(y_test))]
p_fpr, p_tpr, _ = roc_curve(y_test, random_probs, pos_label=1)

#AUC Score
auc_score = roc_auc_score(y_test, pred_prob[:,1])

print('AUC Score is ',auc_score)
```

The Area Under the Curve (AUC) is **the measure of the ability of a classifier to distinguish between classes** and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes [3].

AUC is the area under the ROC. AUC Score is 0.76 (shown in Picture 8 above) and it represents a good accuracy.

```
plt.figure(figsize=(5, 7))
ax = sns.distplot(churn_df2['Churn'], hist=False, color="g", label="Actual
Values")
sns.distplot(y_pred_scaled, hist=False, color="r", label="Predicted Values",
ax=ax)
plt.title('Actual vs Predicted Values for Churn Rate')
plt.show()
```

In the plot below we can see the actual churn values in green and the predicted churn values in red.



Picture 9: Actual vs Predicted Churn Values

2. Discuss the results and implications of your classification analysis.

The AUC score is a good metric to measure binary classification[4]. Our AUC score of 0.76 means that our model has a decent accuracy [2].

In order to try to improve our accuracy, I will implement the **cross validation** method and find the optimal value of “n-neighbors”.

k-Fold Cross Validation: Cross-validation is when the dataset is randomly split up into ‘k’ groups [5]. One group is the test and the rest is the training set. The model is trained and a score is achieved. This same process is repeated for every distinct group (since my cross validation is 5, we will do this same step 5 times, every time using a different group as the test set).

```
#Import GridSearchCV for cross validation of model
from sklearn.model_selection import GridSearchCV

#k-Fold Cross Validation Method to find the optimal number for n-neighbors
param_grid = {'n_neighbors': np.arange(1, 50)}
knn = KNeighborsClassifier()
knn_cv = GridSearchCV(knn , param_grid, cv=5)
#Fit Model
knn_cv.fit(X_train, y_train)
#Print the optimal number of n_neighbors
print('Best parameters for this KNN model: {}'.format(knn_cv.best_params_))
```

```
Initial accuracy score KNN model:  0.73
              precision    recall  f1-score   support

           0       0.77      0.90      0.83      2174
           1       0.52      0.30      0.38       826

    accuracy                           0.73      3000
   macro avg       0.64      0.60      0.60      3000
weighted avg       0.70      0.73      0.70      3000

Confusion Matrix is:  [[1946  228]
 [ 582  244]]
New accuracy score of scaled KNN model: 0.818
              precision    recall  f1-score   support

           0       0.85      0.92      0.88      2174
           1       0.72      0.56      0.63       826

    accuracy                           0.82      3000
   macro avg       0.78      0.74      0.75      3000
weighted avg       0.81      0.82      0.81      3000

Confusion Matrix Scaled Model is:  [[1990  184]
 [ 363  463]]
AUC Score is  0.7563484143442979
Best parameters for this KNN model: {'n_neighbors': 6}

Process finished with exit code 0
```

Picture 10: Displaying Optimal n_neighbors number using k-Fold Cross Validation Method

The result of the cross validation is that my ideal n-neighbor would be 6. Luckily that's exactly the number I chose to begin with.

3. Discuss **one** limitation of your data analysis.

This analysis requires an arbitrary "k" number. Different "k" brings different results. It was a coincidence that my arbitrary number matched the ideal n-neighbor result from the cross validation method.

4. Recommend a course of action for the real-world organizational situation from part A1 based on your results and implications discussed in part E2.

Our model accuracy has a good accuracy but not great. The recommendation would be sign up customers with a longer contract and using multiple services at the same time, trying to decrease the chances of churn.

Part VI: Demonstration

F. Provide a Panopto video recording that includes a demonstration of the functionality of the code used for the analysis and a summary of the programming environment.

The video recorded can be found here:
https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=3e488947-79e3-442a-ba4f-adaf010a9b19

G. Record the web sources used to acquire data or segments of third-party code to support the analysis. Ensure the web sources are reliable.

H. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

[1] (2020, May 25th) KUMAR, Aditya KNN Algorithm: When? Why? How? https://towardsdatascience.com/knn-algorithm-what-when-why-how-41405c16c36f

[2] (2019, May 14th) ZHOU, Xiaoliang Receiver Operating Characteristic (ROC) Area Under the Curve (AUC): A Diagnostic Measure for Evaluating the Accuracy of Predictors of Education Outcomes https://www.tc.columbia.edu/elda/blog/content/receiver-operating-characteristic-roc-area-under-the-curve-auc/

[3] (2020, June 16th) BHANDARI, Aniruddha AUC-ROC Curve in Machine Learning Clearly Explained https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/

[4] (2019, Oct 26th) CHOU, Sin-Yi AUC – Insiders Guide To the Theory and Applications https://sinyi-chou.github.io/classification-auc/

[5] (2018, Sept 26th) ALLIBHAI, Eijaz Building a k-Nearest-Neighbors (k-NN) Model with Scikit-learn https://towardsdatascience.com/building-a-k-nearest-neighbors-k-nn-model-with-scikit-learn-51209555453a

[6] (2021, Jan 25th) BOYLES, Jennifer Beginner's Guide to k-Nearest Neighbors & Pipeline in Classification https://medium.com/analytics-vidhya/beginners-guide-to-k-nearest-neighbors-pipelines-in-classification-704b87f534e2