#Standard Data Science Imports

import numpy as np

import pandas as pd

from pandas import Series, DataFrame

# Visualization libraries

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

# Statistics packages

import pylab

from statsmodels.formula.api import ols

import statistics

from scipy import stats

# Scikit-learn

import sklearn

from sklearn import preprocessing

from sklearn.preprocessing import LabelEncoder

from sklearn import metrics

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix

from sklearn.model\_selection import cross\_val\_score

from sklearn.metrics import classification\_report

# Load data set into Pandas Dataframe

churn\_df = pd.read\_csv(r'C:\Users\Hydraconix\Desktop\churn\_clean.csv')

# Checking for Null Values

churn\_df.isna().sum()

#Summary Statistics

churn\_df.Age.describe()

churn\_df.Children.describe()

churn\_df.Income.describe()

churn\_df.Outage\_sec\_perweek.describe()

churn\_df.Yearly\_equip\_failure.describe()

churn\_df.Tenure.describe()

churn\_df.MonthlyCharge.describe()

churn\_df.Bandwidth\_GB\_Year.describe()

# Rename Last 8 Survey Columns for better description of variables

churn\_df.rename(columns = {'Item1' : 'TimelyResponse',

'Item2' : 'Fixes' ,

'Item3' : 'Replacements' ,

'Item4' : 'Reliability' ,

'Item5' : 'Options' ,

'Item6' : 'Respectfulness' ,

'Item7' : 'Courteous' ,

'Item8' : 'Listening'},

inplace=True)

# Remove less meaningful demographic variables from statistics description

churn\_df = churn\_df.drop(columns=['CaseOrder' ,

'Customer\_id' ,

'Interaction' ,

'UID' ,

'City' ,

'State' ,

'County' ,

'Zip' ,

'Lat' ,

'Lng' ,

'Population' ,

'Area' ,

'TimeZone' ,

'Job' ,

'Marital' ,

'PaymentMethod'])

# Converting binary categorical variables to numeric variables

churn\_df['DummyChurn'] = [1 if v == 'Yes' else 0 for v in churn\_df['Churn']]

churn\_df['DummyTechie'] = [1 if v == 'Yes' else 0 for v in churn\_df['Techie']]

churn\_df['DummyPort\_modem'] = [1 if v == 'Yes' else 0 for v in churn\_df['Port\_modem']]

churn\_df['DummyTablet'] = [1 if v == 'Yes' else 0 for v in churn\_df['Tablet']]

churn\_df['DummyPhone'] = [1 if v == 'Yes' else 0 for v in churn\_df['Phone']]

churn\_df['DummyMultiple'] = [1 if v == 'Yes' else 0 for v in churn\_df['Multiple']]

churn\_df['DummyOnlineSecurity'] = [1 if v == 'Yes' else 0 for v in churn\_df['OnlineSecurity']]

churn\_df['DummyOnlineBackup'] = [1 if v == 'Yes' else 0 for v in churn\_df['OnlineBackup']]

churn\_df['DummyDeviceProtection'] = [1 if v == 'Yes' else 0 for v in churn\_df['DeviceProtection']]

churn\_df['DummyTechSupport'] = [1 if v == 'Yes' else 0 for v in churn\_df['TechSupport']]

churn\_df['DummyStreamingTV'] = [1 if v == 'Yes' else 0 for v in churn\_df['StreamingTV']]

churn\_df['DummyStreamingMovies'] = [1 if v == 'Yes' else 0 for v in churn\_df['StreamingMovies']]

churn\_df['DummyPaperlessBilling'] = [1 if v == 'Yes' else 0 for v in churn\_df['PaperlessBilling']]

# Converting ordinal categorical data into numeric variables

churn\_df['DummyInternetService'] = churn\_df.InternetService.map({'None' : 0, 'DSL' : 1, 'Fiber Optic' : 2})

churn\_df['DummyContract'] = churn\_df.Contract.map({'Month-to-month' : 0, 'One year' : 1, 'Two Year' : 2})

churn\_df['DummyGender'] = churn\_df.Gender.map({'Nonbinary' : 0, 'Male' : 1, 'Female' : 2})

# Drop original categorical variables from dataframe

churn\_df = churn\_df.drop(columns=['Gender' ,

'Churn' ,

'Techie' ,

'Contract' ,

'Port\_modem' ,

'Tablet' ,

'InternetService' ,

'Phone' ,

'Multiple' ,

'OnlineSecurity' ,

'OnlineBackup',

'DeviceProtection' ,

'TechSupport' ,

'StreamingTV',

'StreamingMovies',

'PaperlessBilling'])

#Create histograms of continuous variables

churn\_df[['Children',

'Age' ,

'Income' ,

'Outage\_sec\_perweek' ,

'Email' ,

'Contacts' ,

'Yearly\_equip\_failure' ,

'Tenure' , 'MonthlyCharge' ,

'Bandwidth\_GB\_Year' ,

‘DummyGender’ ,

‘DummyInternetService’ ,

‘DummyContract’]].hist()

plt.savefig('churn\_pyplot.jpg')

plt.tight\_layout()

# Create Seaborn boxplots for continuous variables

sns.boxplot('Tenure' , data = churn\_df)

plt.show()

sns.boxplot('MonthlyCharge' , data = churn\_df)

plt.show()

sns.boxplot('Bandwidth\_GB\_Year' , data = churn\_df)

plt.show()

# Run scatterplots to show direct or inverse relationships between target & independent variables

sns.scatterplot(x=churn\_df['Children'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['Age'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['Income'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['DummyGender'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['Outage\_sec\_perweek'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['Email'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['Contacts'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['Yearly\_equip\_failure'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['DummyTechie'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['Tenure'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['MonthlyCharge'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['Bandwidth\_GB\_Year'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['TimelyResponse'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['Fixes'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['Replacements'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['Reliability'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['Options'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['Respectfulness'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['Courteous'], y=churn\_df['DummyChurn'], color='red')

plt.show()

sns.scatterplot(x=churn\_df['Listening'], y=churn\_df['DummyChurn'], color='red')

plt.show()

# Provide a copy of the prepared data set

churn\_df.to\_csv('churn\_prepared\_log.csv')

# Construct an initial logistic regression model from all predictors that were identified in Part C2

churn\_logit\_model = logit("DummyChurn ~ Children + Age + Income + Outage\_sec\_perweek + Email +Contacts +Yearly\_equip\_failure +Tenure + MonthlyCharge + TimelyResponse + Fixes + Replacements + Reliability + Options + Respectfulness + Courteous + Listening", data=churn\_df).fit()

print(churn\_logit\_model.params)

print(churn\_logit\_model.summary())

churn\_logit\_model2 = logit("DummyChurn ~ Children + Age + Income + Outage\_sec\_perweek + Email +Contacts +Yearly\_equip\_failure +Tenure + MonthlyCharge + TimelyResponse + Fixes + Replacements + Reliability + Options + Respectfulness + Courteous + Listening + Bandwidth\_GB\_Year + DummyTechie + DummyPort\_modem + DummyTablet + DummyPhone + DummyMultiple + DummyOnlineSecurity + DummyOnlineBackup + DummyDeviceProtection + DummyTechSupport + DummyStreamingTV + DummyStreamingMovies + DummyPaperlessBilling + DummyInternetService + DummyContract + DummyGender", data=churn\_df).fit()

print(churn\_logit\_model2.params)

print(churn\_logit\_model2.summary())

# Create dataframe for heatmap bivariate analysis of correlation

churn\_bivariate = churn\_df[['DummyChurn', 'Children', 'Age', 'Income',

'Outage\_sec\_perweek', 'Yearly\_equip\_failure', 'DummyTechie', 'DummyContract',

'DummyPort\_modem', 'DummyTablet', 'DummyInternetService',

'DummyPhone', 'DummyMultiple', 'DummyOnlineSecurity',

'DummyOnlineBackup', 'DummyDeviceProtection',

'DummyTechSupport', 'DummyStreamingTV',

'DummyPaperlessBilling','Email', 'Contacts',

'Tenure', 'MonthlyCharge', 'Bandwidth\_GB\_Year', 'TimelyResponse', 'Fixes',

'Replacements', 'Reliability', 'Options', 'Respectfulness',

'Courteous', 'Listening']]

# Run Seaborn heatmap

sns.heatmap(churn\_bivariate.corr(), annot=False)

plt.show()

churn\_bivariate = churn\_df[['DummyChurn', 'Bandwidth\_GB\_Year', 'Children',

'Tenure', 'TimelyResponse', 'Fixes',

'Replacements', 'Respectfulness',

'Courteous', 'Listening']]

sns.heatmap(churn\_bivariate.corr(), annot=True)

plt.show()

# Run reduced logistic regression

churn\_logit\_model\_reduced = logit("DummyChurn ~ DummyTechie + DummyContract + DummyInternetService + DummyStreamingTV", data=churn\_df).fit()

print(churn\_logit\_model\_reduced.summary())

# Confusion Matrix

# Import the prepared dataset

dataset = pd.read\_csv('churn\_prepared\_log.csv')

X = dataset.iloc[:, 1:-1].values

y = dataset.iloc[:, -1].values

# Split the dataset into the Training set and Test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

# Training the Logistic Regression model on the Training set

classifier = LogisticRegression(random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predict the Test set results

y\_pred = classifier.predict(X\_test)

# Make the Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

y\_predict\_test = classifier.predict(X\_test)

cm2 = confusion\_matrix(y\_test, y\_predict\_test)

sns.heatmap(cm2, annot=True)

# Classification Report

print(classification\_report(y\_test, y\_predict\_test))