Joaquin Emmanuel J. Gonzales 155244
DEC 130 – X 4 BS ME

Telco Churn Model: Logistic Regression

Before playing around with the training and testing dataset, I converted all categorical variables to binary using Excel in order to avoid confusion. Although I had the option to make dummy variables in Python, it would be confusing to do so since most of the columns would be named as 'Yes' or 'No'.

```
telco_df.TotalCharges = pd.to_numeric(telco_df.TotalCharges, errors ='coerce')
telco_df = telco_df.dropna()
X_train, X_test, Y_train, Y_test = train_test_split(telco_df.drop(['customerID','Churn','Partner','CreditCard','BankTransfer','ElectronicCheck',
'Male','Phone','OnineBackup','OnlineSecurity','TechSupport','DeviceProtection'], axis = 1).astype(float), telco_df['Churn'], test_size = 0.2)
```

```
logit_model=sm.Logit(Y_train,X_train)
result=logit_model.fit()
result.summary()
```

After loading the training set, the TotalCharges variable had to be converted to a float datatype. Following a 80-20% train-test proportion, I was able to fit a logistic regression model with Churn as the dependent variable. A test run that included all variables was performed, and the regression table showed that some of the variables are insignificant predictors of customer churn. Some of these variables are Partner, gender, PhoneService, PaymentMethod, OnlineBackup, OnlineSecurity, TechSupport, and DeviceProtection.

```
Optimization terminated successfully.
         Current function value: 0.422671
         Iterations 8
```

Logit Regression Results

Dep. Variable:	Churn	**No. Observations:**	5227
Model:	Logit	**Df Residuals:**	5214
Method:	MLE	**Df Model:**	12
Date:	Thu, 20 Sep 2018	**Pseudo R-squ.:**	0.2722
Time:	21:48:27	**Log-Likelihood:**	-2209.3
converged:	True	**LL-Null:**	-3035.4
		LLR p-value:	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
SeniorCitizen	0.2971	0.096	3.087	0.002	0.109	0.486
Dependents	-0.1701	0.092	-1.847	0.065	-0.351	0.010
tenure	-0.0662	0.006	-10.578	0.000	-0.078	-0.054
MultipleLines	0.5096	0.095	5.341	0.000	0.323	0.697
DSL	1.8173	0.193	9.398	0.000	1.438	2.196
FiberOptic	3.8759	0.307	12.622	0.000	3.274	4.478
StreamingTV	0.7433	0.100	7.404	0.000	0.547	0.940
StreamingMovies	0.6568	0.100	6.555	0.000	0.460	0.853
OneYear	-0.8000	0.123	-6.492	0.000	-1.042	-0.558
TwoYear	-1.5618	0.199	-7.844	0.000	-1.952	-1.172
PaperlessBilling	0.3110	0.083	3.730	0.000	0.148	0.474
MonthlyCharges	-0.0495	0.004	-11.813	0.000	-0.058	-0.041
TotalCharges	0.0004	6.98e-05	5.792	0.000	0.000	0.001

```
np.exp(result.params)
```

```
SeniorCitizen        1.345985
Dependents           0.843598
tenure               0.935955
MultipleLines        1.664651
DSL                  6.155225
FiberOptic          48.224391
StreamingTV          2.102800
StreamingMovies      1.928623
OneYear              0.449334
TwoYear              0.209755
PaperlessBilling     1.364820
MonthlyCharges       0.951745
TotalCharges         1.000404
dtype: float64
```

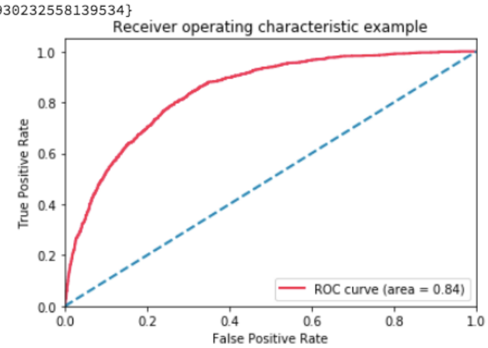
Running the model again, the regression table would show that Dependents is not a significant predictor of customer churn. The e^{β} were also calculated, and results would show that customers that are subscribed to FiberOptic are highly likely to churn out of the telco's service.

```
thresh = 0.549
print(accuracy_score(result.predict(X_train).apply(lambda x: x>thresh), Y_train))
print(accuracy_score(result.predict(X_test).apply(lambda x: x>thresh), Y_test))
```

```
0.8004591543906638
0.8201989288446825
```

```
get_conf(Y_train.values, result.predict(X_train).apply(lambda x: x>0.5).values)
```

```
{'tp': 765,
 'tn': 3412,
 'fp': 417,
 'fn': 633,
 'acc': 0.7991199540845609,
 'tpr': 0.5472103004291845,
 'tnr': 0.8910942804909898,
 'precision': 0.6472081218274112,
 'f1': 0.5930232558139534}
```



By setting the threshold at 54.9%, the training set has an accuracy of 80% while the testing set has an accuracy of 82%. Setting the threshold lower allows the model to minimize false negatives. The logic behind this thinking is the fact that a wrong prediction of a customer **NOT** churning is much more harmful to the telco company than a wrong prediction that the customer would churn. If a customer is predicted not to churn but churns anyway, the company was not able to do something to convince the customer to stay with them. However, if a customer is predicted to churn but does not, it just gives that particular client more perks to stay with the company.

```
significantFeatures = ['tenure','MonthlyCharges','TotalCharges','MultipleLines','DSL',
'FiberOptic','StreamingTV','StreamingMovies','OneYear','TwoYear','PaperlessBilling','SeniorCitizen']
```

```
model = LogisticRegression()
```

```
trainData = telco_train[significantFeatures]
testData = telco_test[significantFeatures]
target = telco_train.Churn
```

```
telco_train.TotalCharges = pd.to_numeric(telco_train.TotalCharges, errors ='coerce')
telco_test.TotalCharges = pd.to_numeric(telco_test.TotalCharges, errors ='coerce')
telco_test.MonthlyCharges = pd.to_numeric(telco_test.MonthlyCharges, errors ='coerce')
telco_train = telco_train.dropna()
telco_test = telco_test.dropna()
```

```
model.fit(trainData,target)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```
predictions = model.predict(testData.head(500))
```

Running this model to the actual test data, the code on the previous page was used to fit the new, learning data and predict whether customers would churn or not. By data cleaning, two clients (with customer ID *4075-WKNIU* and *2775-SEFEE*) were removed from the test set as they do not have values for the TotalCharges variable. The array below are the churn predictions for each customer (in order from left to right). In summary, there are 100 clients of the telco company that are predicted to **churn** whereas 398 are predicted to **not churn**.

```
array([0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0])
```