Part 1: Research Question

A1) Research Question

The question that will be answered is if we can predict which customers are at a higher risk of churn. We will answer this using the k-nearest neighbor’s method.

A2) Defined Goal

The goal of this analysis is to develop a model to help a company identify customers that are at a higher risk of churning using k-nearest neighbors. Also, to identify key features that cause a customer to churn.

Part 2: Method Justification

B1) Classification Method

KNN analyzes the data set by taking a new case and looking at the closest ‘k’ number of data points then taking a ‘majority vote’ on what label that specific unlabeled point should have. (3)

B2) Summary of Assumption

The main assumption of the knn method is that the data points that ‘exist in close proximity to each other are highly similar’. (4)

B3) Packages and Libraries

|  |  |
| --- | --- |
| Pandas | Importing data into a DataFrame and manipulations |
| Numpy | Provides arrays for required calculations |
| Seaborn | Better visualizations |
| Matplotlib.pyplot | Graphs and visualizations |
| Sklearn.model\_selection  Import train\_test\_split | Splitting the data into a training and a testing set |
| Sklearn.model\_selection  Import GridSearchCV | Allows us to use gridsearch cross-validation to identify best k-value |
| Sklearn.metrics  Import confusion\_matrix | Allows us to make a confusion matrix |
| Sklearn.metrics  Import roc\_auc\_score | Computes the area under the curve score |
| Sklearn.metrics  Import classification\_report | Builds a text report for the main classification methods |
| Sklearn.neighbors  Import KNeighborsClassifier | Allows us to use the KNN model |

Part 3: Data Summary and Implications

C1) Preprocessing Goal

Two goals for preprocessing are to check for missing / null values and fill in if any are found. As well as encoding any categorical variables (yes / no) into 1’s and 0’s using dummy variables.

C2) Identifying Dataset Variables

For this analysis I will be using the following variables, as well as the final 8 survey questions.

|  |  |
| --- | --- |
| Variable | Type |
| Children | Continuous |
| Age | Continuous |
| Income | Continuous |
| Outage\_sec\_perweek | Continuous |
| Contacts | Continuous |
| Yearly\_equip\_failure | Continuous |
| Tenure | Continuous |
| Monthly\_charge | Continuous |
| Bandwqidth\_GB\_Year | Continuous |
| Churn | Categorical |
| Techie | Categorical |
| Contract | Categorical |
| InternetService | Categorical |
| Phone | Categorical |
| Multiple | Categorical |
| OnlineSecurity | Categorical |
| OnlineBackup | Categorical |
| DeviceProtection | Categorical |
| TechSupport | Categorical |
| StreamingTV | Categorical |
| StreamingMovies | Categorical |

C3) Steps for Analysis and Code

**Step 1** – Load dataset into Python and Describe the data

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**#Import all packages*

*import pandas as pd*

*import numpy as np*

*from pandas import Series, DataFrame*

*import seaborn as sns*

*import matplotlib.pyplot as plt*

*%matplotlib inline*

*import sklearn*

*from sklearn.neighbors import KNeighborsClassifier*

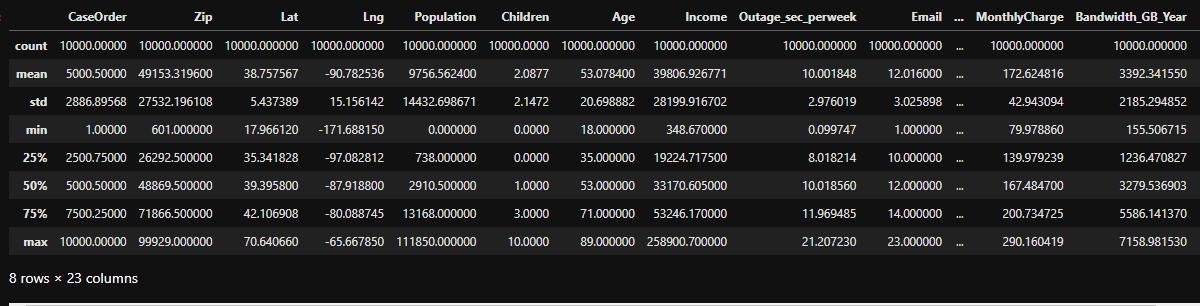
*from sklearn.model\_selection import train\_test\_split, GridSearchCV*

*from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score*

*#Load the data set into Pandas*

*df = pd.read\_csv('churn\_clean.csv')*

*df.describe()*

**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Step 2** – Rename the final 8 columns for better visibility, then drop the less meaningful ones

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*#Rename the final 8 columns for better visibility*

*df.rename(columns = {'Item1' : 'TimelyResponse',*

*'Item2' : 'TimelyFixes',*

*'Item3' : 'TimelyReplacement',*

*'Item4' : 'Reliability',*

*'Item5' : 'Options',*

*'Item6' : 'RespectfulResponse',*

*'Item7' : 'CourteousExchange',*

*'Item8' : 'EvidenceOfListening'},*

*inplace = True)*

*#Drop the less meaningul columns from the data set*

*df = df.drop(columns = ['CaseOrder', 'Customer\_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job', 'Marital', 'Gender', 'Email', 'Port\_modem', 'Tablet', 'PaperlessBilling', 'PaymentMethod'])*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Step 3** – Check for missing data, impute any that are found with either mean/median/mode

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*#Search for missing data*

*df.isnull().sum()*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

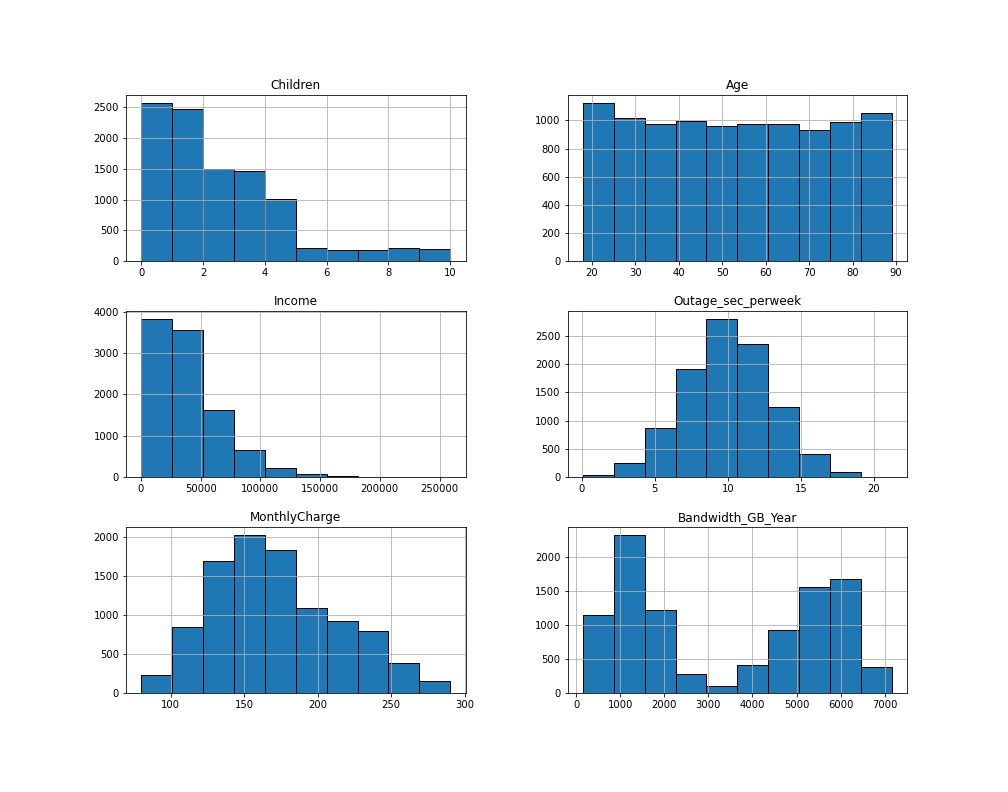
**Step 4** – Create some graphs to better visualize the data

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*#Create histograms for some continuous variables*

*df[['Children', 'Age', 'Income', 'Outage\_sec\_perweek', 'MonthlyCharge', 'Bandwidth\_GB\_Year']].hist(ec = "black", figsize = (14, 11))*

*#plt.savefig('Histogram1.jpg')*

**

*#Create a few seaborn plots for some categorical variables (1)(2)*

*ax = plt.subplots(figsize = (7, 7))*

*ax = sns.countplot(data = df, x = "Contract", hue = "Churn",*

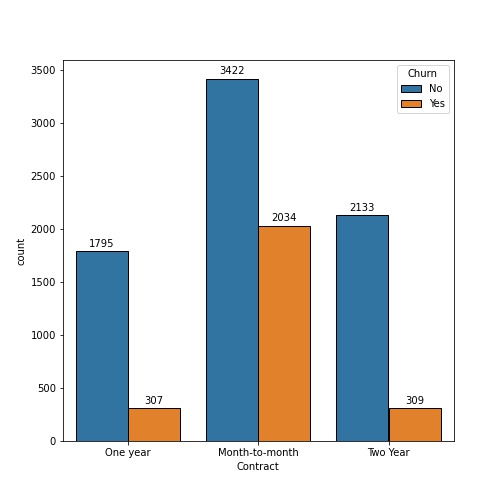
*ec = "black")*

*for x in ax.patches:*

*height = x.get\_height()*

*ax.text(x.get\_x() + x.get\_width() / 2, height + 40, height, ha = "center")*

*#plt.savefig('Contract.jpg')*

**

*#Create a few seaborn plots for some categorical variables (1)(2)*

*ax = plt.subplots(figsize = (7, 7))*

*ax = sns.countplot(data = df, x = "InternetService", hue = "Churn",*

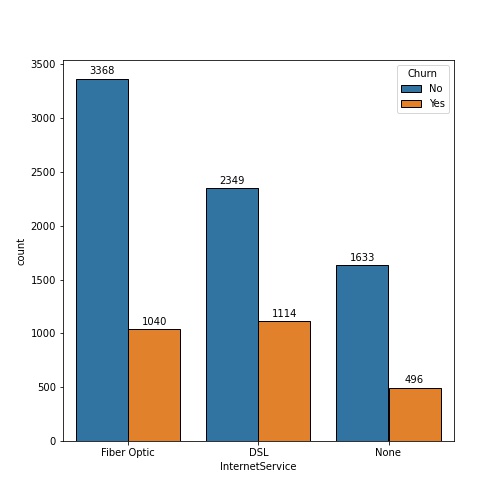
*ec = "black")*

*for x in ax.patches:*

*height = x.get\_height()*

*ax.text(x.get\_x() + x.get\_width() / 2, height + 40, height, ha = "center")*

*#plt.savefig('ISP.jpg')*

**

*#Create a few seaborn plots for some categorical variables (1)(2)*

*ax = plt.subplots(figsize = (5, 7))*

*ax = sns.countplot(data = df, x = "StreamingMovies", hue = "Churn", order = ['Yes', 'No'],*

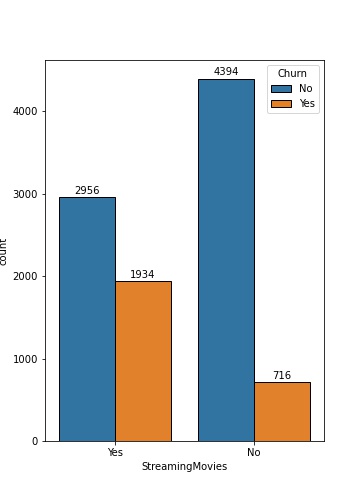
*ec = "black")*

*for x in ax.patches:*

*height = x.get\_height()*

*ax.text(x.get\_x() + x.get\_width() / 2, height + 40, height, ha = "center")*

*#plt.savefig('Movie.jpg')*

**

*#Create a few seaborn plots for some categorical variables (1)(2)*

*ax = plt.subplots(figsize = (5, 7))*

*ax = sns.countplot(data = df, x = "StreamingTV", hue = "Churn", order = ['Yes', 'No'],*

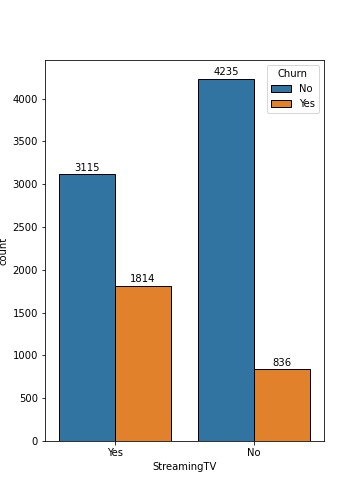
*ec = "black")*

*for x in ax.patches:*

*height = x.get\_height()*

*ax.text(x.get\_x() + x.get\_width() / 2, height + 40, height, ha = "center")*

*#plt.savefig('TV.jpg')*

**

*#Create a few seaborn plots for some categorical variables (1)(2)*

*ax = plt.subplots(figsize = (5, 7))*

*ax = sns.countplot(data = df, x = "Multiple", hue = "Churn", order = ['Yes', 'No'],*

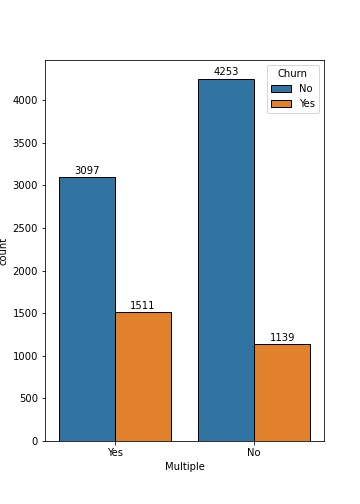
*ec = "black")*

*for x in ax.patches:*

*height = x.get\_height()*

*ax.text(x.get\_x() + x.get\_width() / 2, height + 40, height, ha = "center")*

*#plt.savefig('Multiple.jpg')*

**

*#Create a few seaborn plots for some categorical variables (1)(2)*

*ax = plt.subplots(figsize = (5, 7))*

*ax = sns.countplot(data = df, x = "Techie", hue = "Churn", order = ['Yes', 'No'],*

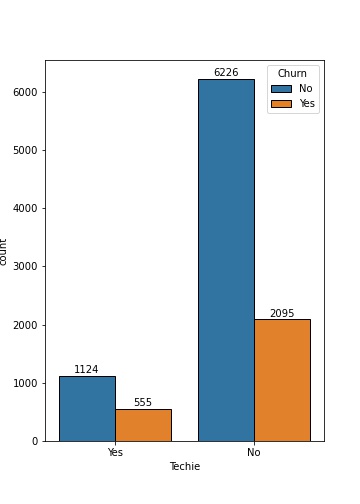
*ec = "black")*

*for x in ax.patches:*

*height = x.get\_height()*

*ax.text(x.get\_x() + x.get\_width() / 2, height + 40, height, ha = "center")*

*#plt.savefig('Techie.jpg')*

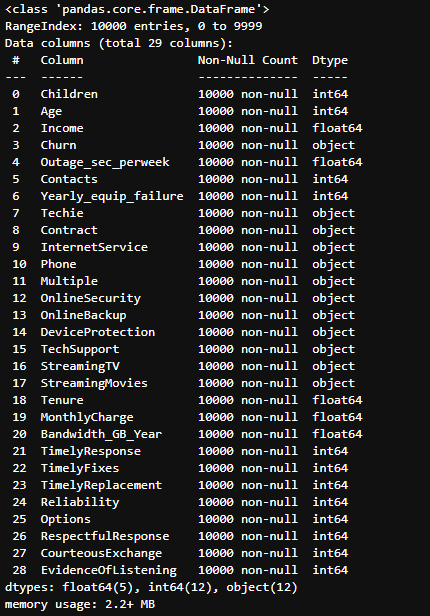
**

**Step 5** – See which variables are categorical, then encode them to numeric values using dummy variables. Finally, extract prepared dataset

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*#See which variables are categorical*

*df.info()*

**

*#No missing data, now to use ordinal encoding to replace the categorical values with numeric ones*

*#Yes to 1, No to 0*

*df['Churn\_num'] = df['Churn']*

*df['Techie\_num'] = df['Techie']*

*df['Contract\_num'] = df['Contract']*

*df['InternetService\_num'] = df['InternetService']*

*df['Phone\_num'] = df['Phone']*

*df['Multiple\_num'] = df['Multiple']*

*df['OnlineSecurity\_num'] = df['OnlineSecurity']*

*df['OnlineBackup\_num'] = df['OnlineBackup']*

*df['DeviceProtection\_num'] = df['DeviceProtection']*

*df['TechSupport\_num'] = df['TechSupport']*

*df['StreamingTV\_num'] = df['StreamingTV']*

*df['StreamingMovies\_num'] = df['StreamingMovies']*

*#Set up dictionary for converting to numeric values*

*dict\_churn = {"Churn\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_techie = {"Techie\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_phone = {"Phone\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_multiple = {"Multiple\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_security = {"OnlineSecurity\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_backup = {"OnlineBackup\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_protection = {"DeviceProtection\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_tech = {"TechSupport\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_tv = {"StreamingTV\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_movie = {"StreamingMovies\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_contract = {"Contract\_num" : {"Month-to-month" : 0, "One year" : 1, "Two Year" : 2}}*

*dict\_isp = {"InternetService\_num" : {"None" : 0, "DSL" : 1, "Fiber Optic" : 2}}*

*#Replace the variables values*

*df.replace(dict\_churn, inplace = True)*

*df.replace(dict\_techie, inplace = True)*

*df.replace(dict\_phone, inplace = True)*

*df.replace(dict\_multiple, inplace = True)*

*df.replace(dict\_security, inplace = True)*

*df.replace(dict\_backup, inplace = True)*

*df.replace(dict\_protection, inplace = True)*

*df.replace(dict\_tech, inplace = True)*

*df.replace(dict\_tv, inplace = True)*

*df.replace(dict\_movie, inplace = True)*

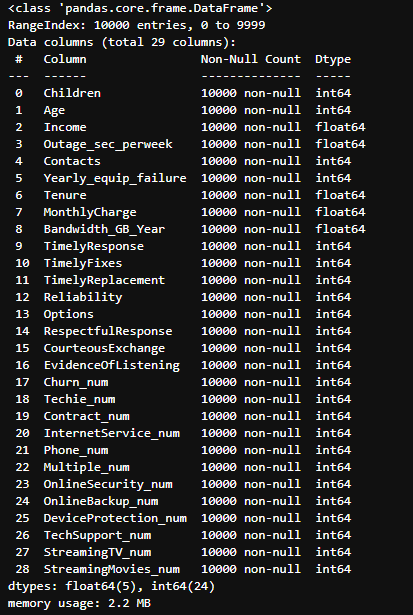
*df.replace(dict\_contract, inplace = True)*

*df.replace(dict\_isp, inplace = True)*

*#Now that we have those as numeric, we can drop the original columns*

*df = df.drop(columns = ['Churn', 'Techie', 'Phone', 'Multiple', 'OnlineSecurity','OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies','Contract', 'InternetService'])*

*df.info()*

**

*#Extract prepared dataset*

*df.to\_csv('churn\_prepared.csv')*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Part 4: Analysis

D1) Train / Test Split

Attached

D2) Describe Technique

The way I will go about this analysis is that after the data is split into the training and testing sets, I’ll use GridSearchCV to identify the ideal k-value for the model. After that I’ll fit the model using the number of neighbors previously identified then check the accuracy of the analysis.

D3) Code

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*#Import all packages*

*import pandas as pd*

*import numpy as np*

*from pandas import Series, DataFrame*

*import seaborn as sns*

*import matplotlib.pyplot as plt*

*%matplotlib inline*

*import sklearn*

*from sklearn.neighbors import KNeighborsClassifier*

*from sklearn.model\_selection import train\_test\_split, GridSearchCV*

*from sklearn.metrics import classification\_report, confusion\_matrix, roc\_auc\_score*

*#Load the data set into Pandas*

*df = pd.read\_csv('churn\_prepared.csv', index\_col = 0)*

*df.describe()*

*#Set up for train - test split*

*X = df.drop('Churn\_num', axis = 1).values*

*y = df['Churn\_num'].values*

*#Split the data set with an 80/20 split*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size = 0.8, test\_size = 0.2, random\_state = 25)*

*#Save the training and testing sets as csv files*

*pd.DataFrame(X\_train).to\_csv('X\_train.csv')*

*pd.DataFrame(X\_test).to\_csv('X\_test.csv')*

*pd.DataFrame(y\_train).to\_csv('y\_train.csv')*

*pd.DataFrame(y\_test).to\_csv('y\_test.csv')*

*#Run gridsearch cv to find best number of k*

*param\_grid = {'n\_neighbors': np.arange(1, 30)}*

*knn = KNeighborsClassifier()*

*knn\_cv = GridSearchCV(knn, param\_grid)*

*knn\_cv.fit(X\_train, y\_train)*

*print('The best parameters for this model: {}'.format(knn\_cv.best\_params\_))*

**

*#Fit the KNN model using grid search result of k = 20*

*knn = KNeighborsClassifier(n\_neighbors = 20)*

*knn.fit(X\_train, y\_train)*

*y\_pred = knn.predict(X\_test)*

*y\_pred\_prob = knn.predict\_proba(X\_test)[:,1]*

*#Print accuracy and AUC*

*print("The accuracy of the model is: ", knn.score(X\_test, y\_test))*

*print("The area under the curve (AUC) is: ", roc\_auc\_score(y\_test, y\_pred\_prob))*

**

*#Print confusion matrix*

*cnf\_matrix = confusion\_matrix(y\_test, y\_pred)*

*print(cnf\_matrix)*

**

*#Use seaborn heatmap to visualize the confusion matrix*

*sns.heatmap(pd.DataFrame(cnf\_matrix), annot = True, fmt = 'g')*

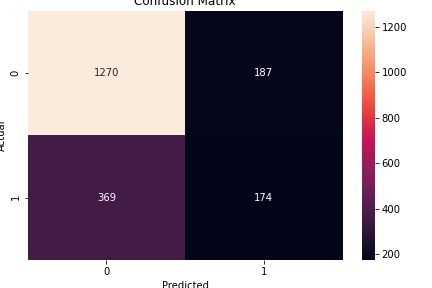
*plt.tight\_layout()*

*plt.title('Confusion Matrix')*

*plt.ylabel('Actual')*

*plt.xlabel('Predicted')*

*plt.savefig('matrix1.jpg')*

**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Part 5: Data Summary and Implications

E1) Accuracy and AUC of Model

As seen above, after fitting a KNN model to the data set the accuracy, being the performance of the model found by dividing the number of correct predictions by the number of total predictions is 0.722. The AUC, which gives a measure of performance across all classification thresholds is 0.767.

E2) Results and Implications

With the accuracy score shown above, we can determine this model isn’t as accurate as we would want. To get a better score we could try to scale the data and try to eliminate a few other less meaningful columns. The implication here being that the amount of different data points might be too complex for a k-nn analysis and we could be better served using a different classification method.

E3) Limitation

One limitation of the KNN algorithm and with my analysis is that KNN doesn’t work as well with higher level dimensions as it would take much more computational power. (5)

E4) Course of Action

From here the best course of action would be further analysis on which variables have the highest impact on customer churn and remove the others. This could provide a more accurate model to determine the rate of churn through the KNN model.

Part 6: Demonstration

F) Panopto Video

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=0160b6d6-01d0-40ac-acb7-aecc0163f248>

G) Third-Party Code Sources

1) “Dataforeverybody.com.” *Dataforeverybodycom*, https://www.dataforeverybody.com/countplot-seaborn-order-size-values/.

2) Tjs01Tjs01 35722 gold badges44 silver badges1414 bronze badges, et al. “Display Count on Top of Seaborn Barplot.” *Stack Overflow*, 1 Sept. 1966, https://stackoverflow.com/questions/55104819/display-count-on-top-of-seaborn-barplot.

H) Sources

3) “The Classification Challenge: Python.” *Campus.datacamp.com*, https://campus.datacamp.com/courses/machine-learning-with-scikit-learn/classification?ex=6.

4) Nelson, Daniel. “What Is a KNN (K-Nearest Neighbors)?” *Unite.AI*, 23 Aug. 2020, https://www.unite.ai/what-is-k-nearest-neighbors/.

5) Jain, Deepak. “KNN: Failure Cases, Limitations and Strategy to Pick Right K.” *Medium*, Level Up Coding, 17 July 2020, https://levelup.gitconnected.com/knn-failure-cases-limitations-and-strategy-to-pick-right-k-45de1b986428.