Part 1: Research Question

A1) Research Question

Using a list of explanatory variables can we predict the chances of a customer churning?

A2) Objectives and Goals

The goal of this analysis is to be able to use the prediction to determine which services have the biggest impact on the likelihood of a customer churning and using that information stakeholders can make decisions to keep customers subscribed for as long as possible.

Part 2: Method Justification

B1) Assumptions of Multiple Regression Model

The assumptions of a multiple linear regression model include:

* That it is based on the Bernoulli distribution rather than the Gaussian because the dependent variable is binary
* That the predicted values are restricted to a range of nomial values like ‘Yes’ and ‘No’
* That it predicts the probability of particular outcomes rather than the outcome itself
* That it is the logarithm of the odds of achieving 1

B2) Benefits of Tools

For this analysis I will be using Python for the coding and Jupyter notebook as the IDE.

While R is a powerful language for data analysis, Python is a more versatile language considering it is primarily an object-oriented language. For me personally, when starting out with programming I worked primarily with C# and Java, as I am sure many other people do. Using Python feels more natural coming from a background using other object-oriented languages, and Python has many different libraries and packages that are making it one of the strongest tools for data analysis. Jupyter notebook is also simple to use, as it runs right in the browser and has a clean user interface. [6]

B3) Appropriate Technique

Logistic regression is the appropriate technique to use for the analysis of my research question because the response variable I am predicting is a categorical value of ‘Yes’ and ‘No’, which will be encoded to the binary 1 and 0.

C1) Describe Data Goals

The goal of the data preparation is to manipulate the data to better fit using logistic regression.

To achieve this goal, first I will load the data set into python and evaluate the data to gain a better understanding of the data set. Normally the first part of my manipulation would be to re-name the final 8 survey questions for better visibility, but since those are going to be dropped as less meaningful, I am going to skip that part and go right to dropping all the less meaningful columns from the data set. The next step is to check the data for any missing or null values, should any be found, I will impute the missing records using either the mean, median, or mode depending on the type of data that specific variable is. Any column that has categorical variables of ‘yes’ or ‘no’ will have their values replaced by 1 and 0. Out of the remaining columns of categorical values, for ‘Contract’, ‘InternetService’, and ‘Gender’ I will use the pandas method get\_dummies to split those columns into 0’s and 1’s. The remaining columns with categorical values will be dropped, such as the columns ‘City’, ‘State’, and so on. Finally, I will create some univariate and bivariate graphs for visualization before saving the prepared data set.

C2) Discuss Summary Statistics

When loaded into pandas, we see that the data set consists of 50 columns each with 10,000 records. For this analysis, my target variable is the categorical variable ‘Churn’. As for the predictor variables, I dropped the less meaningful columns that likely would not have any effect on our response variable. These columns include less meaningful data and customer demographic data such as:

* CaseOrder
* Customer\_id
* Interaction, UID
* City, State, Country, Zip, Lat, Lng
* Population, Area
* TimeZone
* Job
* Marital Status
* Email (number of emails sent to customer)
* Contacts (number of times customer contacted tech support)
* Techie (whether the customer considers themselves technically inclined)
* PaperlessBilling
* Port\_modem
* PaymentMethod
* All 8 survey questions

When looking through the data set, I found that it appeared to be clean with no missing data. Finally, all remaining categorical values with fields of ‘yes’ and ‘no’ were replaced by 1 and 0. The variables ‘Contract’, ‘InternetService’, and ‘Gender’ all had 3 fields and so I used the pandas method get\_dummies to properly turn those columns variables to 0’s and 1’s since they are not in any specific order, along with the drop\_first = True modifier to get n-1 columns, then the original columns will be dropped. This leaves the remaining columns of:

* Children
* Age
* Income
* Gender (split with dummy values)
* Churn (target variable)
* Outage\_sec\_perweek
* Yearly\_equip\_failure
* Contract (split with dummy values)
* Tablet
* InternetService (split with dummy values)
* Phone
* Multiple
* OnlineSecurity
* OnlineBackup
* DeviceProtection
* StreamingTV
* StreamingMovies
* Tenure
* Bandwidth\_GB\_Year

C3) Data Preparation Steps

* Load the data set into pandas dataframe
* Examine the data set
* Drop less meaningful columns
* Search for missing or null values
* Impute missing fields with mean, median, or mode if necessary
* Replace categorical values with numeric values
* Create dummy variables for the three columns listed above
* Create histograms for univariate visualizations
* Create regplots for bivariate visualizations
* Extract the now prepared data set as ‘churn\_prepared.csv’

C3) 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 statsmodels.api as sm*

*from scipy import stats*

*import sklearn*

*from sklearn import preprocessing*

*from sklearn.linear\_model import LogisticRegression*

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

*from sklearn import metrics*

*from sklearn.metrics import classification\_report*

*#Load the data set into Pandas*

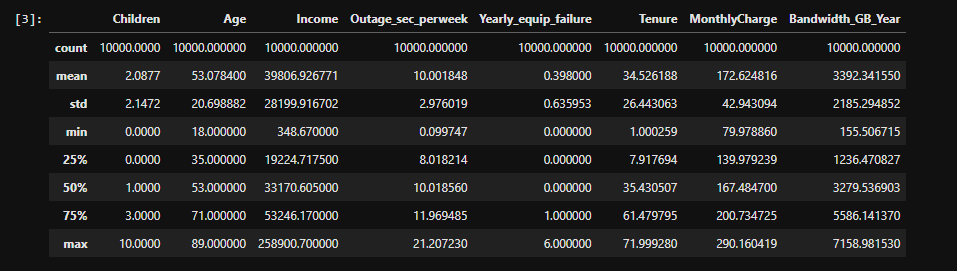
*df = pd.read\_csv('churn\_clean.csv')*

*df.describe()*

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

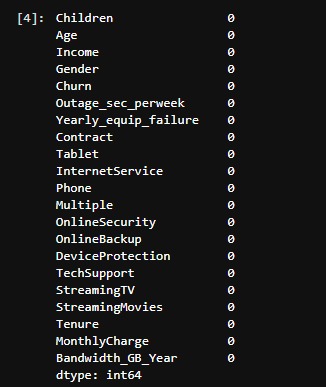
*df = df.drop(columns = ['CaseOrder', 'Customer\_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job', 'Marital', 'Email', 'Contacts', 'Techie','PaperlessBilling', 'Port\_modem', 'PaymentMethod', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'])*

*df.describe()*

**

*#Search for missing data*

*df.isnull().sum()*

**

*#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['Tablet\_num'] = df['Tablet']*

*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\_tablet = {"Tablet\_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}}*

*#Replace the variables values*

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

*df.replace(dict\_tablet, 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)*

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

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

*#df.info()*

*#Now we use dummies and one-hot encoding for the categorical variables with n-levels*

*contract = pd.get\_dummies(df['Contract'], drop\_first = True)*

*internet = pd.get\_dummies(df['InternetService'], drop\_first = True)*

*gender = pd.get\_dummies(df['Gender'], drop\_first = True)*

*df = df.join(contract)*

*df = df.join(internet)*

*df = df.join(gender)*

*df.info()*

Here I’m using drop\_first = True for the dummy values so that I can get n-1 columns.

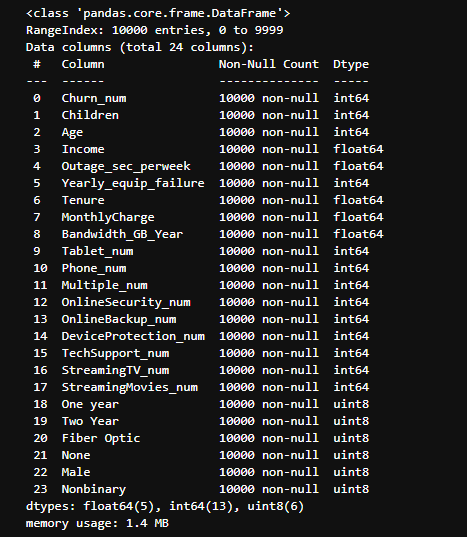
*#Drop original columns*

*df = df.drop(columns = ['Contract', 'InternetService', 'Gender'])*

*#Move response variable to the front*

*df = df.set\_index('Churn\_num').reset\_index()*

*df.info()*

**

C4) Univariate and Bivariate Visualizations

For the univariate visualizations I made histograms for the numeric variables. For the bivariate visualizations I made regplots of several variables with my target variable, Churn, on the y-axis of each plot.

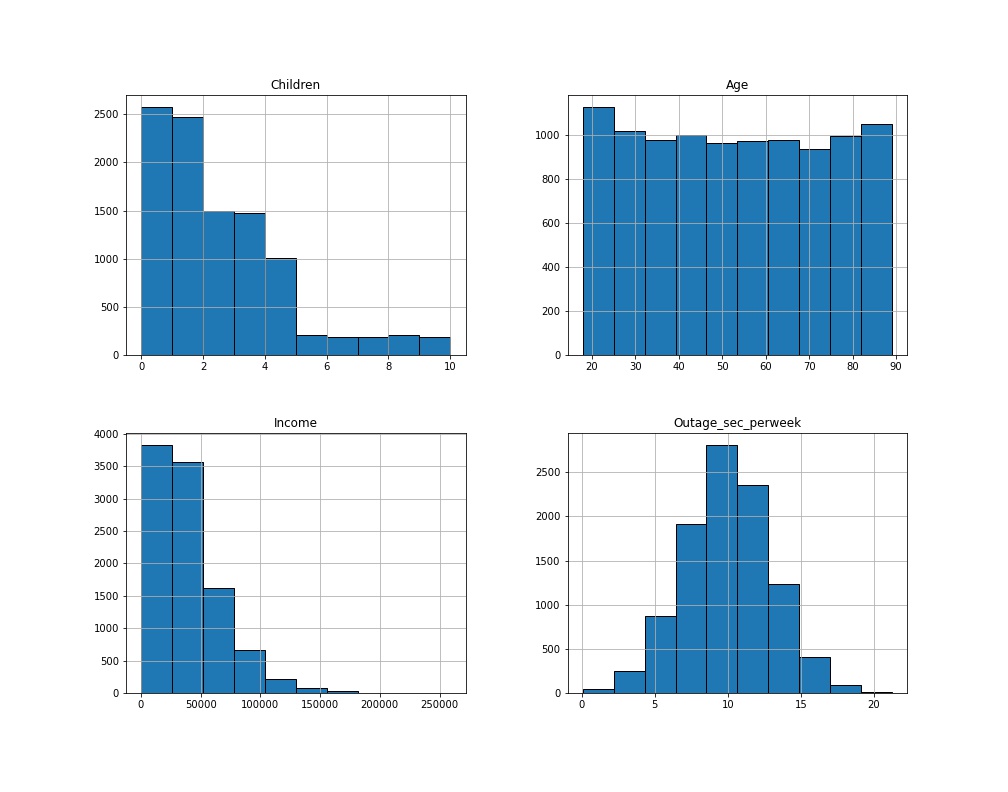
Also, for the bivariate visualizations I decided to make a subset of a random sampling of 25% of the data set so the plots have better visualizations.

C4) Code

*#For univariate statistics, create histograms for the continuous and categorical variables*

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

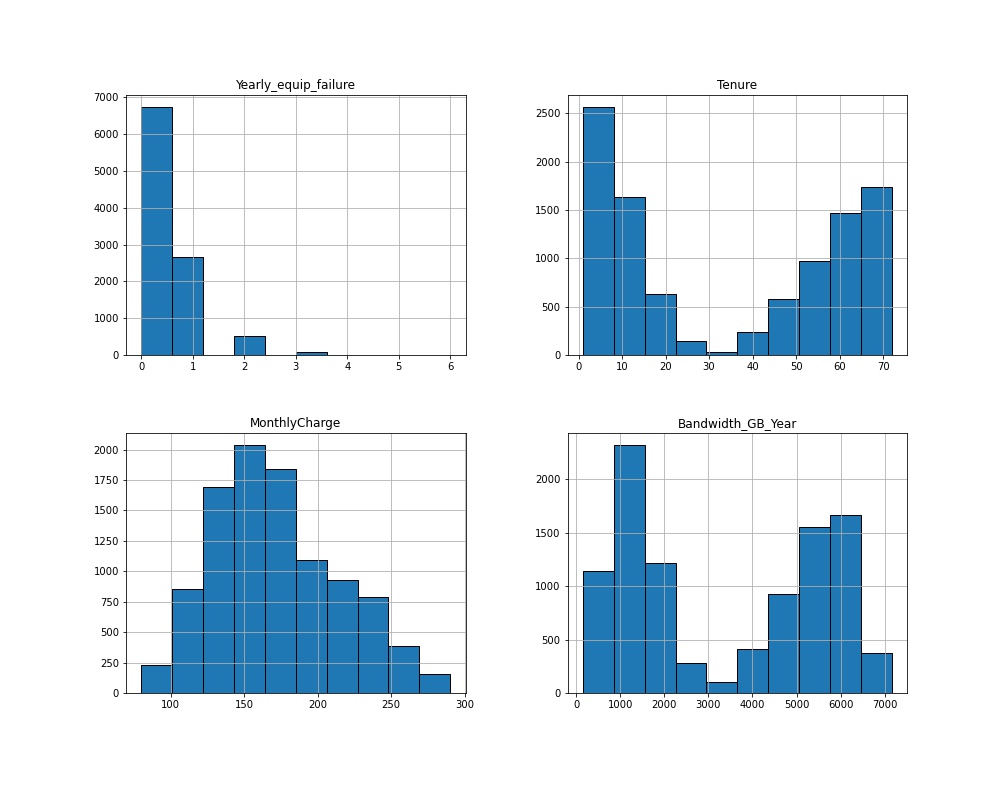
*#plt.savefig('Histogram1.jpg')*

**

*#For univariate statistics, create histograms for the continuous and categorical variables*

*df[['Yearly\_equip\_failure', 'Tenure', 'MonthlyCharge', 'Bandwidth\_GB\_Year']].hist(ec = "black", figsize = (14, 11))*

*#plt.savefig('Histogram2.jpg')*

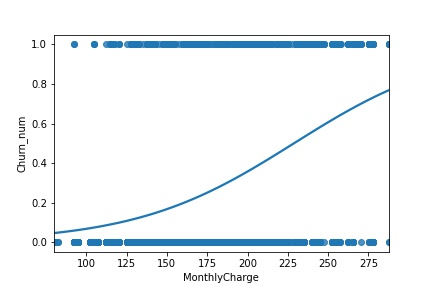
**

*#For bivariate statistics create some scatterplots with a few variables with our response variable as the y-axis*

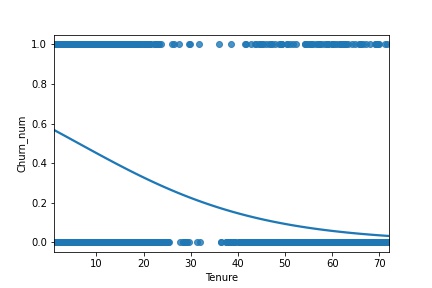
*#First, creating a random sampling of 25% of the data set for the scatterplots to improve visibility[3]*

*subset = df.sample(frac = 0.25)*

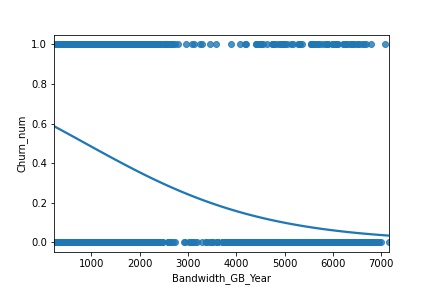
*ax = sns.regplot(x = "MonthlyCharge", y = "Churn\_num", data = subset, logistic = True, ci = None)*

**

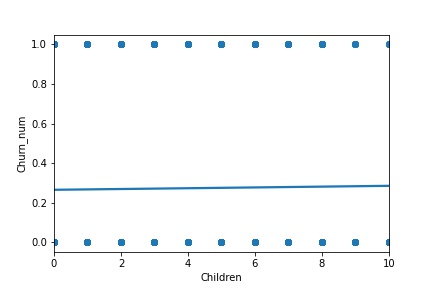
*ax = sns.regplot(x = "Tenure", y = "Churn\_num", data = subset, logistic = True, ci = None)*

**

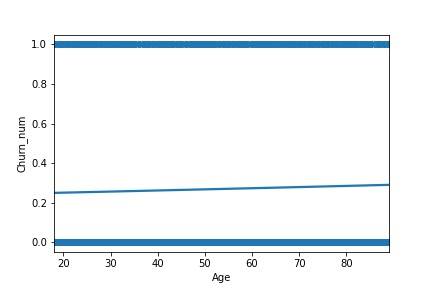
*ax = sns.regplot(x = "Bandwidth\_GB\_Year", y = "Churn\_num", data = subset, logistic = True, ci = None)*

**

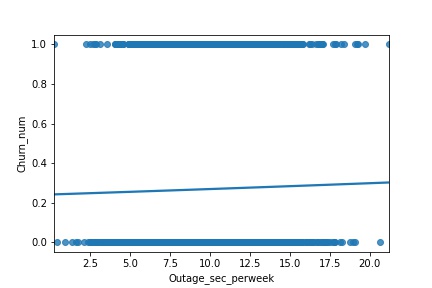
*ax = sns.regplot(x = "Children", y = "Churn\_num", data = subset, logistic = True, ci = None)*

**

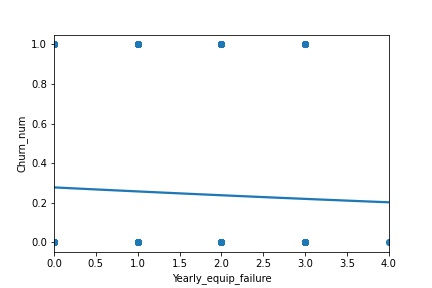
*ax = sns.regplot(x = "Age", y = "Churn\_num", data = subset, logistic = True, ci = None)*

**

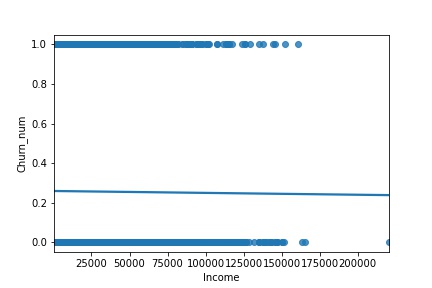
*ax = sns.regplot(x = "Outage\_sec\_perweek", y = "Churn\_num", data = subset, logistic = True, ci = None)*

**

*ax = sns.regplot(x = "Yearly\_equip\_failure", y = "Churn\_num", data = subset, logistic = True, ci = None)*

**

*ax = sns.regplot(x = "Income\_num", y = "Churn\_num", data = subset, logistic = True, ci = None)*

**

C5) Provide Copy of the Prepared Data Set

*#Extract prepared dataset*

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

Part 4: Model Comparison and Analysis

D1) Construct Initial Logistic Regression Model

*#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 statsmodels.api as sm*

*from scipy import stats*

*import sklearn*

*from sklearn import preprocessing*

*from sklearn.linear\_model import LogisticRegression*

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

*from sklearn import metrics*

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

*#Load the data set into Pandas*

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

*df.describe()*

*#Prepare data for train test split [4][5]*

*feature = ['Children', 'Age', 'Income', 'Outage\_sec\_perweek', 'Yearly\_equip\_failure', 'Tenure', 'MonthlyCharge', 'Bandwidth\_GB\_Year', 'Tablet\_num', 'Phone\_num', 'Multiple\_num', 'OnlineSecurity\_num', 'OnlineBackup\_num', 'DeviceProtection\_num', 'TechSupport\_num', 'StreamingTV\_num', 'StreamingMovies\_num', 'One year', 'Two Year', 'Fiber Optic', 'None', 'Male', 'Nonbinary']*

*X = df[feature]*

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

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

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

*#Run logistic regression model and make prediction [4]*

*model = LogisticRegression(solver = 'liblinear', random\_state = 0)*

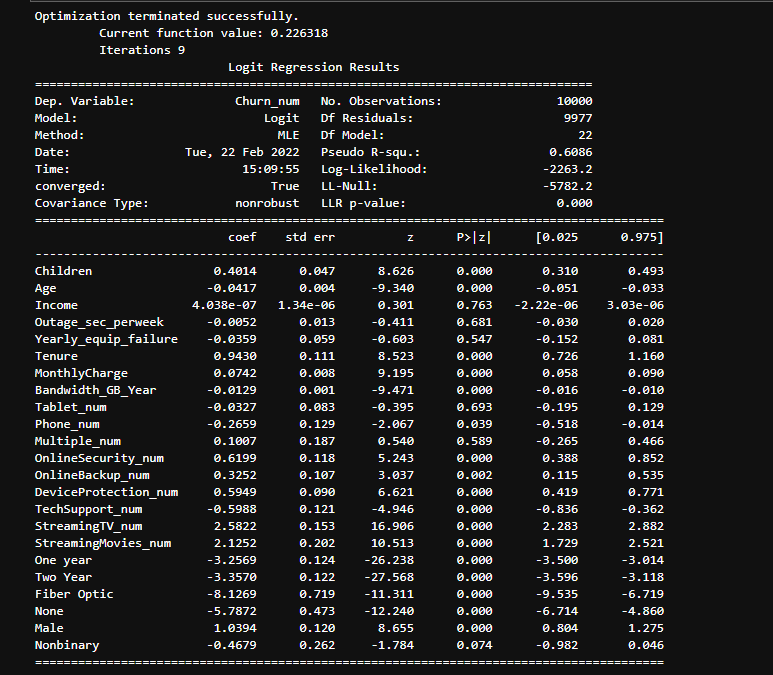
*model.fit(X\_train, y\_train)*

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

*#Run initial model with logit*

*initial\_model = sm.Logit(y, X).fit()*

*print(initial\_model.summary())*

**

D2) Selection Procedure

Note: For the following some of the code is being put slightly out of order to better discuss in the actual paper, if needed I am uploading a word document with strictly code alongside the assessment.

For the initial variable selection, we can look at the summary and disregard the variables with p-values above 0.05. In addition, we can use some methods to evaluate the model visually to get a better understanding of the correlations.

First thing is to create a confusion matrix, then use seaborn to create a heatmap of the confusion matrix to visualize it better.

*#Print out a confusion matrix [4]*

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

*print(cnf\_matrix)*

*#Use seaborn heatmap to visualize the confusion matrix*

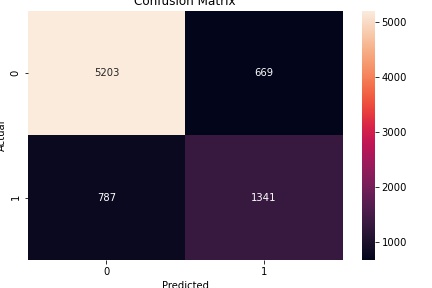
*sns.heatmap(pd.DataFrame(cnf\_matrix), annot = True, fmt = 'g')*

*plt.tight\_layout()*

*plt.title('Confusion Matrix')*

*plt.ylabel('Actual')*

*plt.xlabel('Predicted')*

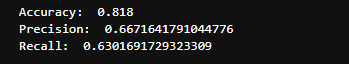
**

*#Display the accuracy, precision and recall of the model*

*print("Accuracy: ", metrics.accuracy\_score(y\_test, y\_pred))*

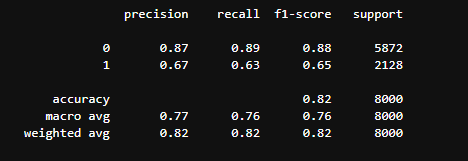
*print("Precision: ", metrics.precision\_score(y\_test, y\_pred))*

*print("Recall: ", metrics.recall\_score(y\_test, y\_pred))*

**

*#Print classification report*

*print(classification\_report(y\_test, y\_pred))*

**

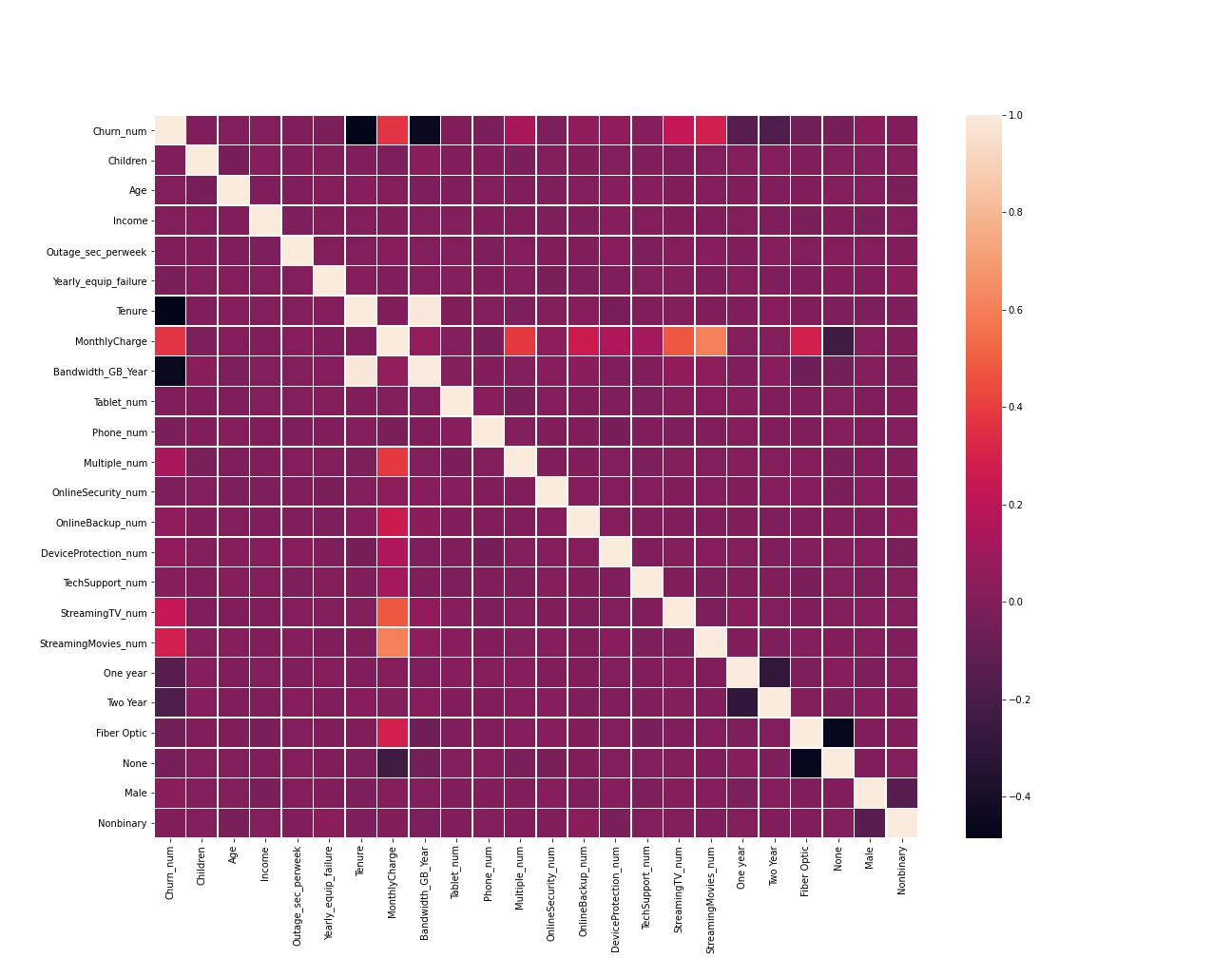
From the output listed above, the heatmap confusion matrix shows that the correct responses strongly outweigh the false positives and false negatives. From the printout it shows the accuracy of the model at 0.818, or roughly 81.80% so we can see that the initial model is fairly accurate. To further narrow down the number of variables that will be used in the reduced model I created a correlation heatmap to see which variables have the strongest interaction with the target variable ‘Churn’.

*#Create correlation heatmap for better visualization of model reduction and variable selection*

*plt.figure(figsize = (18, 14))*

*sns.heatmap(df.corr(), linewidth = 0.5)*

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

**

D3) Reduced Regression Model

Using the p-values from the initial summary and the correlation heatmap I was able to reduce the number of variables to use from 20 down to 8.

*#Prepare for reduced model*

*feature2 = ['Tenure', 'MonthlyCharge', 'Bandwidth\_GB\_Year', 'Multiple\_num', 'StreamingTV\_num', 'StreamingMovies\_num', 'One year', 'Two Year']*

*X2 = df[feature2]*

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

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X2, y, test\_size = 0.8, random\_state = 25)*

*#Run reduced model and make prediction*

*model2 = LogisticRegression(solver = 'liblinear', random\_state = 0)*

*model2.fit(X\_train, y\_train)*

*y\_pred2 = model2.predict(X\_test)*

Now to explore the data of the reduced model further using another confusion matrix to compare to the initial.

*#Display reduced confusion matrix*

*cnf\_matrix2 = metrics.confusion\_matrix(y\_test, y\_pred2)*

*print(cnf\_matrix2)*

*#Use seaborn for better visualization of reduced confusion matrix*

*sns.heatmap(pd.DataFrame(cnf\_matrix2), annot = True, fmt = 'g')*

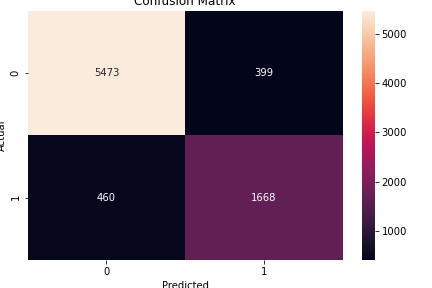
*plt.tight\_layout()*

*plt.title('Confusion Matrix')*

*plt.ylabel('Actual')*

*plt.xlabel('Predicted')*

*#plt.savefig('matrix2.jpg')*

**

From the confusion matrix we see that the values that are correct are better than in the initial model. Whereas before the true negative was 5203 the reduced model true negative is 5473. Likewise for the true positive being 1341 in the initial model and 1668 in the reduced. This also shows that the false negatives went down from 787 in the initial to 460 in the reduced, while the false positives went down from 669 in the initial to 339 in the reduced.

This can also be shown by printing out the accuracy, precision, and recall from both the initial and reduced models and compare them.

*#Display the accuracy, precision and recall of the model*

*print("Accuracy Initial: ", metrics.accuracy\_score(y\_test, y\_pred))*

*print("Accuracy Reduced: ", metrics.accuracy\_score(y\_test, y\_pred2))*

*print("")*

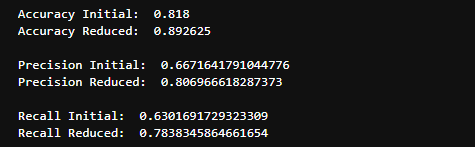
*print("Precision Initial: ", metrics.precision\_score(y\_test, y\_pred))*

*print("Precision Reduced: ", metrics.precision\_score(y\_test, y\_pred2))*

*print("")*

*print("Recall Initial: ", metrics.recall\_score(y\_test, y\_pred))*

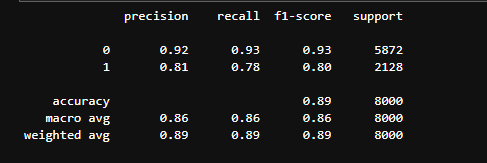
*print("Recall Reduced: ", metrics.recall\_score(y\_test, y\_pred2))*

**

This shows that the accuracy went up quite a bit, about 8%. However the other two went up a decent amount more, with the precious going up almost 14% and the recall almost 15%.

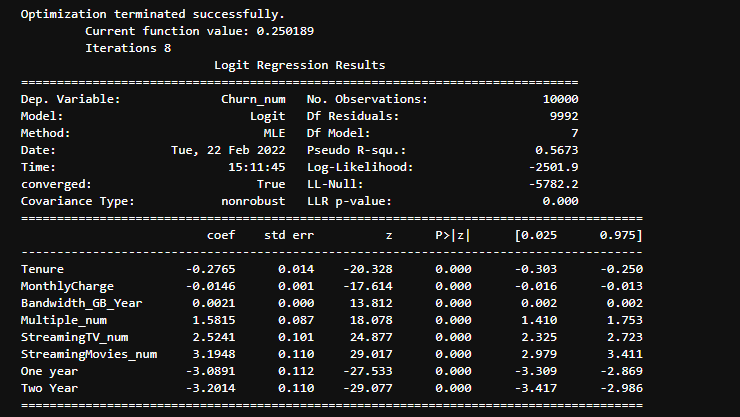
*#Print classification report*

*print(classification\_report(y\_test, y\_pred2))*

**

*#Run reduced model with logit*

*reduced\_model = sm.Logit(y, X2).fit()*

*print(reduced\_model.summary())*

E1) Process

As stated above, for variable selection I looked at the p-values from the initial model and at a correlation heatmap to reduce the number of variables that would be used for the reduced model. For the model evaluation I looked at the differences in the confusion matrix as well as the accuracy, precision and recall to see how much the reduced model improved the initial model. However, looking at the summary statistics the pseudo R^2 value in the initial model is 0.6086 whereas in the reduced it is 0.5673. This would suggest that the initial model is better at predicting an outcome, but the confusion matrix and accuracy/precision printouts show otherwise.

E2) Output

All output and confusion matrices are listed above

E3) Code

All code used for the logistic regression is listed above as well as in the code word document uploaded with this assessment

Part 5: Data Summary and Implications

F1) Discuss Results

The regression equal for the reduced model is as follows:

Y = -0.28(Tenure) – 0.014(MonthlyCharge) + 0.0021(Bandwidth\_GB\_Year) + 1.58(Multiple\_num) + 2.52(StreamingTV\_num) + 3.19(StreamingMovies\_num) – 3.09(One year) – 3.20(Two Year)

F1.1) Interpretation of Coefficients

The coefficients are a multiplier of the predictor variable either positively or negatively, with a positive value leaning towards churn while a negative value leans against churn.

For the positive coefficients it appears that the customers who have multiple devices, streaming TVs and that stream movies often are at the highest risk of churn.

For the negative coefficients it shows that a customer’s tenure, monthly charge, and whether they are in a one- or two-year contract have a negative effect on whether a customer will churn.

While still a positive value, the bandwidth used per year is low, so I feel it is probably safe to say that it does not have too strong of an impact either way, only slightly leaning toward churn.

F1.2) Limitations

The main limitation of this data analysis is that our data set is only a representative of the entire population, and this specific data set is only 10,000 records of a specific window of time.

F2) Recommendation

While churn will always happen, the strongest variable for customers churning is how much they stream movies, with the second being if they have a streaming tv. Put together with the very slight positive value for bandwidth used per year one recommendation could be that if customers are churning because they hit a data cap from excessive streaming, increasing the bandwidth cap could keep customers happy.

Part 6: Demonstration

G) Panopto Video

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=c3bfafe6-a6d2-430f-97a0-ae44014b6040>

H) Sources for Third-Party Code

[1] “Seaborn.heatmap.” *Seaborn.heatmap - Seaborn 0.11.2 Documentation*, https://seaborn.pydata.org/generated/seaborn.heatmap.html.

[2] Carvalho, Thiago. “Heatmap Basics with Python's Seaborn.” *Medium*, Towards Data Science, 24 Sept. 2021, https://towardsdatascience.com/heatmap-basics-with-pythons-seaborn-fb92ea280a6c.

[3] Duca, Angelica Lo. “How to Sample a Dataframe in Python Pandas.” *Medium*, Towards Data Science, 7 July 2021, https://towardsdatascience.com/how-to-sample-a-dataframe-in-python-pandas-d18a3187139b.

[4] Navlani, Avinash. “Python Logistic Regression with Sklearn & Scikit.” *DataCamp Community*, https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python.

[5] [5] Larose, C. D., & Larose, D. T. (2019). Chapter 11: Regression Modeling. In *Data Science using python and R*. essay, Wiley Blackwell.

J) Sources

[6] “Python vs. R: What's the Difference?” *IBM*, https://www.ibm.com/cloud/blog/python-vs-r