Predictive Modeling of Telecommunications Churn Data with Logistic Regression

Jeffrey Van Anderson

<u>Part 1: Research Question:</u>

1.A

When inspecting telecommunications churn data briefly, one specific question arises before all others: What variables influence customer churn most? However, this type of question only retroactively asks why customers have churned in the past. Perhaps a question centered around prediction is more useful. Consequentially, the **research question** this analysis strives to answer is: Is it possible to predict which telecommunications customers will churn and which will not from the telecommunications dataset? If so, which selection of predictor variables is most parsimonious in predicting 'Churn'?

1.B

For a deliverable result from this analysis, several goals must be met. First, the telecommunications 'Churn' dataset must be prepared for building and testing a logistic regression model. This preparation includes pivoting categorical variables into binary indicator dummy levels, encoding binary variables as "1" and "0", and ensuring there are no null values or duplicate rows for any variable. Next, an initial logistic regression model with every variable must be inspected for numerical issues and statistical significance. Then, a reduced with a most parsimonious selection of variables is compared against it. Both these models are trained on 70% of the data, evaluated against 30% of the data, and evaluated with a crosstabulation table. The ultimate goal of this analysis is to create parsimonious model that accurately predicts customer 'Churn' with a selection of statistically significant predictor variables.

(Massaron, 2016)

Part II: Method Justification:

2.A

To trust the interpretation of a logistic regression's classifications and subsequent predictions, several assumptions must be met. First, for the binary logistic regression analysis used for the prediction of 'Churn', it is assumed that 'Churn' is a binary variable and may be classified and predicted as such. Second, each observation in the dataset is independent. If this is held true, each observation can be interpreted and included in either the model training or testing. Third, each predictor variable used in the logistic regression model must be independent and certainly not multicollinear. (Stoltzfus, 2011) Multicollinear variables lead to instability of coefficients, which reduces their significance and interpretability. Further, these redundant variables add unnecessary complexity to a model. A fourth assumption for a logistic regression model respectively is that there is roughly a linear relationship between the logit transformation of the target variable and each of the continuous predictor variables. This way, variables of the continuous datatype may be reasonably be used in a model to explain variance. Finally, the data must be assumed to have no strongly influential outliers. It must be assumed that there are no data outliers that reduce model accuracy or change the relationship between the predictor and response variables significantly more than other observations. If each of the five assumption above are met for a logistic regression analysis, then the results of a predictive analysis may be reasonably interpreted.

(Stoltzfus, 2011)

2.B

To effectively perform a logistic regression analysis, it is important to select an efficient and robust tool. For this analysis, the Python programming language is used within the Jupyter Notebook interactive development web application. The major benefit of using the Jupyter Notebook programming interface is for documentation and reproducibility. (Project Jupyter, 2020) With a Jupyter notebook, each section of code may be organized into a "cell". Each cell then has the capability of being run independently with output displayed after. As a result, code generated for a multiple regression analysis is easily interpreted. The main benefit of using the Python programming language for logistic regression analysis is the availability of extensive packages developed by an open-sourced community. (Python, 2020) For example, the "Pandas" package uses the efficiency of vectors and their respective arrays to perform highly efficient transformations with large amounts of data. (Pandas, 2020) In turn, the process of preparing data for analysis is highly efficient even on computers with limited resources. Another example of a useful package is the "SciKit Learn" library. (SciKit Learn, 2020) This package includes many prebuilt algorithms including logistic regression for the training, testing and visualization of predictive models. In summary, Python is an adequate tool for this logistic regression analysis due to its robust open-sourced nature.

2.C

As the research question is targeted at predicting a binary response variable with multiple input variables of various datatypes, an appropriate tool for analysis is the logistic regression model. Since the logistic model fits a sigmoid function against a binary outcome in this case, the classification of a "1" or "0" for each customer 'Churn' observation may be calculated from a selection of predictor variables. Subsequently, the logistic regression model produced may be used to predict customer ‘Churn’ on a subset of data where ‘Churn’ is unknown. Though there are many options for models to predict a binary response variable, logistic regression is a simple, interpretable, and logical first choice.

(Scikit Learn, 2020)

Part III: Data Preparation:

3.A

To prepare the telecommunications churn data for analysis, specific data preparation goals must be met. One goal is to ensure that the dataset contains only relevant predictor variables and a single response variable, ‘Churn’. This is achieved by removing all unique reference variables and those categorical variables which have clearly exhaustive levels. This is performed by using the “Pandas.drop()” function for any column that fits the aforementioned criteria. (Pandas, 2020) Another data preparation goal is that each predictor variable in the analysis must be interpreted as numeric. This goal must be met by using several variable transformations. First, each binary variable must be encoded as ‘1’ or ‘0’ rather than ‘yes’ or ‘no’ respectively. Second, categorical variables are transformed into indicator binary levels with one category removed. These resulting levels are also encoded as ‘1’ or ‘0’ for indication of that categorical level. Third, the datatype for every variable must be verified as numeric. A final, but perhaps most important goal for data preparation is to ensure that each observation in the analysis is unique and that there are no null values. This is achieved by first querying null counts for each variable in the telecommunications dataset using the “Pandas.info()” function. Second, the “Pandas.drop_duplicates()” function is used to remove any completely duplicated rows. (Pandas, 2020) The resulting dataset has no missing observations and is completely numeric, which makes analysis using logistic regression possible.’

3.B

To answer the research question with a logistic regression model, each predictor variable must be inspected and compared with ‘Churn’ with select **summary statistics**. To define which predictor variables to summarize in the analysis, the following categories of variables according to their datatypes are outlined. These categories exclude any reference variable or categorical variable with exhaustive levels.

- **Binary**: 'Techie', 'Port_modem', 'Tablet', 'Phone', 'Multiple', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBilling'
- **Categorical**: 'Area', 'TimeZone', 'Marital', 'Gender', 'Contract', 'InternetService', 'PaymentMethod'
- **Ordinal**: 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7'
- **Continuous**: 'Population', 'Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email', 'Contacts', 'Yearly_equip_failure', 'MonthlyCharge', 'Bandwidth_GB_Year', ‘Tenure’
- **Binary Target Variable**: ‘Churn’

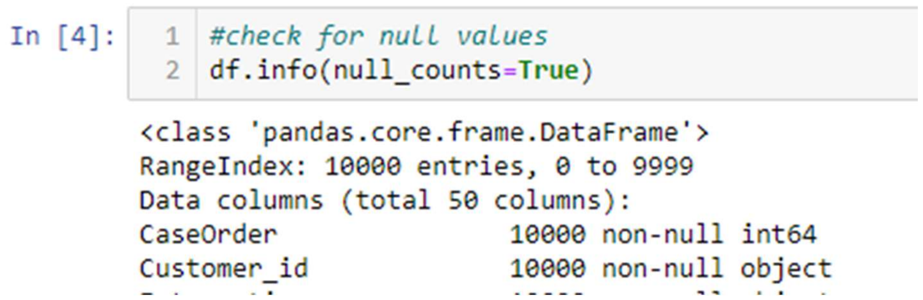
First, to preliminarily inspect the relationship of the binary target variable with ordinal, categorical, and binary levels, crosstabulation tables comparing standardized residuals for each variable versus ‘Churn’ are calculated. The standardized residual for each level of churn and the variable in question explain the magnitude of significance versus Churn. The further away from zero the standardized residual is, the more important the effect of the categorical, ordinal, or binary level is. (Pennsylvania, 2018) With this information, better understanding of the relationship between Churn and non-continuous variables is gathered. A next series of summary statistic that are used to inspect the initial logistic regression model with all predictor variables versus a logistic regression with reduced predictor variables are derived from the ‘Summary()’ function in the ‘Statsmodels.Logit’ library. (Statsmodels, 2020) This table importantly shows the ‘coef’ and ‘p-value’ for each variable and categorical level versus churn. The ‘coef’ helps explain the magnitude of importance for each variable versus churn, while the p-value explains how likely it is that the variable in question has no relationship at all with ‘Churn’. Generally, variables with p-values lower than 0.05 have a statistically significant relationship with ‘Churn’. The comparison of these summary statistics helps to determine whether a variable should be included in the logistic regression model or eliminated for the sake of a simple generalizing predictive model. Finally, to programmatically reduce the selected variables in the logistic regression model, an elimination of each predictive variable based their standardized coefficient contribution is calculated. This is calculated by multiplying each variable’s ‘coef’ value by that variable’s relative standard deviation with the rest of the model input variables. This is performed with each variable and categorical level in the model until the model with the highest AUC score or “area under the roc curve” is calculated. (Scikit Learn, 2020). The AUC score is a method of describing model accuracy. The closer the AUC score is to 1, the higher the predictive accuracy of the model. The higher the AUC score, the occurrences of false negatives and false positives are less frequent. Clearly, through the investigation and programmatic elimination of variables in this analysis, select summary statistics help build a model that answers whether it is possible to predict customer ‘Churn’ from a reduced selection of input variables.

(Glen, 2015)

3.C

To prepare this dataset for analysis with logistic regression, the following steps are sequentially taken:

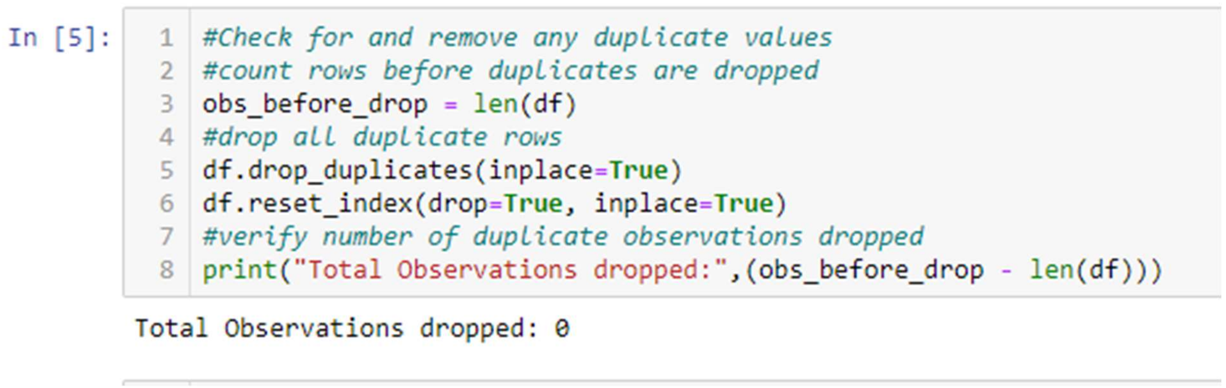
1. First to ensure that the provided dataset has no missing values or duplicated rows, two actions are taken. First the pandas '.info()' function is used to display the value counts for each variable in the data frame. Since each variable shows 10,000 non-null observations in the output, it is evident that the dataset has no null values. See the code sample below for the output generated:



```
In [4]:
#check for null values
df.info(null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
CaseOrder                10000 non-null int64
Customer_id              10000 non-null object
```

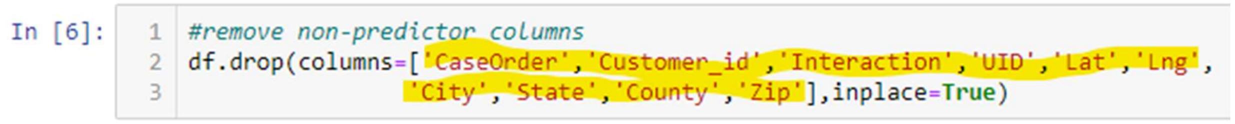
Next, the entire dataset is queried for duplicate rows. If there are any, they are completely removed. The difference between the length of the dataset before and after removal is calculated and printed after. Fortunately, there are not duplicate observations.



```
In [5]:
#Check for and remove any duplicate values
#count rows before duplicates are dropped
obs_before_drop = len(df)
#drop all duplicate rows
df.drop_duplicates(inplace=True)
df.reset_index(drop=True, inplace=True)
#verify number of duplicate observations dropped
print("Total Observations dropped:",(obs_before_drop - len(df)))
```

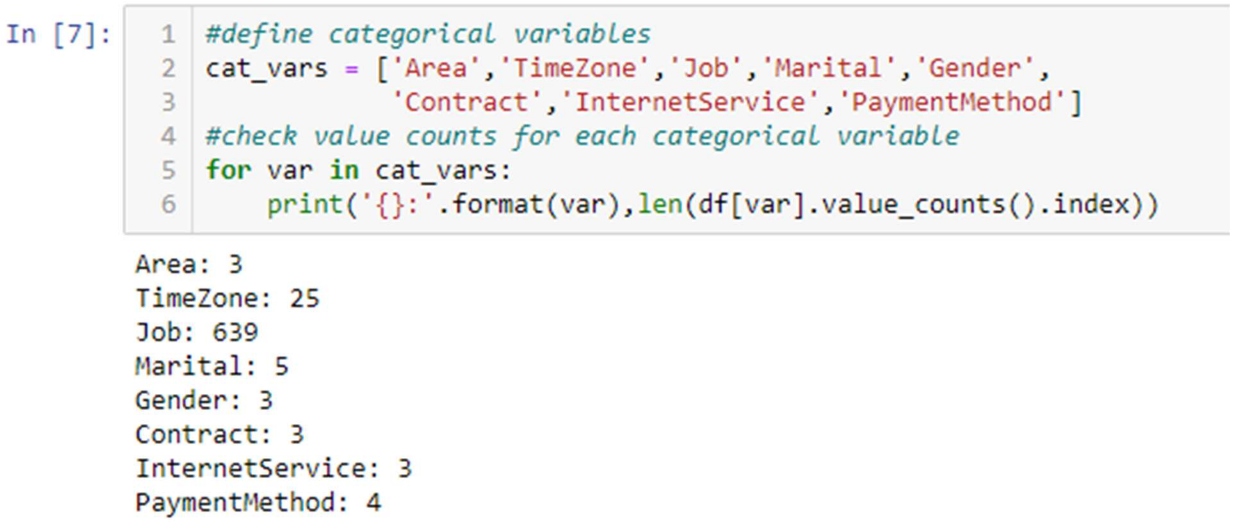
```
Total Observations dropped: 0
```

2. Second, any column that clearly cannot be used in analysis is removed. This includes unique key variables and any variable that has excessive levels, like 'Zip'. The ten variables removed in this step are highlighted in the code block below.



```
In [6]:  #remove non-predictor columns
         df.drop(columns=['CaseOrder','Customer_id','Interaction','UID','Lat','Lng',
                          'City','State','County','Zip'],inplace=True)
```

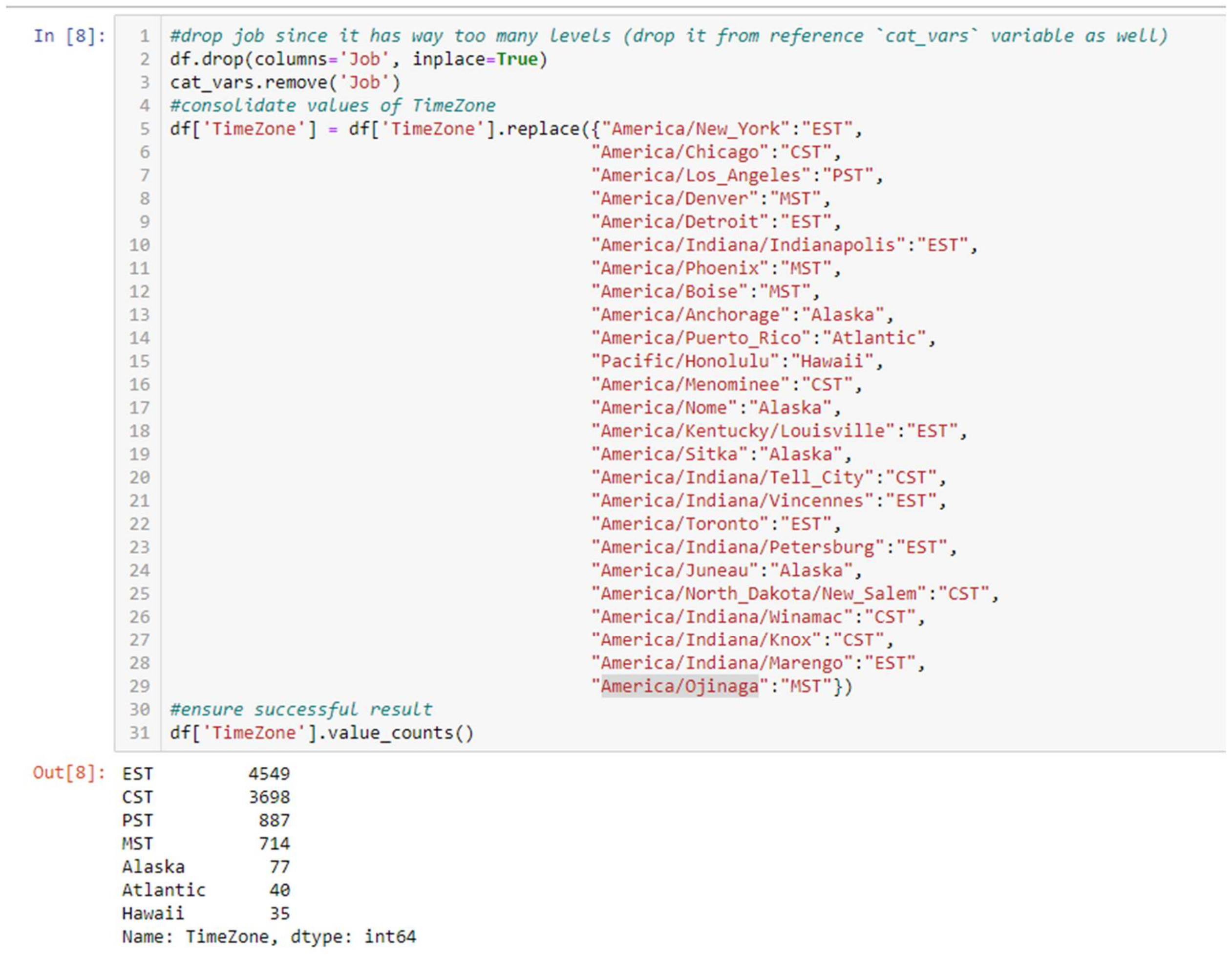
3. Third, each categorical variable in the dataset must be inspected for excessive or redundant levels prior to being included in a logistic regression model. Each categorical variable is queried for the number of unique levels in the following code block.



```
In [7]:  #define categorical variables
         cat_vars = ['Area','TimeZone','Job','Marital','Gender',
                     'Contract','InternetService','PaymentMethod']
         #check value counts for each categorical variable
         for var in cat_vars:
             print('{}:'.format(var),len(df[var].value_counts().index))

         Area: 3
         TimeZone: 25
         Job: 639
         Marital: 5
         Gender: 3
         Contract: 3
         InternetService: 3
         PaymentMethod: 4
```

Clearly, 'Job' has an extraordinary number of levels, and is not included in this analysis for the sake of simplicity and time. However, 'TimeZone' appears to have a high number of redundant levels. For example, the 'America/Denver' and 'America/Ojinaga' levels are clearly in the mountain time zone and may be reduces to 'MST'. Subsequently, the 25 levels of 'TimeZone' are reduced to a total of seven levels and 'Job' is removed in the following code block.

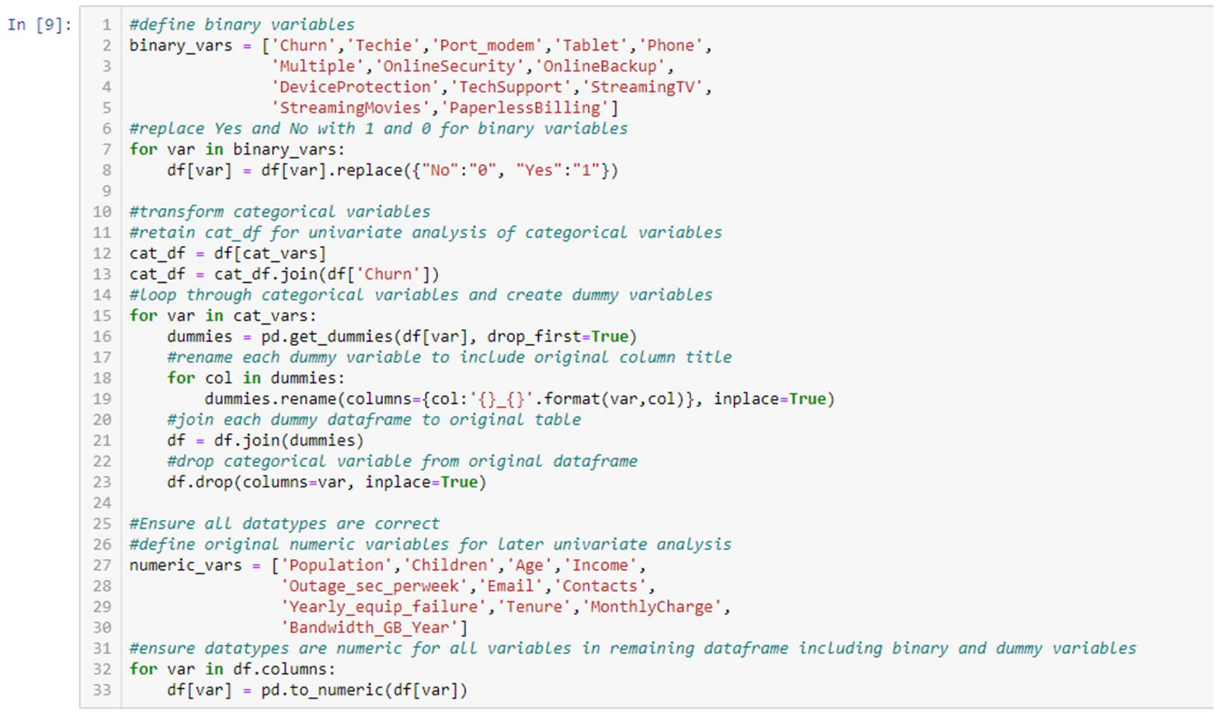


```
In [8]:
#drop job since it has way too many levels (drop it from reference `cat_vars` variable as well)
df.drop(columns='Job', inplace=True)
cat_vars.remove('Job')
#consolidate values of TimeZone
df['TimeZone'] = df['TimeZone'].replace({"America/New_York":"EST",
                                         "America/Chicago":"CST",
                                         "America/Los_Angeles":"PST",
                                         "America/Denver":"MST",
                                         "America/Detroit":"EST",
                                         "America/Indiana/Indianapolis":"EST",
                                         "America/Phoenix":"MST",
                                         "America/Boise":"MST",
                                         "America/Anchorage":"Alaska",
                                         "America/Puerto_Rico":"Atlantic",
                                         "Pacific/Honolulu":"Hawaii",
                                         "America/Menominee":"CST",
                                         "America/Nome":"Alaska",
                                         "America/Kentucky/Louisville":"EST",
                                         "America/Sitka":"Alaska",
                                         "America/Indiana/Tell_City":"CST",
                                         "America/Indiana/Vincennes":"EST",
                                         "America/Toronto":"EST",
                                         "America/Indiana/Petersburg":"EST",
                                         "America/Juneau":"Alaska",
                                         "America/North_Dakota/New_Salem":"CST",
                                         "America/Indiana/Winamac":"CST",
                                         "America/Indiana/Knox":"CST",
                                         "America/Indiana/Marengo":"EST",
                                         "America/Ojinaga":"MST"})
#ensure successful result
df['TimeZone'].value_counts()
```

```
Out[8]:
EST         4549
CST         3698
PST          887
MST          714
Alaska        77
Atlantic      40
Hawaii        35
Name: TimeZone, dtype: int64
```

4. Finally, prior to creating a logistic regression model, each variable in the dataset must be interpreted as numeric. For binary variables, this is achieved simply by encoding ‘Yes’ and ‘No’ as ‘0’ and ‘1’ respectively. For the categorical variables in the dataset, each of their levels must be transformed into ‘dummy’ indicator variables with one level removed. This means that each level, like the ‘MST’ level of ‘TimeZone’ is represented as its own variable with ‘1’ indicating that value and ‘0’ for all other values. One level is excluded for each of the categorical variables since it is mathematically redundant and may the interpreted as a reference for all the other levels. For the ordinal and continuous variables, they are simply ensured to be the numeric datatype along with all other

variables prior to inclusion in a logistic regression model. The following code block performs the entire transformation process for each of the four datatypes outlined above.



In [9]:
```python
#define binary variables
binary_vars = ['Churn','Techie','Port_modem','Tablet','Phone',
               'Multiple','OnlineSecurity','OnlineBackup',
               'DeviceProtection','TechSupport','StreamingTV',
               'StreamingMovies','PaperlessBilling']
#replace Yes and No with 1 and 0 for binary variables
for var in binary_vars:
    df[var] = df[var].replace({"No":"0", "Yes":"1"})

#transform categorical variables
#retain cat_df for univariate analysis of categorical variables
cat_df = df[cat_vars]
cat_df = cat_df.join(df['Churn'])
#loop through categorical variables and create dummy variables
for var in cat_vars:
    dummies = pd.get_dummies(df[var], drop_first=True)
    #rename each dummy variable to include original column title
    for col in dummies:
        dummies.rename(columns={col:'{}_{}'.format(var,col)}, inplace=True)
    #join each dummy dataframe to original table
    df = df.join(dummies)
    #drop categorical variable from original dataframe
    df.drop(columns=var, inplace=True)

#Ensure all datatypes are correct
#define original numeric variables for later univariate analysis
numeric_vars = ['Population','Children','Age','Income',
                'Outage_sec_perweek','Email','Contacts',
                'Yearly_equip_failure','Tenure','MonthlyCharge',
                'Bandwidth_GB_Year']
#ensure datatypes are numeric for all variables in remaining dataframe including binary and dummy variables
for var in df.columns:
    df[var] = pd.to_numeric(df[var])
```

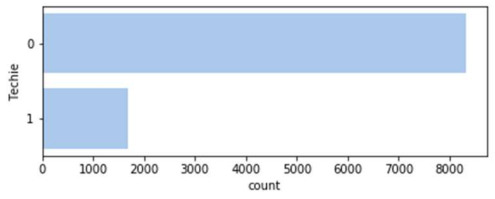
(Pandas, 2020)

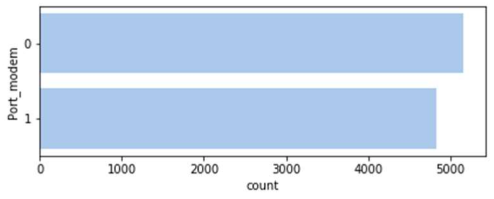
(Python, 2020)

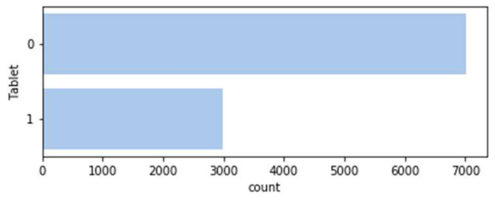
3.D

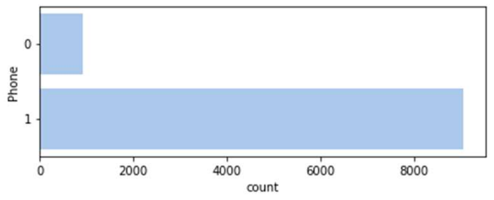
On the following pages, univariate visualizations for each predictor variable are displayed. Additionally, bivariate visualizations for each variable versus Churn are generated. (See the source code for reference) Note that each heatmap displays the standardized residual for categorical, binary, and ordinal levels versus churn. The further away from zero each value is, the intensity of red or blue is higher. The higher the intensity is for red or blue in each heatmap, the more influential that variable level is in analysis.

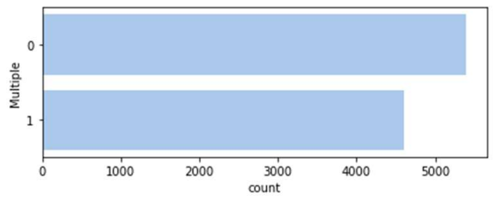
Univariate bar charts of each binary variable:

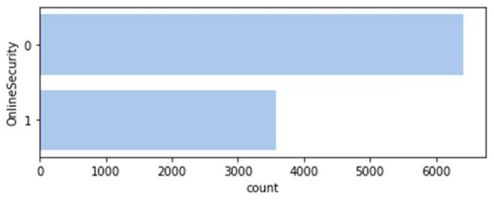


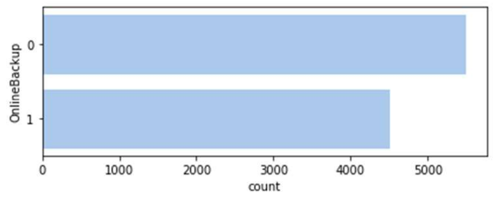


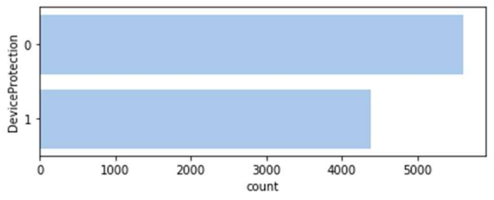


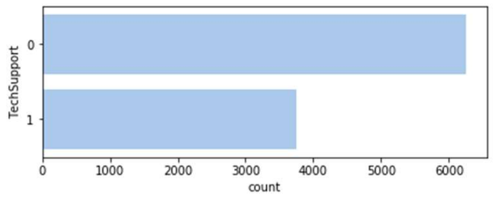


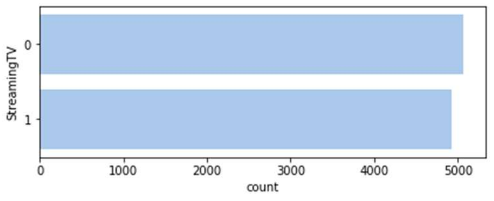


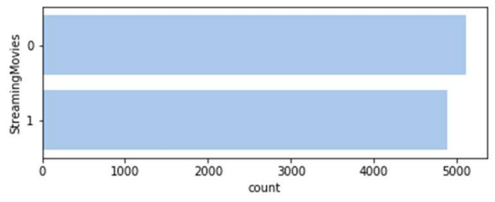


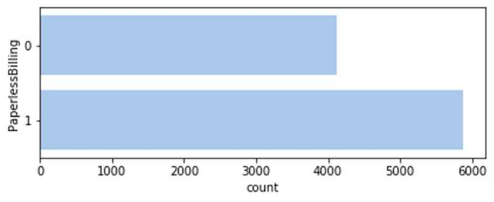


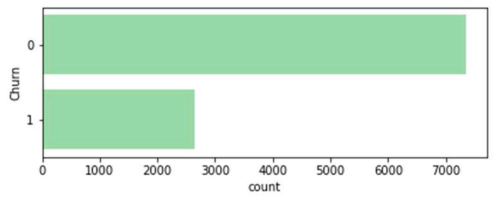




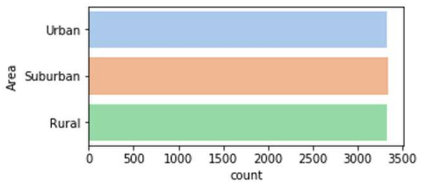


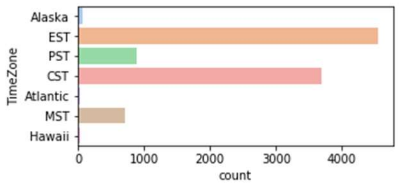


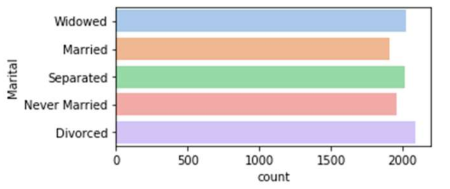


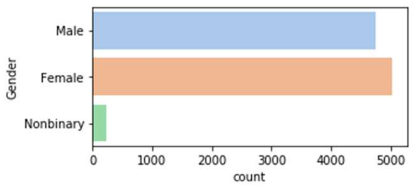


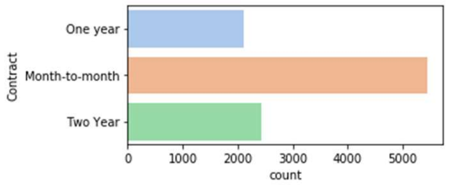
Univariate bar charts of each categorical and ordinal variable:

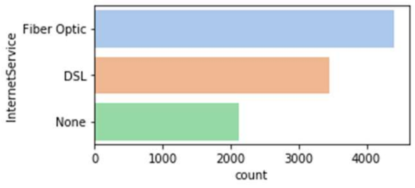


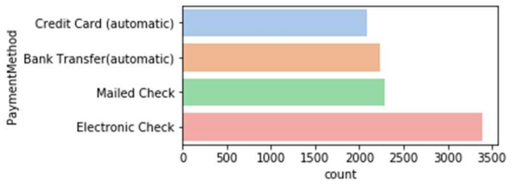


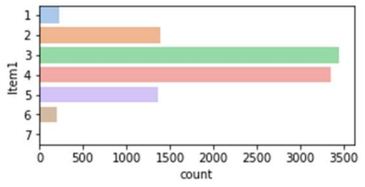


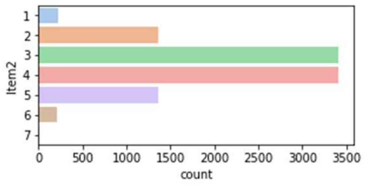


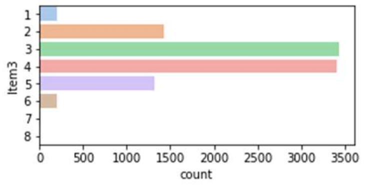


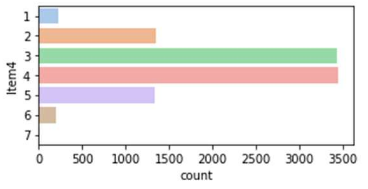


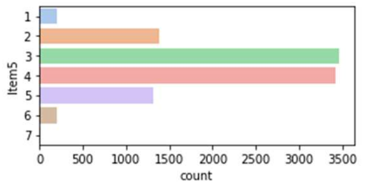


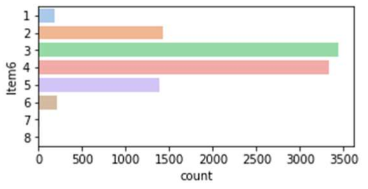


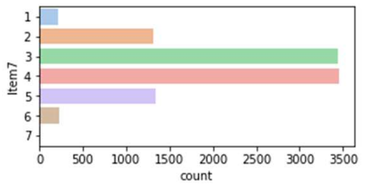


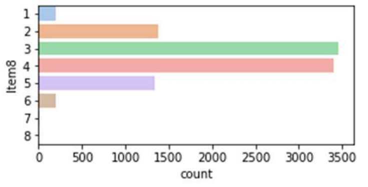




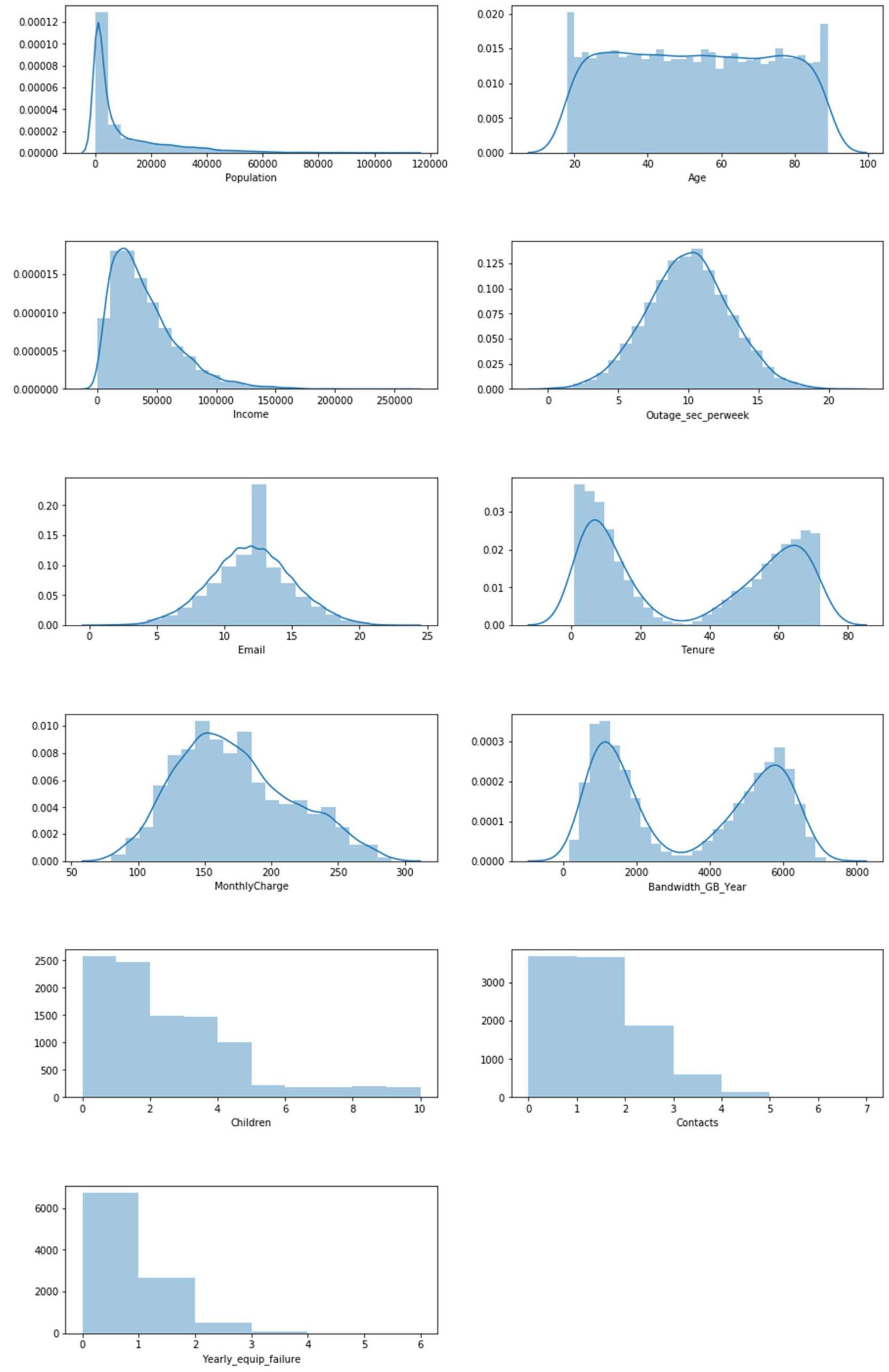




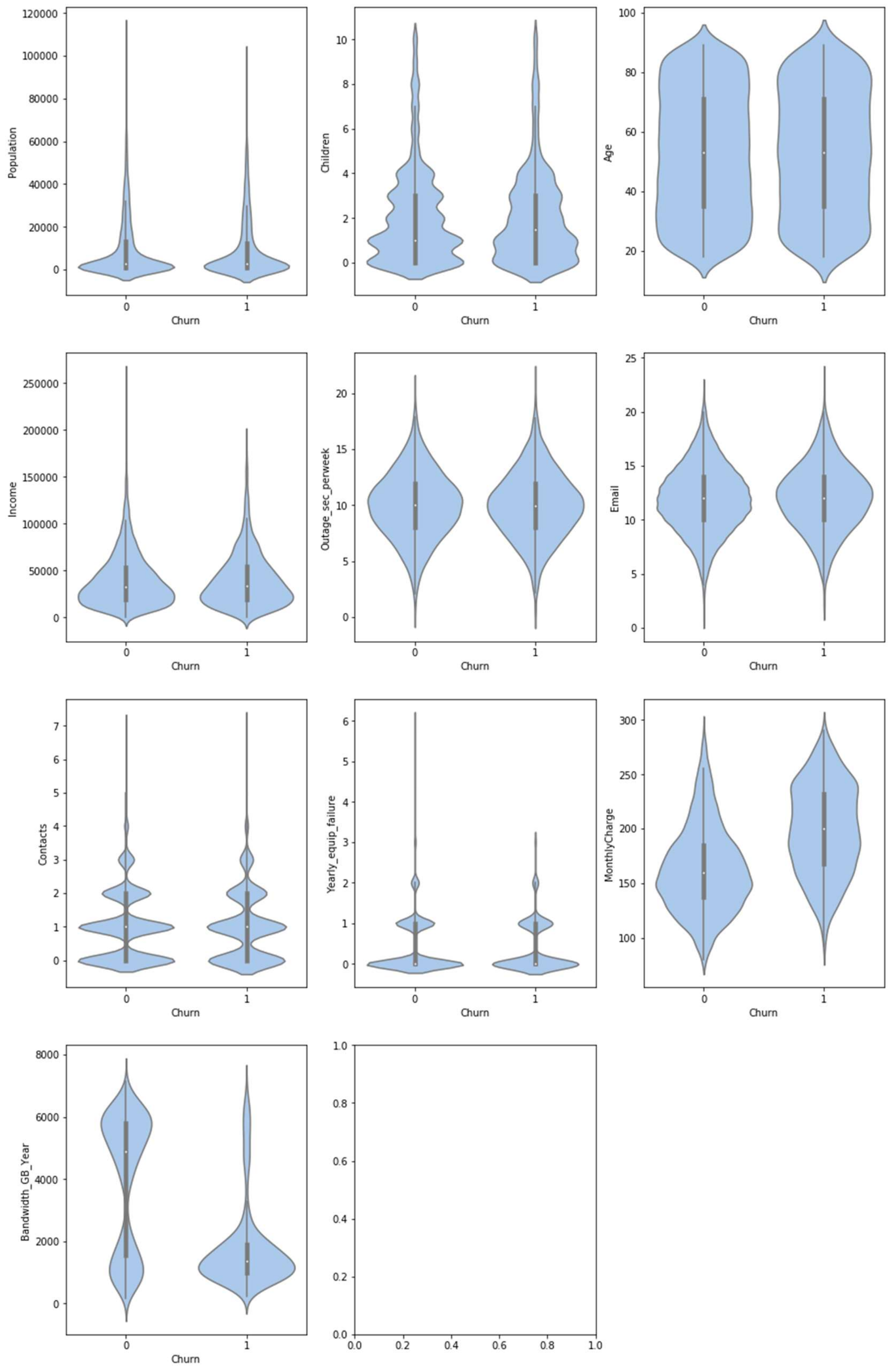




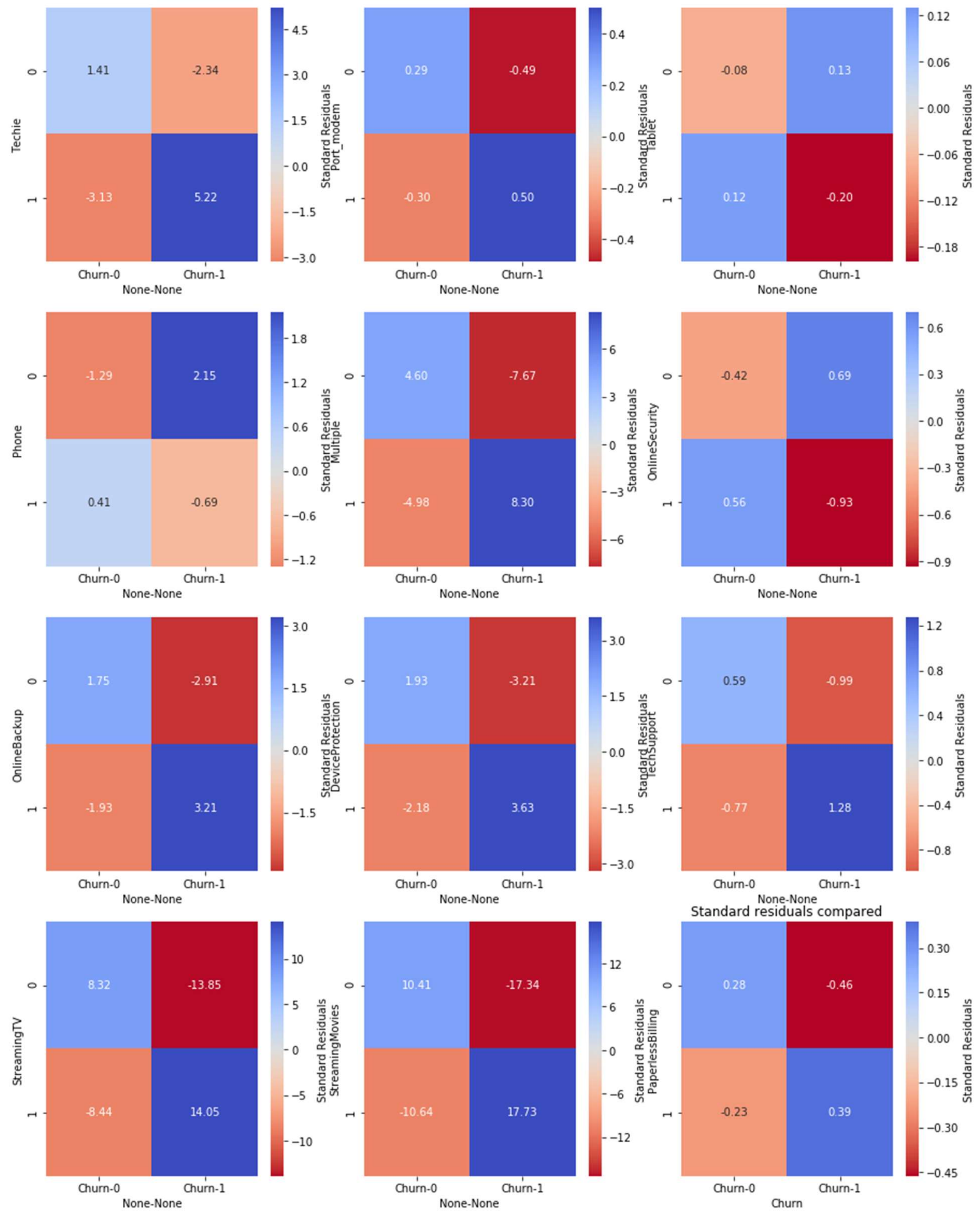
Univariate histograms of each numeric variable:



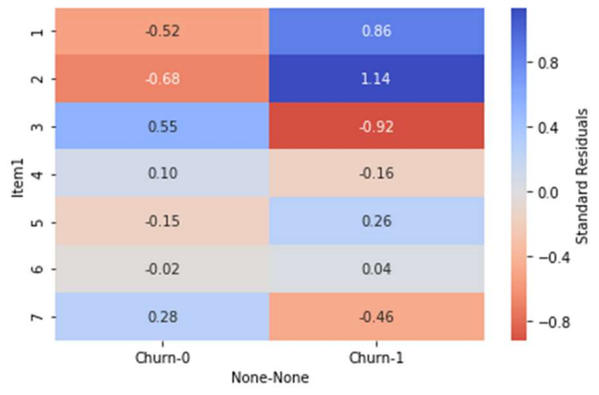
Bivariate violin plots of each numeric variable:

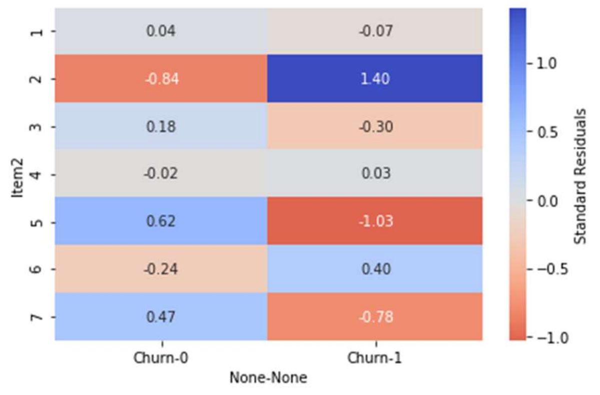


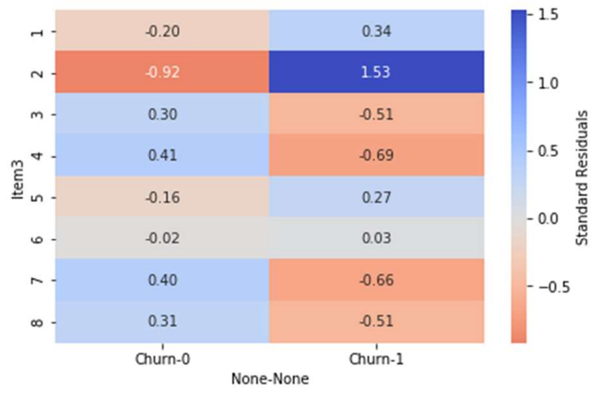
Bivariate standard residual heatmaps of each binary variable:

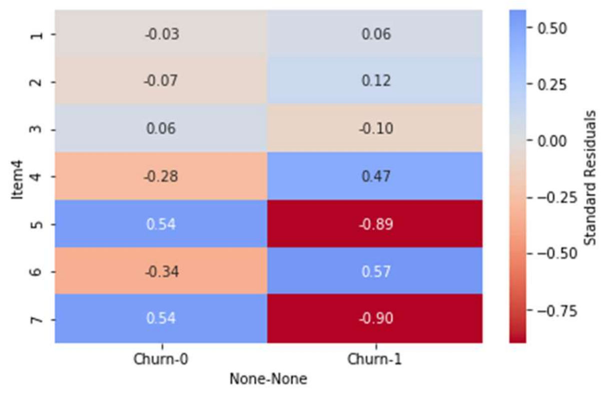


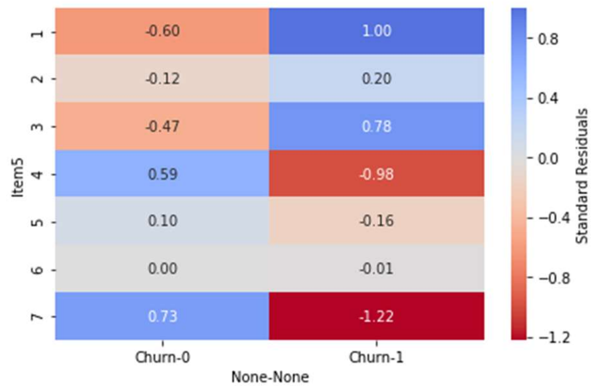
Bivariate standard residual heatmaps of each ordinal variable:

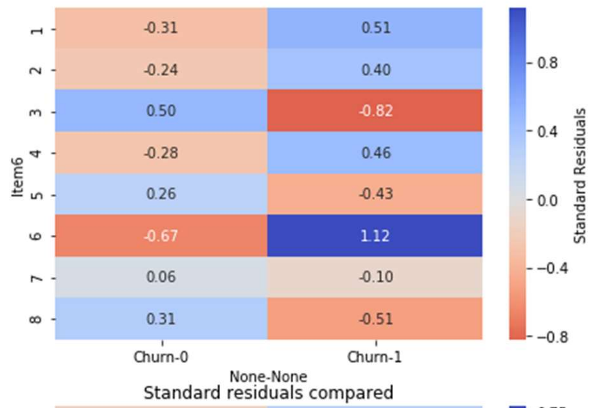


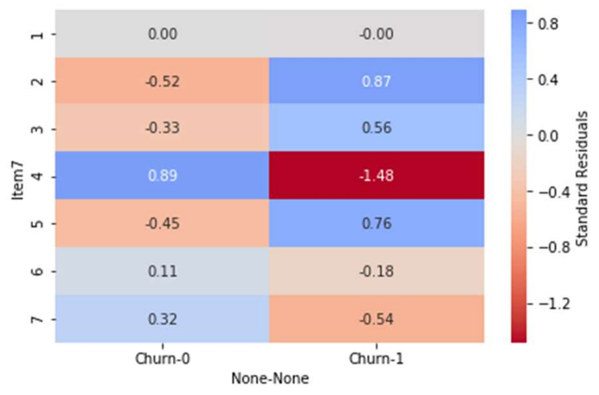


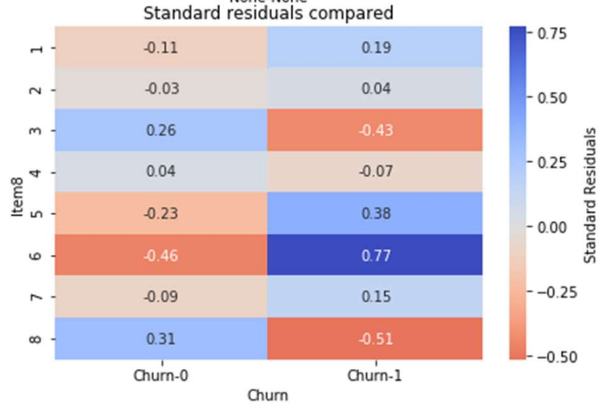




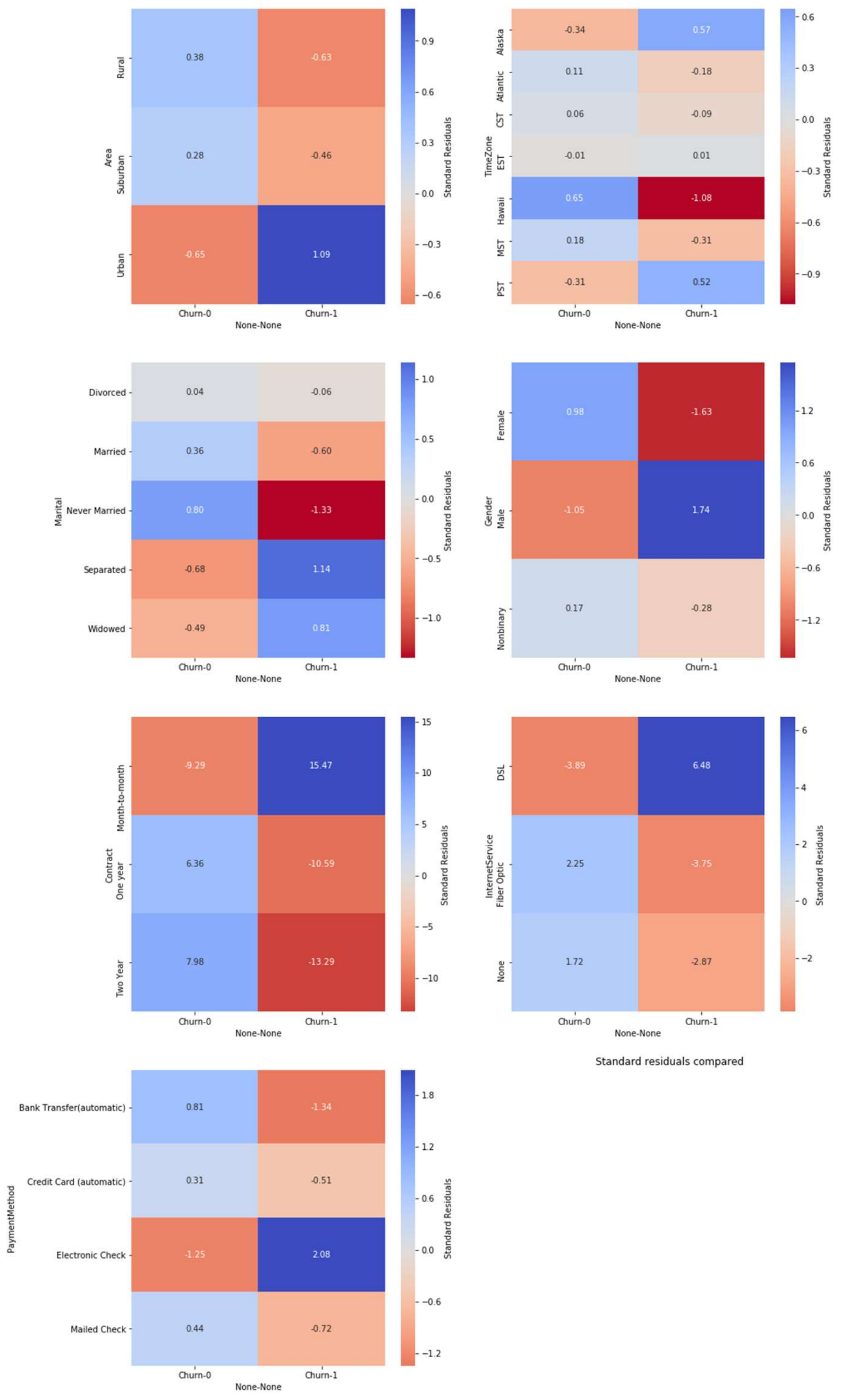








Bivariate standard residual heatmaps of each categorical variable:



(Seaborn, 2020)

From the above bivariate visualizations, one distribution of note is ‘Bandwidth_GB_Year’ versus ‘Churn’. Unlike the distribution of all other numeric variables versus the binary outcome of ‘Churn’, this distribution is differs depending on the outcome. Perhaps, ‘Churn’ is an important predictor in the subsequently generated logistic regression models.

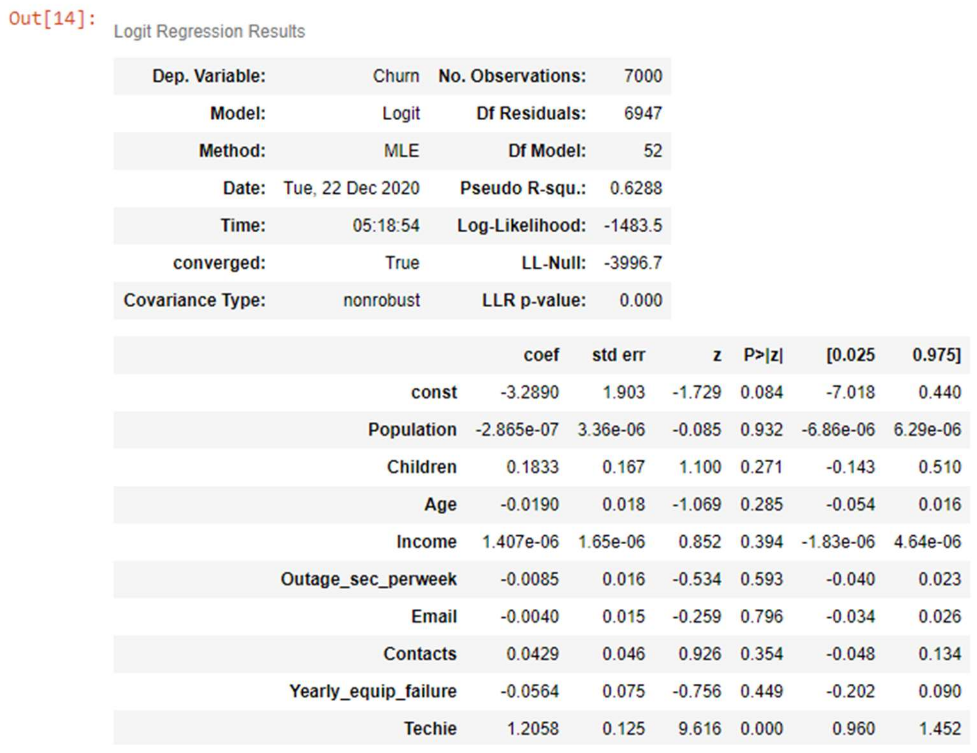
3.E

The entire prepared dataset is exported in the source code associated with this analysis and is included in this directory under the name: ‘**prepared_dataset.csv**’

<u>Part IV: Model Comparison and Analysis:</u>

4.A

The output below is a summary of a logistic regression model generated from the ‘Satatsmodels.Logit’ library. The model included every potential predictor variable in the dataset as a part of the equation to predict ‘Churn’. Note that the model is trained on 70% of the dataset while 30% is preserved for model evaluation.

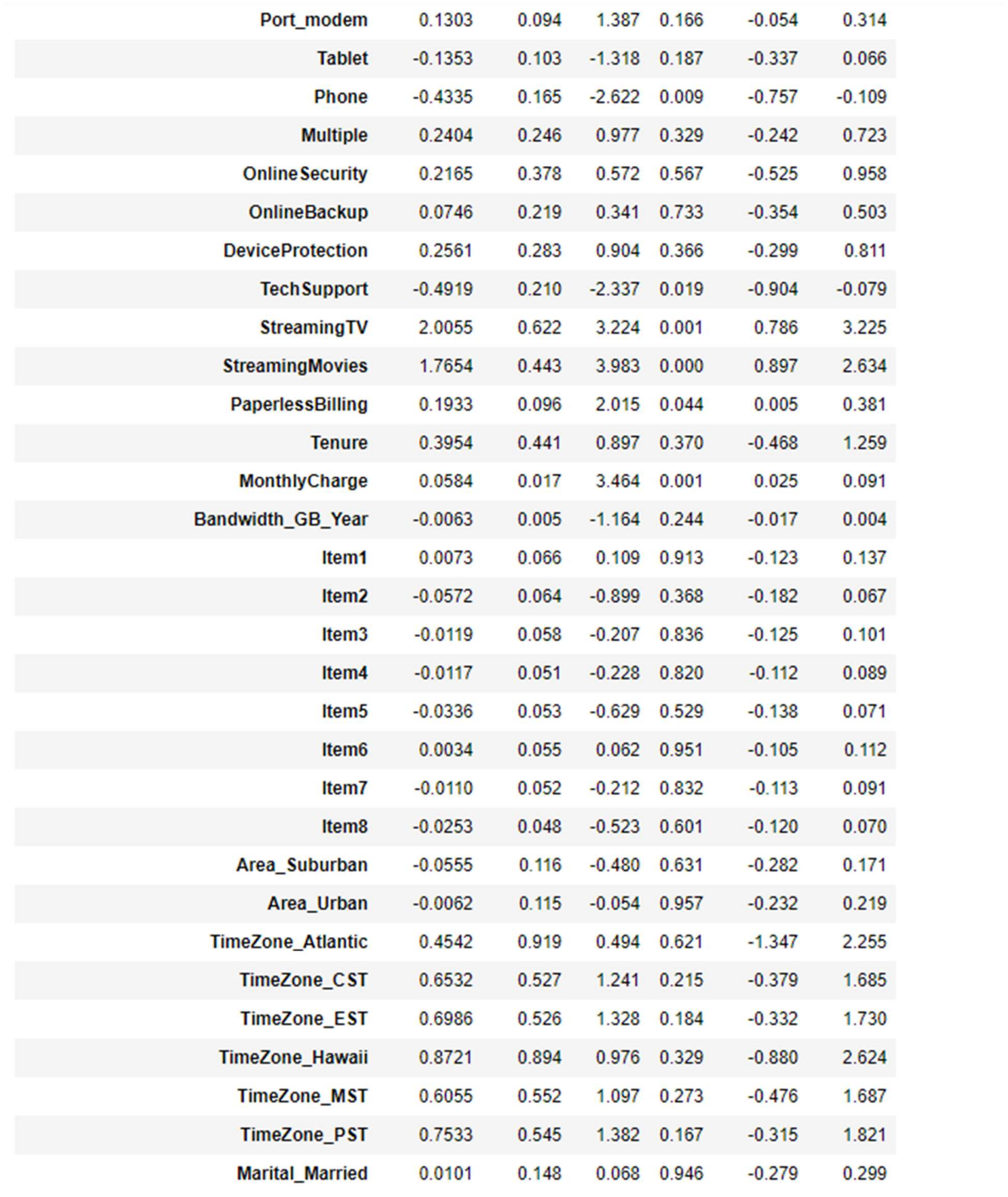

Out[14]:

Logit Regression Results

Dep. Variable:	Churn	**No. Observations:**	7000
Model:	Logit	**Df Residuals:**	6947
Method:	MLE	**Df Model:**	52
Date:	Tue, 22 Dec 2020	**Pseudo R-squ.:**	0.6288
Time:	05:18:54	**Log-Likelihood:**	-1483.5
converged:	True	**LL-Null:**	-3996.7
Covariance Type:	nonrobust	**LLR p-value:**	0.000

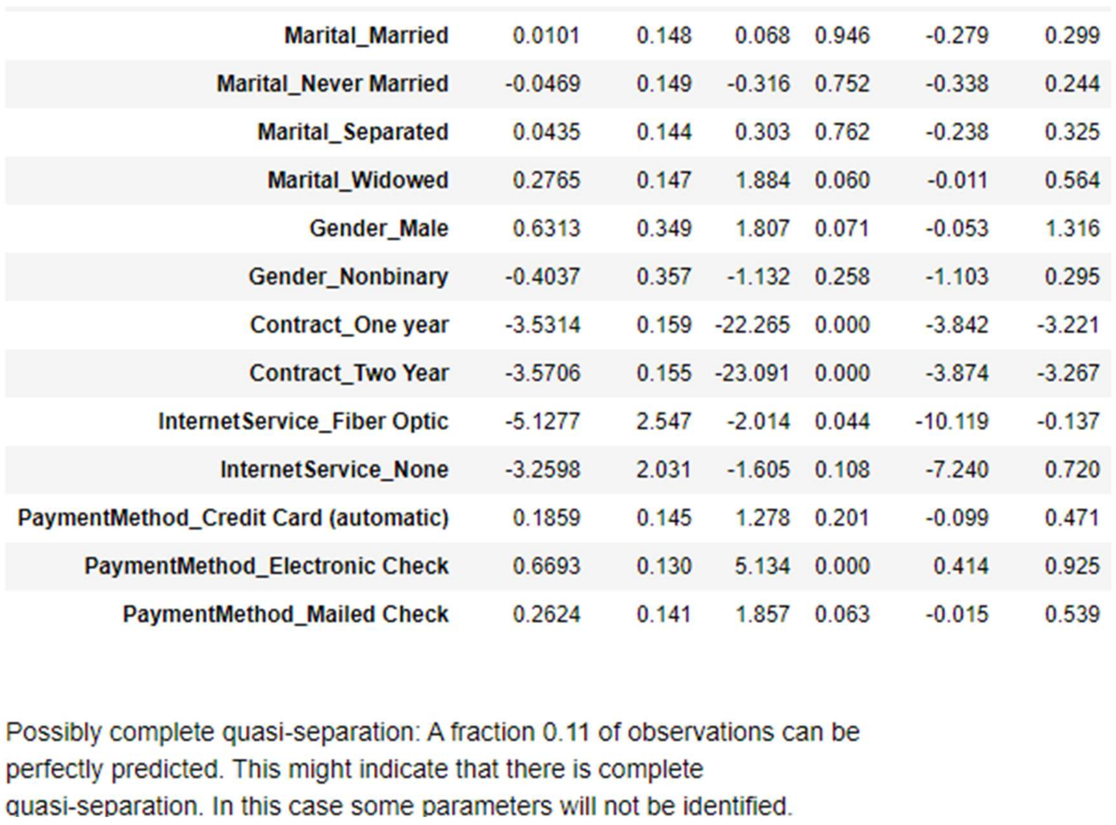
	coef	std err	z	P>|z|	[0.025	0.975]
const	-3.2890	1.903	-1.729	0.084	-7.018	0.440
Population	-2.865e-07	3.36e-06	-0.085	0.932	-6.86e-06	6.29e-06
Children	0.1833	0.167	1.100	0.271	-0.143	0.510
Age	-0.0190	0.018	-1.069	0.285	-0.054	0.016
Income	1.407e-06	1.65e-06	0.852	0.394	-1.83e-06	4.64e-06
Outage_sec_perweek	-0.0085	0.016	-0.534	0.593	-0.040	0.023
Email	-0.0040	0.015	-0.259	0.796	-0.034	0.026
Contacts	0.0429	0.046	0.926	0.354	-0.048	0.134
Yearly_equip_failure	-0.0564	0.075	-0.756	0.449	-0.202	0.090
Techie	1.2058	0.125	9.616	0.000	0.960	1.452

Model summary continued:



Port_modem	0.1303	0.094	1.387	0.166	-0.054	0.314
Tablet	-0.1353	0.103	-1.318	0.187	-0.337	0.066
Phone	-0.4335	0.165	-2.622	0.009	-0.757	-0.109
Multiple	0.2404	0.246	0.977	0.329	-0.242	0.723
OnlineSecurity	0.2165	0.378	0.572	0.567	-0.525	0.958
OnlineBackup	0.0746	0.219	0.341	0.733	-0.354	0.503
DeviceProtection	0.2561	0.283	0.904	0.366	-0.299	0.811
TechSupport	-0.4919	0.210	-2.337	0.019	-0.904	-0.079
StreamingTV	2.0055	0.622	3.224	0.001	0.786	3.225
StreamingMovies	1.7654	0.443	3.983	0.000	0.897	2.634
PaperlessBilling	0.1933	0.096	2.015	0.044	0.005	0.381
Tenure	0.3954	0.441	0.897	0.370	-0.468	1.259
MonthlyCharge	0.0584	0.017	3.464	0.001	0.025	0.091
Bandwidth_GB_Year	-0.0063	0.005	-1.164	0.244	-0.017	0.004
Item1	0.0073	0.066	0.109	0.913	-0.123	0.137
Item2	-0.0572	0.064	-0.899	0.368	-0.182	0.067
Item3	-0.0119	0.058	-0.207	0.836	-0.125	0.101
Item4	-0.0117	0.051	-0.228	0.820	-0.112	0.089
Item5	-0.0336	0.053	-0.629	0.529	-0.138	0.071
Item6	0.0034	0.055	0.062	0.951	-0.105	0.112
Item7	-0.0110	0.052	-0.212	0.832	-0.113	0.091
Item8	-0.0253	0.048	-0.523	0.601	-0.120	0.070
Area_Suburban	-0.0555	0.116	-0.480	0.631	-0.282	0.171
Area_Urban	-0.0062	0.115	-0.054	0.957	-0.232	0.219
TimeZone_Atlantic	0.4542	0.919	0.494	0.621	-1.347	2.255
TimeZone_CST	0.6532	0.527	1.241	0.215	-0.379	1.685
TimeZone_EST	0.6986	0.526	1.328	0.184	-0.332	1.730
TimeZone_Hawaii	0.8721	0.894	0.976	0.329	-0.880	2.624
TimeZone_MST	0.6055	0.552	1.097	0.273	-0.476	1.687
TimeZone_PST	0.7533	0.545	1.382	0.167	-0.315	1.821
Marital_Married	0.0101	0.148	0.068	0.946	-0.279	0.299

Model summary continued:



Marital_Married	0.0101	0.148	0.068	0.946	-0.279	0.299
Marital_Never Married	-0.0469	0.149	-0.316	0.752	-0.338	0.244
Marital_Separated	0.0435	0.144	0.303	0.762	-0.238	0.325
Marital_Widowed	0.2765	0.147	1.884	0.060	-0.011	0.564
Gender_Male	0.6313	0.349	1.807	0.071	-0.053	1.316
Gender_Nonbinary	-0.4037	0.357	-1.132	0.258	-1.103	0.295
Contract_One year	-3.5314	0.159	-22.265	0.000	-3.842	-3.221
Contract_Two Year	-3.5706	0.155	-23.091	0.000	-3.874	-3.267
InternetService_Fiber Optic	-5.1277	2.547	-2.014	0.044	-10.119	-0.137
InternetService_None	-3.2598	2.031	-1.605	0.108	-7.240	0.720
PaymentMethod_Credit Card (automatic)	0.1859	0.145	1.278	0.201	-0.099	0.471
PaymentMethod_Electronic Check	0.6693	0.130	5.134	0.000	0.414	0.925
PaymentMethod_Mailed Check	0.2624	0.141	1.857	0.063	-0.015	0.539

```
Possibly complete quasi-separation: A fraction 0.11 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

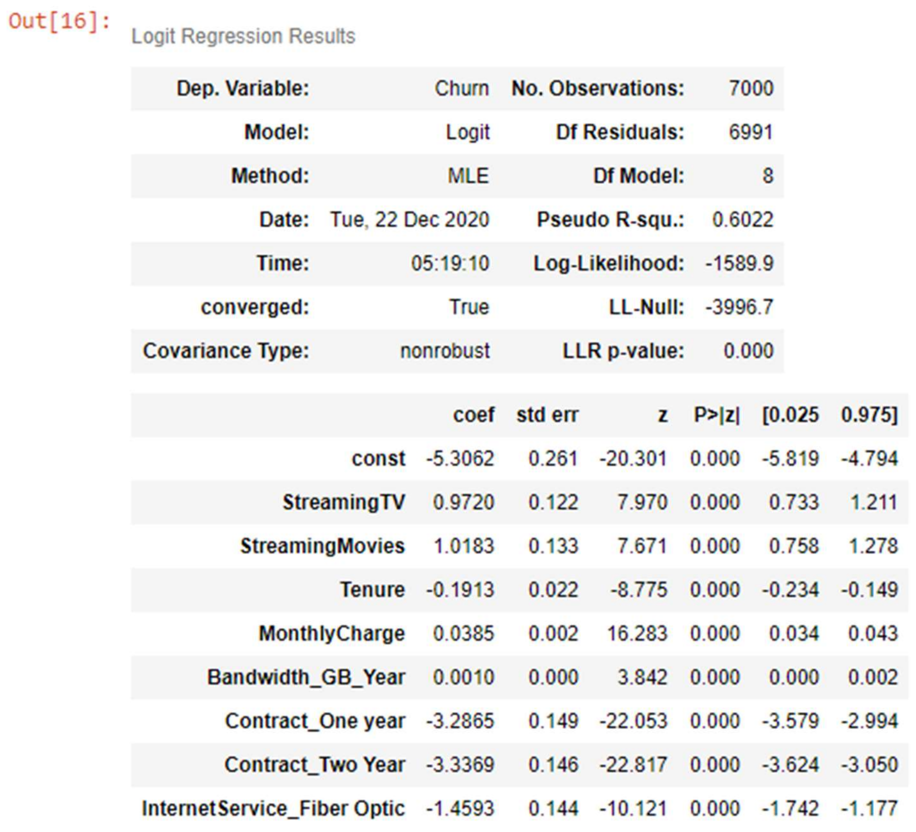
(Statsmodels, 2020)

4.B

From the output above, it is clear that a full model with every predictor variable has many unnecessary variables to create as simple of a predictive model as possible. Many variables have insignificant coefficients and extremely high p-values. Perhaps a more stable model with equally or higher predictive accuracy might be generated with significantly fewer input variables. To recursively eliminate variables in the predictive model in a statistical way, each variable is ranked by a standardized coefficient. This statistic is calculated for each variable by multiplying its coefficient with the relative standard deviation of all the other predictor variables. The resulting value is a numerical indication of that variable's or categorical level's contribution to the full model. (Scikit Learn, 2020) Then each variable with the lowest standardized coefficient is removed from the model until there are no predictor variables or categorical levels left. Each of the 52 iterations of this recursive removal train a model, test the predictive accuracy of the model, then calculate an AUC statistic for that model. The AUC statistic is equivalent to the area underneath a “receiver operator characteristic” curve. The ratio of the area under the curve to each correctly and incorrectly predicted observation results in a predictive accuracy statistic between 0 and 1. (Scikit Learn, 2020) The resulting model with the highest AUC score is then selected as the ideal logistic regression model. The result of the method outlined above creates a model with eight predictor variables. Each of the eight predictor variables (or categorical levels) have a statistically significant p-value. Additionally, the model has significantly less variables, yet generalizes more effectively than the full model. The result of this statistical variable reduction method is a model with higher predictive accuracy and a selection of variables that are more indicative in predicting customer ‘Churn’.

4.C

Below is an output of the reduced logistic regression model. For more details, see the 16th code cell of the source code.



```
Out[16]:
Logit Regression Results
```

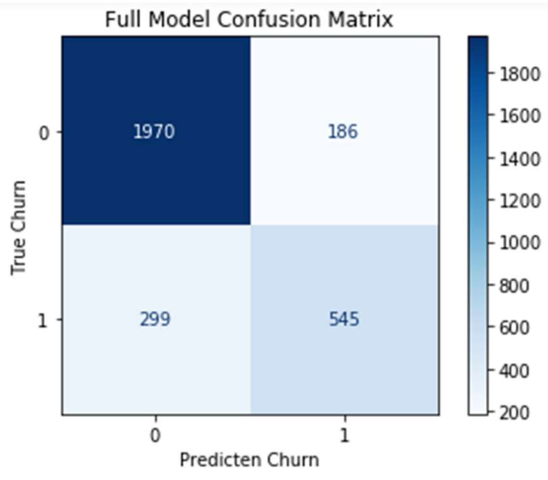
Dep. Variable:	Churn	**No. Observations:**	7000
Model:	Logit	**Df Residuals:**	6991
Method:	MLE	**Df Model:**	8
Date:	Tue, 22 Dec 2020	**Pseudo R-squ.:**	0.6022
Time:	05:19:10	**Log-Likelihood:**	-1589.9
converged:	True	**LL-Null:**	-3996.7
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-5.3062	0.261	-20.301	0.000	-5.819	-4.794
StreamingTV	0.9720	0.122	7.970	0.000	0.733	1.211
StreamingMovies	1.0183	0.133	7.671	0.000	0.758	1.278
Tenure	-0.1913	0.022	-8.775	0.000	-0.234	-0.149
MonthlyCharge	0.0385	0.002	16.283	0.000	0.034	0.043
Bandwidth_GB_Year	0.0010	0.000	3.842	0.000	0.000	0.002
Contract_One year	-3.2865	0.149	-22.053	0.000	-3.579	-2.994
Contract_Two Year	-3.3369	0.146	-22.817	0.000	-3.624	-3.050
InternetService_Fiber Optic	-1.4593	0.144	-10.121	0.000	-1.742	-1.177

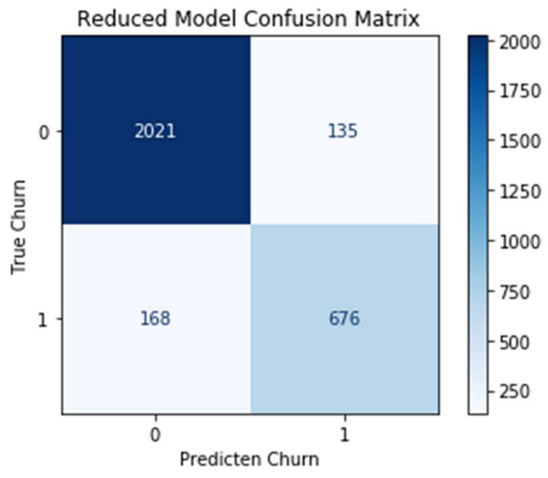
4.D

In concordance with the statistically based variable selection technique described in section D.2, the logic variable selection technique is outlined for clarity. Additionally, the resulting full and reduced regression models must be compared with specific evaluation metrics.

First, to compare the full and reduced logistic regression models, a simple comparison technique is the confusion matrix. In graphic below, the first confusion matrix displays the predictive accuracy of the full logistic regression model with every input variable while the second shows the same exact plot calculated from the reduced regression model. Additionally, the useful statistics of sensitivity, specificity and accuracy are displayed below each matrix.



```
specificity:  0.646
sensitivity:  0.914
accuracy:  0.838
```



```
specificity:  0.801
sensitivity:  0.937
accuracy:  0.899
```

(Scikit Learn, 2020)

In evaluation of the tables above, the reduced model is clearly more accurate in its predictive generalization. Note that specificity, sensitivity, and accuracy are all significantly higher. Further, the counts of true positives and true negatives are notably higher for the reduced model. In this case, a reduced model is superior or predicting customer 'Churn'. To emphasize this significance, in some cases it may be desirable to reduce the input variables in a model even if predictive accuracy is slightly less. Another point of support for the use of a reduced logistic regression model is the 'Statsmodels.Logit' summary produced in section D.3. For each predictor variable, the p-value is less than 0.001 which is indicative of a high statistical significance for each of the predictor variables included. (Statsmodels, 2020)

The comparison of the confusion matrix statistics above indicates a successful variable reduction technique described in section D.2. However, it may be useful to elaborate on the precise logic of that variable selection technique and why is lead to a more parsimonious model in terms of complexity. Essentially, the technique for variable selection in this analysis recursively eliminates every input variable in the full model until there are none left. At each step, the program removes the least contributive variable in terms of its standardized coefficient. The program then trains a logistic regression model with the least contributive variable removed and stores the AUC statistic in a Pandas data frame entitled 'auc_df'. Note that the number of variables remaining is stored as a reference key in 'auc_df'. For variable name storage, a tuple containing a list the exact variable names and number of variables for a reference key are stored in a dictionary entitled 'var_dict'. After the steps above are performed 52 times in a recursive loop for each dimension of the full regression model, the selection of 'auc_df' with the highest AUC score is queried in 'var_dict'. The subset of variables in 'var_dict' is the selection that created the logistic regression model with the highest AUC score, therefore this is the best reduced model that is used for analysis.

(Python, 2020)

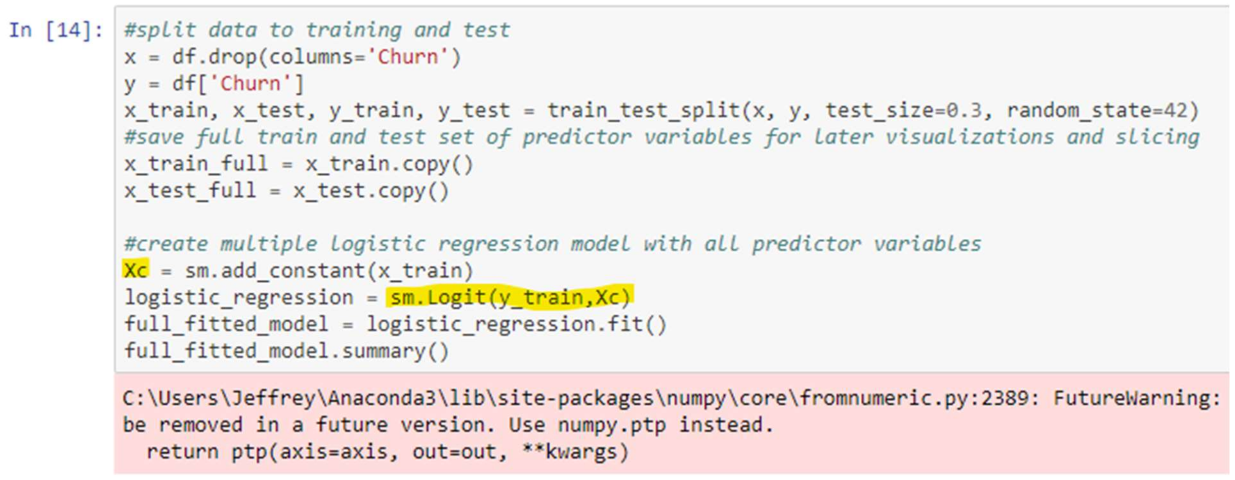
(Pandas, 2020)

(Scikit Learn, 2020)

4.E

Each calculation performed in variable reduction and model comparison is clearly included and annotated in the '**Part IV: Model Comparison and Analysis**' section of the source code, however, they are each outlined below for increased clarity.

First, the summary of the full regression model was produced with the following block of code. In the first three lines of code the dataset is split into a training and testing datasets. The training dataset is then used with the 'sm.Logit' function highlighted below. This function calculates an entire logistic regression model with 'y_train' containing the 'Churn' independent variable only. The 'sm.summary()' function is then used to display useful statistics for the entire model including p-values and coefficient values. (Statsmodels, 2020) Note, that after the variable reduction is performed on 'x_train' and 'y_train', nearly the same exact code is used to summarize the logistic regression model.

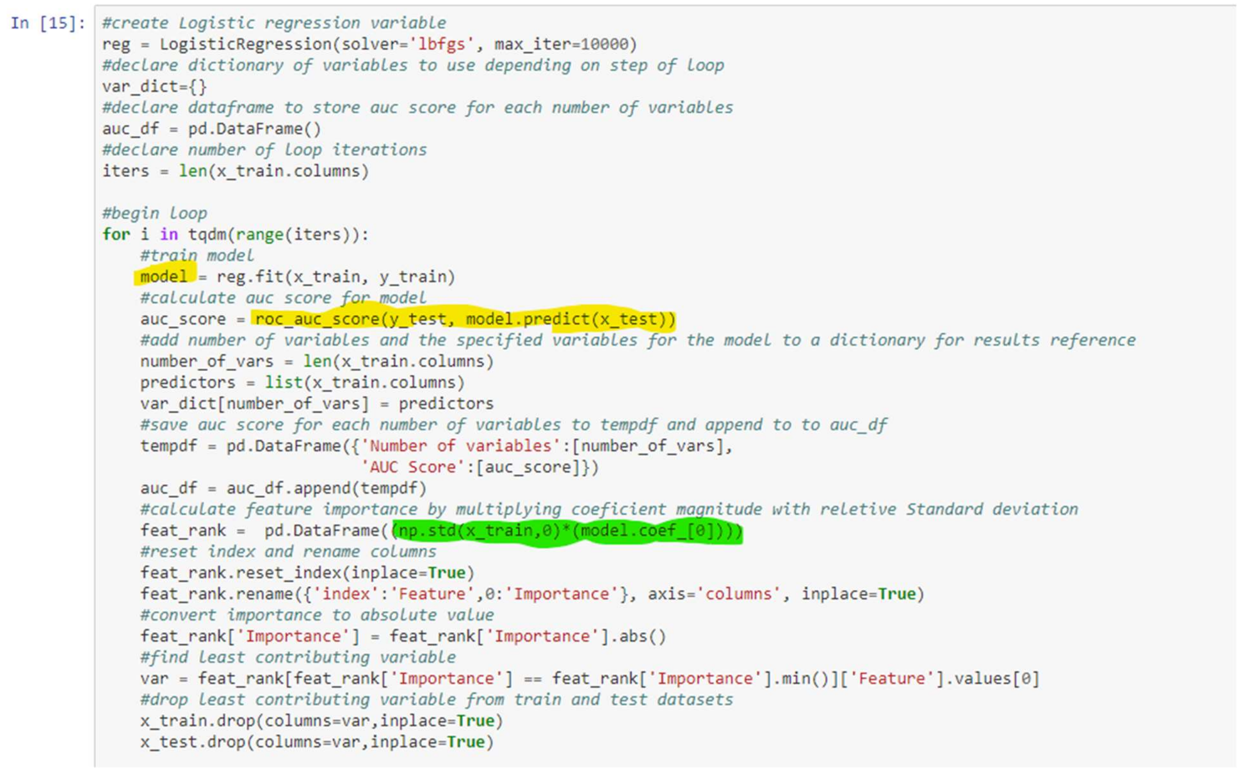


```
In [14]: #split data to training and test
         x = df.drop(columns='Churn')
         y = df['Churn']
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
         #save full train and test set of predictor variables for later visualizations and slicing
         x_train_full = x_train.copy()
         x_test_full = x_test.copy()

         #create multiple logistic regression model with all predictor variables
         Xc = sm.add_constant(x_train)
         logistic_regression = sm.Logit(y_train,Xc)
         full_fitted_model = logistic_regression.fit()
         full_fitted_model.summary()

         C:\Users\Jeffrey\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:
         be removed in a future version. Use numpy.ptp instead.
           return ptp(axis=axis, out=out, **kwargs)
```

Next, during variable reduction, the two statistical metrics calculated are the AUC score and the standardized coefficient. First the standardized coefficient is calculated for every variable during each step of the for-loop iterations in the code block previewed below. The standardized coefficient is calculated by multiplying the relative standard deviation of each variable with its coefficient calculated from the scikit Learn ‘LogisticRegression’ model built from the ‘x_train’ and ‘y_train’ datasets. (Scikit Learn, 2020) The standardized coefficient calculation is highlighted in green below. Next, the AUC score is calculated with the section highlighted in yellow below. Fist, on the inner portion of the formula, the ‘predict()’ function is used to test the scikit Learn logistic regression model by predicting ‘y_test’ values with the ‘x_test’ dataset. Then the outer portion of the formula is the scikit Learn ‘roc_auc_score()’ function. This calculates the exact AUC score based on the test dataset prediction results. (Scikit Learn, 2020)

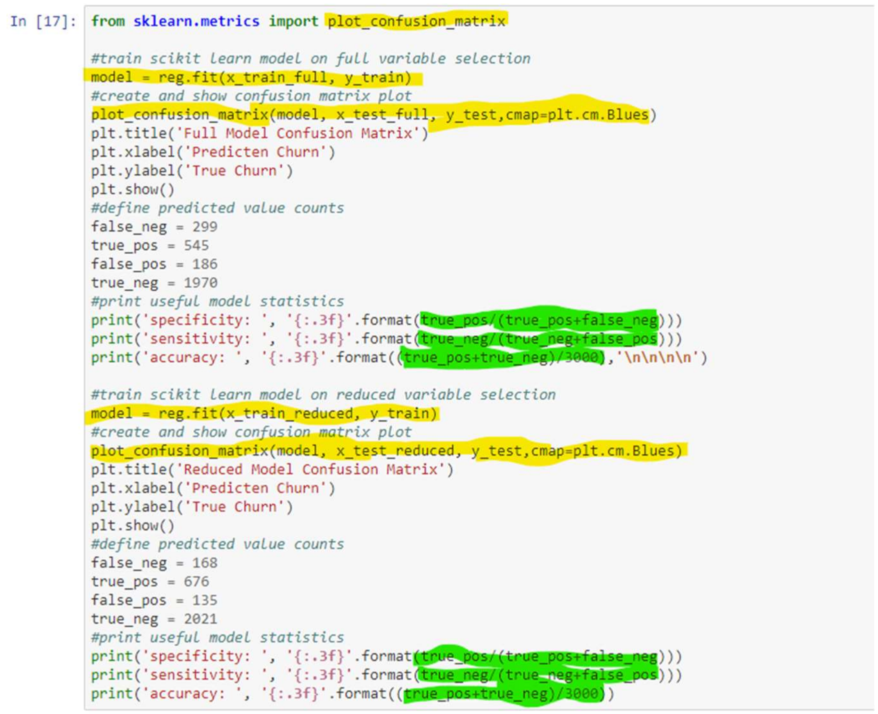


```
In [15]: #create Logistic regression variable
         reg = LogisticRegression(solver='lbfgs', max_iter=10000)
         #declare dictionary of variables to use depending on step of loop
         var_dict={}
         #declare dataframe to store auc score for each number of variables
         auc_df = pd.DataFrame()
         #declare number of loop iterations
         iters = len(x_train.columns)

         #begin loop
         for i in tqdm(range(iters)):
             #train model
             model = reg.fit(x_train, y_train)
             #calculate auc score for model
             auc_score = roc_auc_score(y_test, model.predict(x_test))
             #add number of variables and the specified variables for the model to a dictionary for results reference
             number_of_vars = len(x_train.columns)
             predictors = list(x_train.columns)
             var_dict[number_of_vars] = predictors
             #save auc score for each number of variables to tempdf and append to to auc_df
             tempdf = pd.DataFrame({'Number of variables':[number_of_vars],
                                    'AUC Score':[auc_score]})
             auc_df = auc_df.append(tempdf)
             #calculate feature importance by multiplying coeficient magnitude with reletive Standard deviation
             feat_rank =  pd.DataFrame((np.std(x_train,0)*(model.coef_[0])))
             #reset index and rename columns
             feat_rank.reset_index(inplace=True)
             feat_rank.rename({'index':'Feature',0:'Importance'}, axis='columns', inplace=True)
             #convert importance to absolute value
             feat_rank['Importance'] = feat_rank['Importance'].abs()
             #find least contributing variable
             var = feat_rank[feat_rank['Importance'] == feat_rank['Importance'].min()]['Feature'].values[0]
             #drop least contributing variable from train and test datasets
             x_train.drop(columns=var,inplace=True)
             x_test.drop(columns=var,inplace=True)
```

Finally, to calculate the confusion matrices for the full and reduced models, the following code block is used. Note that the ‘plot_confusion_matrix’ function is imported from the 'sklearn.metrics' library to achieve this task. (Scikit Learn, 2020) This function is implemented within the constraints of a matplotlib figure. The function directly takes the logistic regression model trained on the full training dataset in the first figure and the reduced dataset in the second figure, then tests the models on the full and reduced test datasets respectively. The prediction results are directly applied to the printed confusion matrices previewed in section E.1 of this text. These confusion matrix calculations are highlighted in yellow below. Additionally, highlighted in green are the printed statistics calculated from each confusion matrix. The Specificity is calculated by dividing the predicted positives from all actual positives while the Sensitivity is calculated by dividing the predicted negatives from all actual negatives. Finally, the accuracy is calculated as a ratio of all correct predictions to the total 3000 test observations.

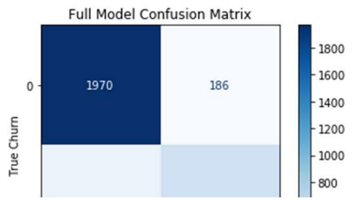
(Scikit Learn, 2020)



```
In [17]: from sklearn.metrics import plot_confusion_matrix

         #train scikit learn model on full variable selection
         model = reg.fit(x_train_full, y_train)
         #create and show confusion matrix plot
         plot_confusion_matrix(model, x_test_full, y_test,cmap=plt.cm.Blues)
         plt.title('Full Model Confusion Matrix')
         plt.xlabel('Predicten Churn')
         plt.ylabel('True Churn')
         plt.show()
         #define predicted value counts
         false_neg = 299
         true_pos = 545
         false_pos = 186
         true_neg = 1970
         #print useful model statistics
         print('specificity: ', '{:.3f}'.format(true_pos/(true_pos+false_neg)))
         print('sensitivity: ', '{:.3f}'.format(true_neg/(true_neg+false_pos)))
         print('accuracy: ', '{:.3f}'.format((true_pos+true_neg)/3000),'\n\n\n\n')

         #train scikit learn model on reduced variable selection
         model = reg.fit(x_train_reduced, y_train)
         #create and show confusion matrix plot
         plot_confusion_matrix(model, x_test_reduced, y_test,cmap=plt.cm.Blues)
         plt.title('Reduced Model Confusion Matrix')
         plt.xlabel('Predicten Churn')
         plt.ylabel('True Churn')
         plt.show()
         #define predicted value counts
         false_neg = 168
         true_pos = 676
         false_pos = 135
         true_neg = 2021
         #print useful model statistics
         print('specificity: ', '{:.3f}'.format(true_pos/(true_pos+false_neg)))
         print('sensitivity: ', '{:.3f}'.format(true_neg/(true_neg+false_pos)))
         print('accuracy: ', '{:.3f}'.format((true_pos+true_neg)/3000))
```



4.F

Every line of code outlined and screenshotted above is available for reproducibility in ‘**Task 2 Source Code Notebook.ipynb**’. Additionally, the output of the entire code after successful execution is stored in ‘**Task 2 Source Code Output.html**’.

Part V: Data Summary and Implications:

5.A

The final reduced logistic regression model from this analysis is summarized the following equation:

Predicted ‘Churn’ = -5.3062

+0.9720*’StreamingTV’

+1.0183*’StreamingMovies’

-0.1913*’Tenure’

+0.0385*’MonthlyCharge’

+0.0010*’Bandwidth_GB_Year’

-3.2865*’Contract_One year’

-3.3369*’Contract_Two year’

-1.4593*’InternetService_Fiber Optic’

(Statsmodels, 2020)

In this final model, each of the eight predictor variables are statistically significant. This is evident because each of the variables has a p-value of less than 0.001. To further emphasize the statistical significance of the effect for each of the predictor variables, the lowest z-score is for ‘Bandwidth_GB_Year’ which is still more than two standard deviations from the hypothesized mean at a value of 3.842. To interpret the coefficients of specific variables in the final model, the datatype, sign, and magnitude of the value must be considered. For ‘Tenure’, the effect is clearly negative. This means that as ‘Tenure’ increases holding all other variables equal, the occurrence of ‘Churn’ decreases. Since ‘Tenure’ is a continuous variable, the magnitude of the coefficient depends on the value of ‘Tenure’. If the value of ‘Tenure’ is 10, then the effect of ‘Tenure’ on ‘Churn’ is -1.913. For the binary variable, ‘StreamingMovies’, the coefficient is positive 1.0183. This means that if a customer has ‘StreamingMovies’ service, holding all other variables equal, the logit effect on Churn is 1.0183 times higher. Basically, a customer with ‘StreamingMovies’ service is more likely to churn, which is supported by the p-value of less than 0.001. This same logic of interpreting magnitude and sign of coefficients can be applied for all eight variables in the model.

(Glen, 2015)

Though this model is numerically sound with statistically significant variables, there are limitations. First, the model does lack a certain amount of accuracy. As evident from the contingency tables previewed in section E.1, the model predicts customer ‘Churn’ correctly approximately 0.899 percent of the time. Consequentially, implementing this model comes with a certain amount of risk. Although each variable appears to be statistically significant, there is a substantial amount of discordance which results in false positive and false negative values. Second, it is only assumed that the dataset is representative of unique observations. There may be observations in the dataset where the same customer has two accounts and two addresses. Finally, the size of the dataset is limited. Perhaps more data would contribute to a logistic regression model that is superior for generalizing ‘Churn’ predictions.

5.B

As a result of this final logistic regression model, two courses of action may be taken. First, the model may be applied to future customers to predict whether they will ‘Churn’. The telecommunications company may then investigate whether one of the significant effects in the model is affecting certain customer segments more significantly than others. Of course, this would need to be achieved with subsequent analysis and more model types. Perhaps another classification model may help define these segments. Second, this model may need to be continuously and recursively refined. With more data, the model will need to be retrained and retested. Perhaps the same data may be applied to a neural network in lieu of a logistic regression model. (Scikit Learn, 2020) In summary, the analysis process never ends, however, a telecommunications company may gain some preliminary insight from this logistic regression analysis.

References

Glen, S. (2015, September 21). *Variance Inflation Factor*. Statistics How To. https://www.statisticshowto.com/variance-inflation-factor/

Massaron, L., & Boschetti, A. (2016). *Regression analysis with Python*.

Pandas 0.25.1 documentation. (2020). Retrieved from https://pandas.pydata.org/pandas-docs/stable/whatsnew/index.html

Pennsylvania State University. (2018). *STAT 504: Analysis of Discrete Data*. PennState Eberly College of Science. https://online.stat.psu.edu/stat504/node/86/

Project Jupyter. (2020). *Jupyter*. https://jupyter.org/

Python Standard Library. (2020). Retrieved from https://docs.python.org/3/library/

Scikit Learn User Guide. (2020). Retrieved from https://scikit-learn.org/stable/user_guide.html

Seaborn: statistical data visualization. (2020). Retrieved from https://seaborn.pydata.org/

Statsmodels v0.12.1. (2020). Retrieved from https://www.statsmodels.org/stable/index.html

Stoltzfus, J.C. (2011, October 18). *Logistic regression: a brief primer*. National Library of Medicine. https://pubmed.ncbi.nlm.nih.gov/21996075/