Machine Learning Engineer Nanodegree

Capstone Project

Mohammed Ashraf Farouq Ali
June 9th, 2019

I. Definition

Project Overview

Context

Customer attrition, also known as customer churn, customer turnover, or customer defection, is the loss of clients or customers.

Telephone service companies, Internet service providers, pay TV companies, insurance firms, and alarm monitoring services, often use customer attrition analysis and customer attrition rates as one of their key business metrics because the cost of retaining an existing customer is far less than acquiring a new one. Companies from these sectors often have customer service branches which attempt to win back defecting clients, because recovered long-term customers can be worth much more to a company than newly recruited clients.

Companies usually make a distinction between voluntary churn and involuntary churn. Voluntary churn occurs due to a decision by the customer to switch to another company or service provider, involuntary churn occurs due to circumstances such as a customer's relocation to a long-term care facility, death, or the relocation to a distant location. In most applications, involuntary reasons for churn are excluded from the analytical models. Analysts tend to concentrate on voluntary churn, because it typically occurs due to factors of the

company-customer relationship which companies control, such as how billing interactions are handled or how after-sales help is provided.

predictive analytics use churn prediction models that predict customer churn by assessing their propensity of risk to churn. Since these models generate a small prioritized list of potential defectors, they are effective at focusing customer retention marketing programs on the subset of the customer base who are most vulnerable to churn.

DataSet & Input:

The data set includes information about:

- **Customers who left within the last month – the column is called Churn**
- **Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies**
- **Customer account information – how long they've been a customer, contract, payment method, paperless billing, monthly charges, and total charges**
- **Demographic info about customers – gender, age range, and if they have partners and dependents**
- **As well it consists of 7044 * 21 Cells of Info Ranging From Customer ID's to their Churn State**

Problem Statement

In this Case of the Project we'll be dealing with :

- **Exploring the Dataset**
- **Manipulation of data to Tenure Groups for later classification**
- **Use of Info provided to create Statistical Analysis of the state of Each Customer**
- **Understand the Cause of Customer Churning from the overall Subscribed Services & Other Info**
- **Visualizing the descriptive statistics of the whole Dataset**
- **Preprocess Data & Remove Un-Necessary Data**
- **Remove Un-Correlated Data**
- **Shuffle & Split Data**
- **Train & Test Both SVM & Log Reg Models**

- **Resample, Shuffle & Split Data then retrain**
- **Measure & Compare Final Scores and Improvements**

Metrics

Recall and precision: Precision-Recall is a useful measure of success of prediction when the classes are very imbalanced. In information retrieval, precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned. A system with high recall but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels. A system with high precision but low recall is just the opposite, returning very few results, but most of its predicted labels are correct when compared to the training labels. An ideal system with high precision and high recall will return many results, with all results labeled correctly. And here a very good article about Recall and precision score.

F-Beta: F-Beta score is a way of measuring a certain accuracy for a model. It takes into consideration both the Recall and Precision metrics. If you don't know what those are, it's highly recommended to check the previous post about Recall & Precision.

--A quick definition of recall and precision, in a non-mathematical way:

Precision: high precision means that an algorithm returned substantially more relevant results than irrelevant ones **Recall:** high recall means that an algorithm returned most of the relevant results.

Good article and info here.

Confusion Matrix for calculating Logistic Regression Scores: A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing. Quick easy article here.

F-1 Score: is a measure of a test's accuracy, as it considers both the precision p and the recall r of the test to compute the score: p is the number of correct positive results divided by the number of all positive results returned by the classifier, and r is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. More Info here.

II. Analysis

Data Exploration

DataSet & Input:

The data set includes information about:

- **Customers who left within the last month – the column is called Churn**
- **Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies**

- **Customer account information – how long they've been a customer, contract, payment method, paperless billing, monthly charges, and total charges**
- **Demographic info about customers – gender, age range, and if they have partners and dependents**
- **As well it consists of 7044 * 21 Cells of Info Ranging From Customer ID's to their Churn State**

Load Dataset

3.1.1 Load DataSet

In [404]:
```
#load the dataset
data = telcom
data2 = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")

#display first 5 rows
data.head()
```

Out[404]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	TechSupport	StreamingTV
0	7590-VHVEG	Female	No	Yes	No	1	No	No phone service	DSL	No	...	No	No
1	5575-GNVDE	Male	No	No	No	34	Yes	No	DSL	Yes	...	No	No
2	3668-QPYBK	Male	No	No	No	2	Yes	No	DSL	Yes	...	No	No
3	7795-CFOCW	Male	No	No	No	45	No	No phone service	DSL	Yes	...	Yes	No
4	9237-HQITU	Female	No	No	No	2	Yes	No	Fiber optic	No	...	No	No

5 rows × 22 columns

Checking For missing Data and Unique Values

3.1.2 Checking for Missing Data

In [405]:
```
#Check if there's a missing data at each column
data.isnull().sum().max()
```

Out[405]: 0

3.1.5 Data Unique Values Calculation

In [408]:
```
print ("Rows     : " ,telcom.shape[0])
print ("Columns  : " ,telcom.shape[1])
print ("\nFeatures : \n" ,telcom.columns.tolist())
print ("\nMissing values :  ", telcom.isnull().sum().values.sum())
print ("\nUnique values :  \n",telcom.nunique())
```

```
Rows     :  7032
Columns  :  22

Features :
 ['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure', 'PhoneService', 'MultipleLines', 'InternetServic
e', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'Paperle
ssBilling', 'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn', 'tenure_group']

Missing values :   0

Unique values :
 customerID          7032
gender                 2
SeniorCitizen          2
Partner                2
Dependents             2
tenure                72
PhoneService           2
MultipleLines          3
InternetService        3
OnlineSecurity         2
OnlineBackup           2
DeviceProtection       2
TechSupport            2
StreamingTV            2
StreamingMovies        2
Contract               3
PaperlessBilling       2
PaymentMethod          4
MonthlyCharges      1584
TotalCharges        6530
Churn                  2
tenure_group           5
dtype: int64
```

Data Formatting

3.1.3 Data Format Description

In [406]:
```
#check data formating
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7032 entries, 0 to 7031
Data columns (total 22 columns):
customerID          7032 non-null object
gender              7032 non-null object
SeniorCitizen       7032 non-null object
Partner             7032 non-null object
Dependents          7032 non-null object
tenure              7032 non-null int64
PhoneService        7032 non-null object
MultipleLines       7032 non-null object
InternetService     7032 non-null object
OnlineSecurity      7032 non-null object
OnlineBackup        7032 non-null object
DeviceProtection    7032 non-null object
TechSupport         7032 non-null object
StreamingTV         7032 non-null object
StreamingMovies     7032 non-null object
Contract            7032 non-null object
PaperlessBilling    7032 non-null object
PaymentMethod       7032 non-null object
MonthlyCharges      7032 non-null float64
TotalCharges        7032 non-null float64
Churn               7032 non-null object
tenure_group        7032 non-null object
dtypes: float64(2), int64(1), object(19)
memory usage: 1.2+ MB
```

Descriptive Analysis

3.1.4 Data Descriptive Analysis

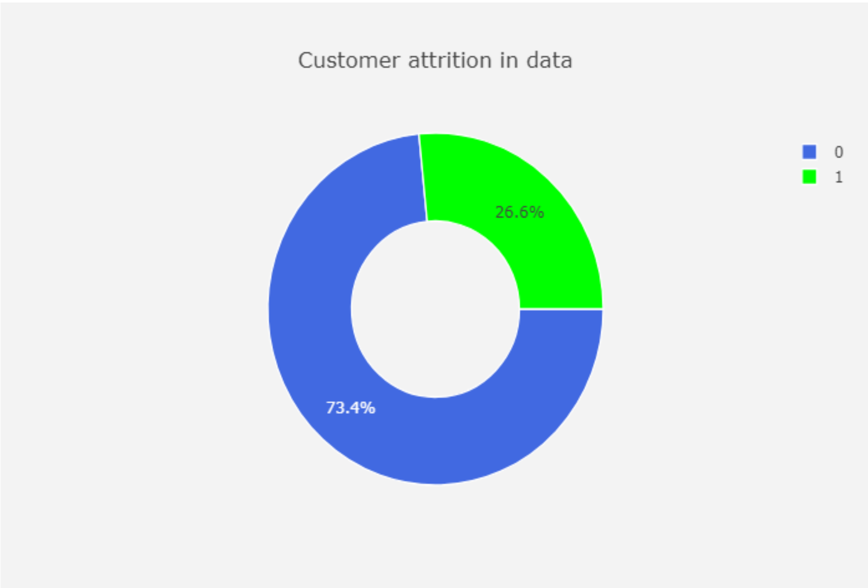
In [407]:
```
#Descriptive analysis
data.describe()
```

Out[407]:

	tenure	MonthlyCharges	TotalCharges
count	7032.000000	7032.000000	7032.000000
mean	32.421786	64.798208	2283.300441
std	24.545260	30.085974	2266.771362
min	1.000000	18.250000	18.800000
25%	9.000000	35.587500	401.450000
50%	29.000000	70.350000	1397.475000
75%	55.000000	89.862500	3794.737500
max	72.000000	118.750000	8684.800000

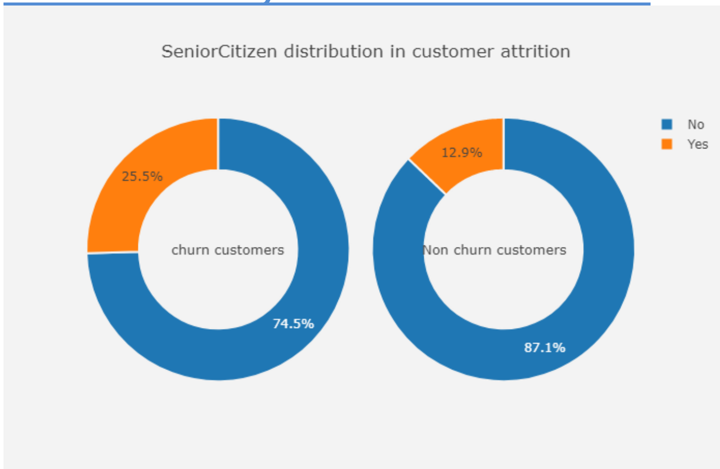
Exploratory Visualization

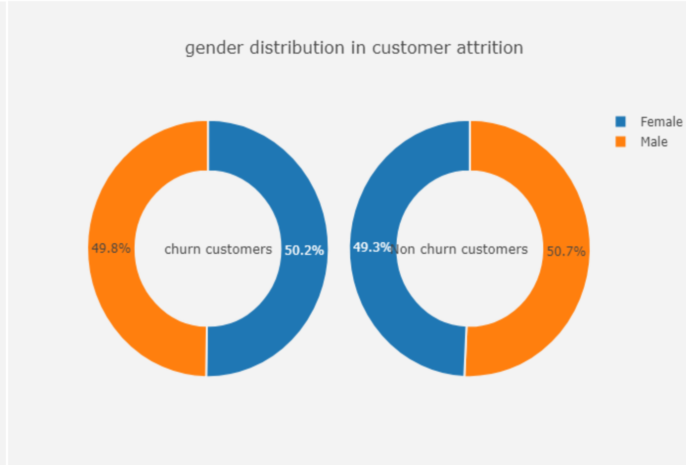
1.Churn to Non-Churn Proportion

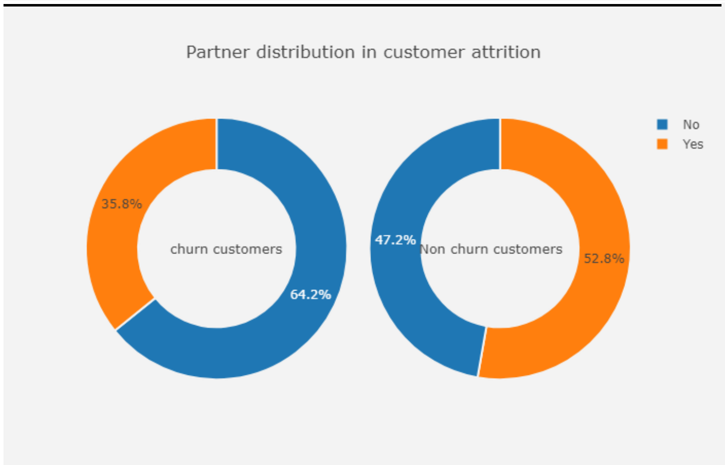


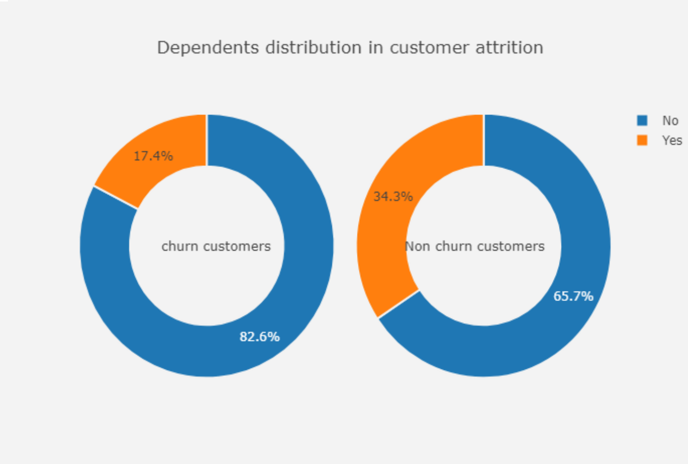
- The data is mostly unbalanced, Evaluate modeling on unbalanced data is a terrible mistake. Will see later why and how to work with unbalanced data.

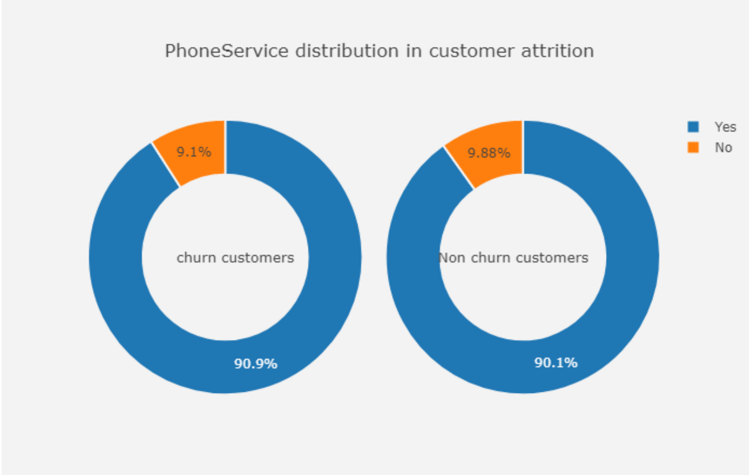
2.Statistical Analysis in Customer Attrition

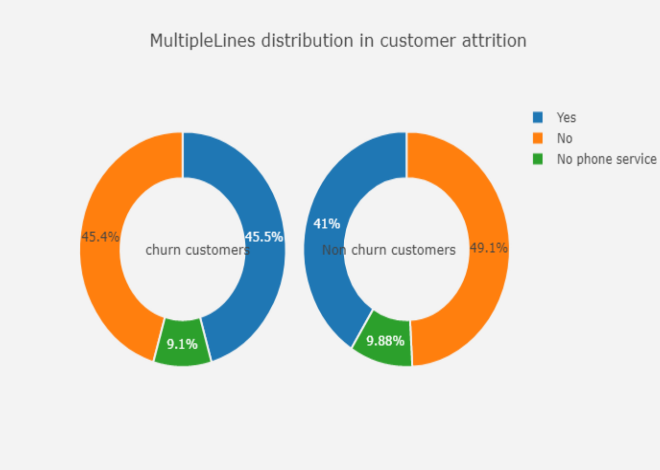


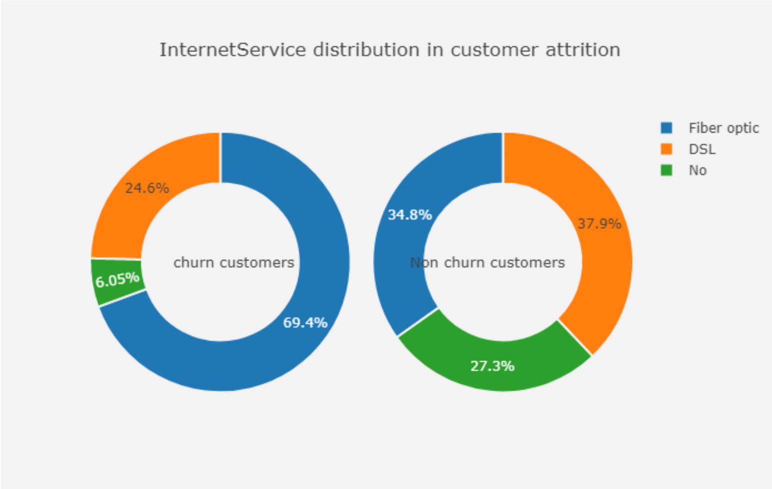


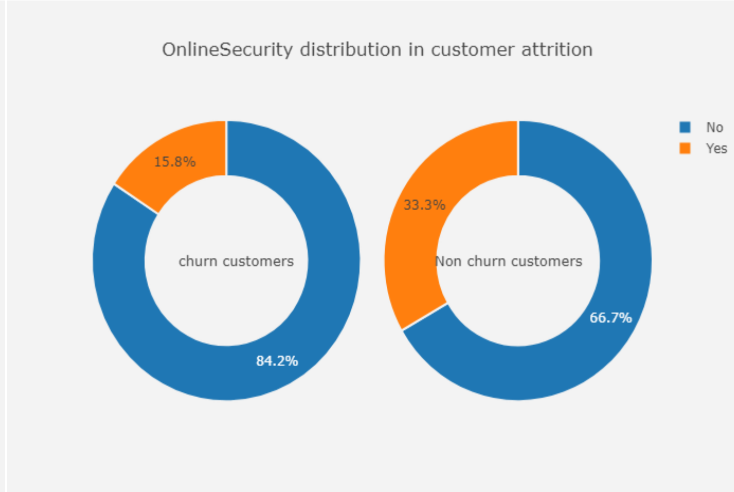


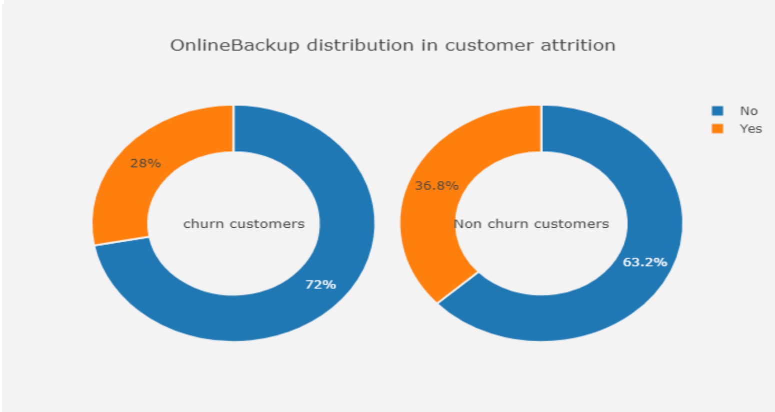


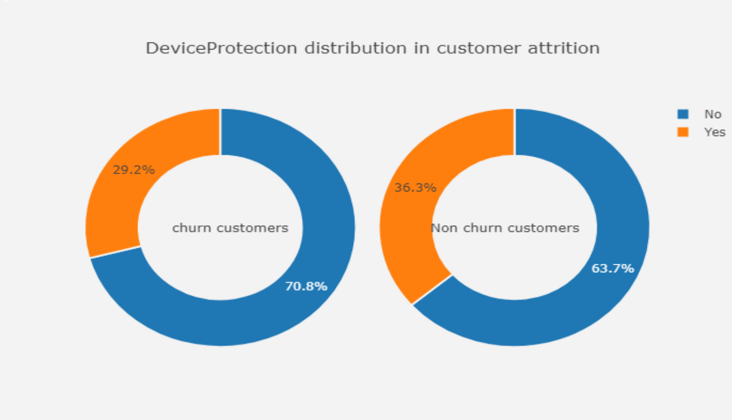


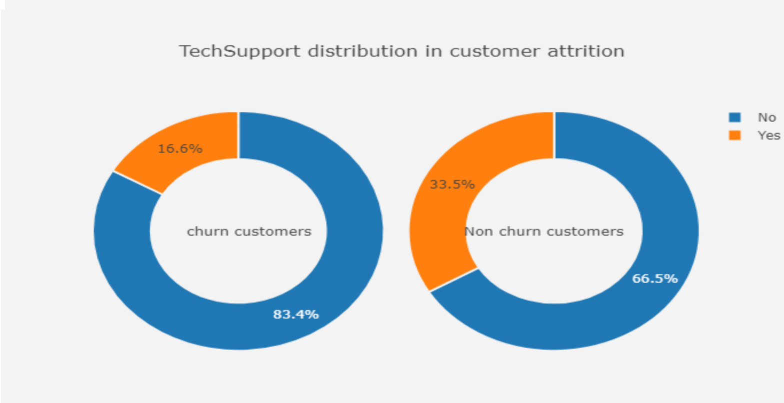


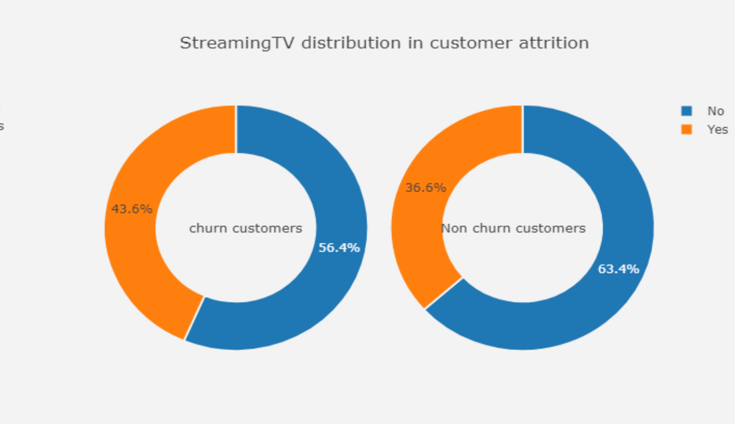

InternetService distribution in customer attrition
Fiber optic
DSL
No
churn customers
24.6%
6.05%
69.4%
Non churn customers
34.8%
37.9%
27.3%

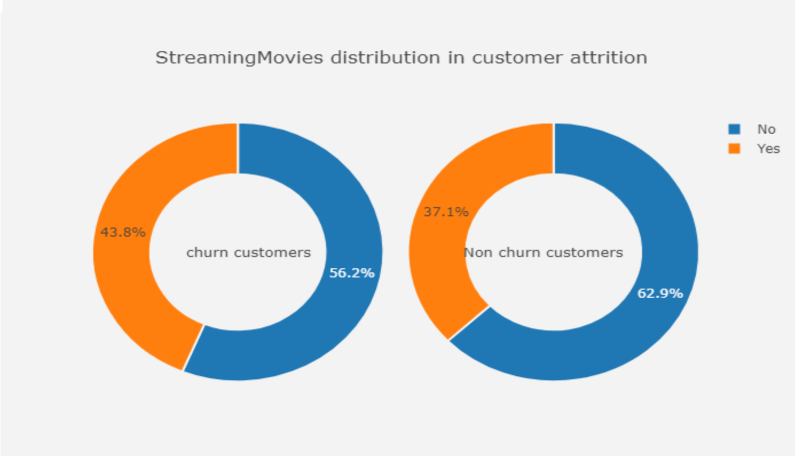

OnlineSecurity distribution in customer attrition
No
Yes
churn customers
15.8%
84.2%
Non churn customers
33.3%
66.7%

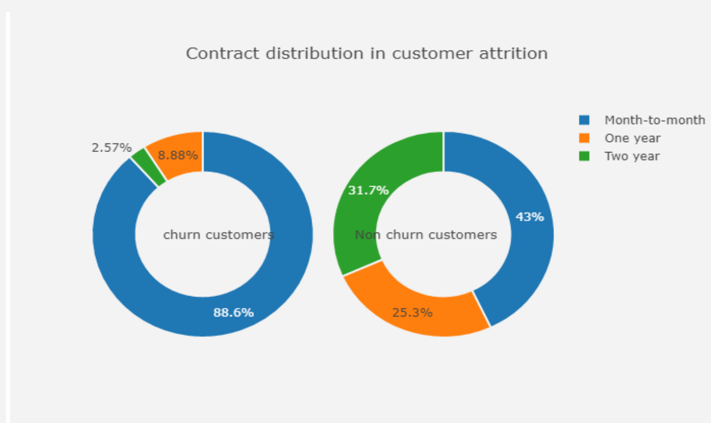

OnlineBackup distribution in customer attrition
No
Yes
churn customers
28%
72%
Non churn customers
36.8%
63.2%

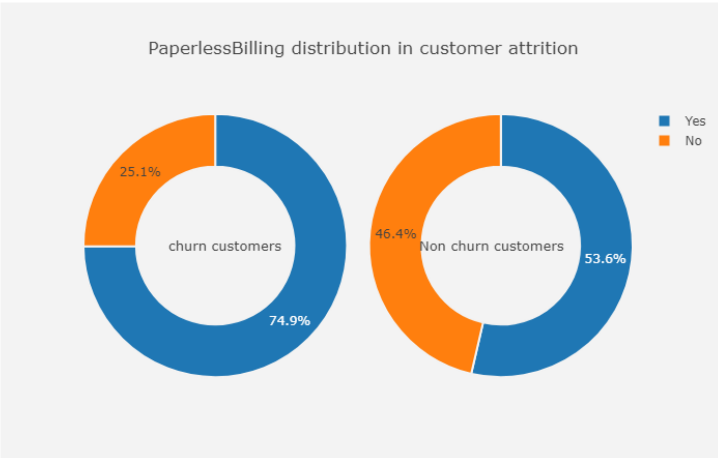

DeviceProtection distribution in customer attrition
No
Yes
churn customers
29.2%
70.8%
Non churn customers
36.3%
63.7%

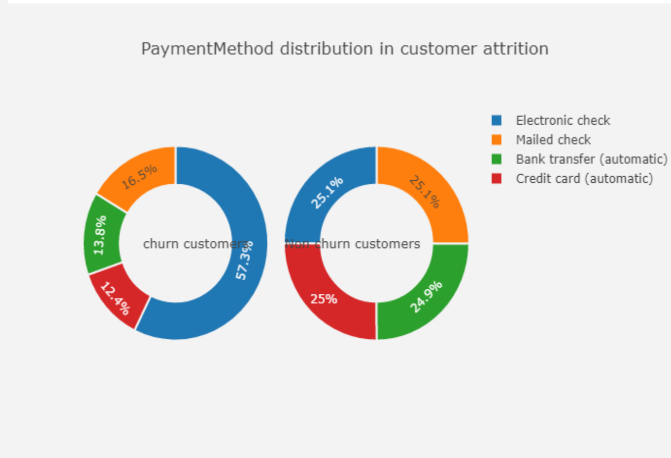

TechSupport distribution in customer attrition
No
Yes
churn customers
16.6%
83.4%
Non churn customers
33.5%
66.5%

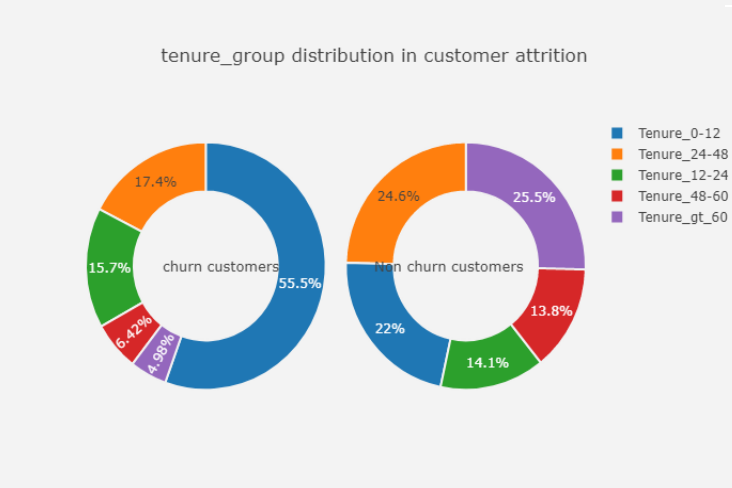

StreamingTV distribution in customer attrition
No
Yes
churn customers
43.6%
56.4%
Non churn customers
36.6%
63.4%

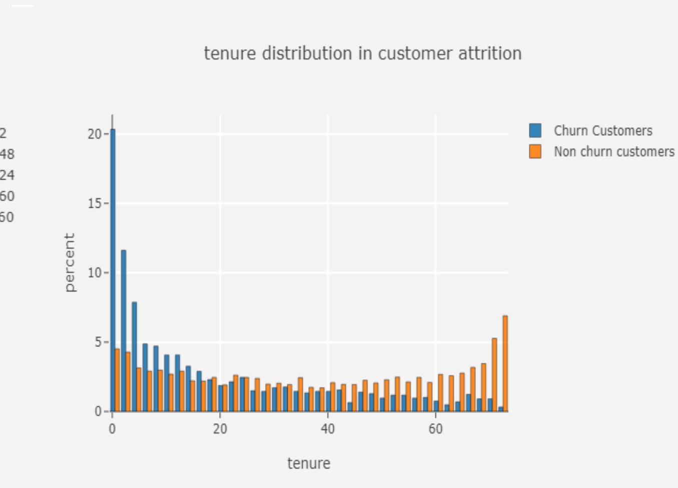

StreamingMovies distribution in customer attrition
No
Yes
churn customers
43.8%
56.2%
Non churn customers
37.1%
62.9%

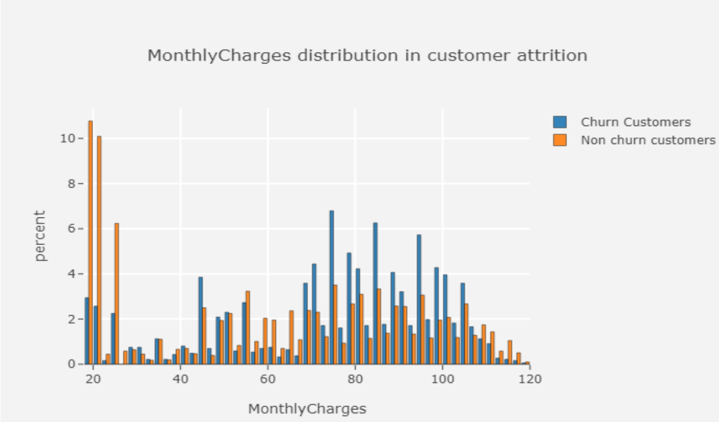

Contract distribution in customer attrition
Month-to-month
One year
Two year
churn customers
2.57%
8.88%
88.6%
Non churn customers
31.7%
43%
25.3%

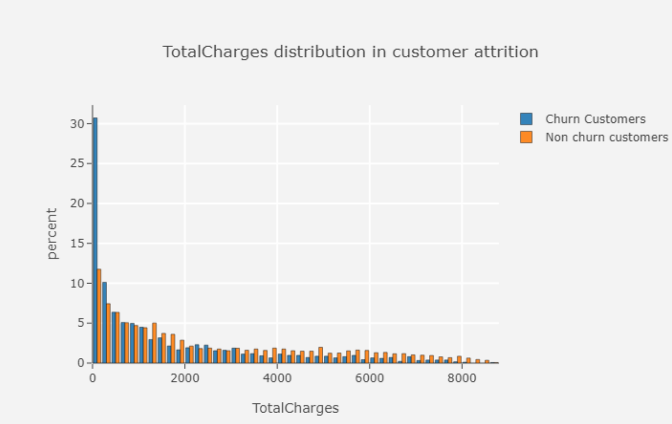

PaperlessBilling distribution in customer attrition
Yes
No
churn customers
Non churn customers
25.1%
74.9%
46.4%
53.6%

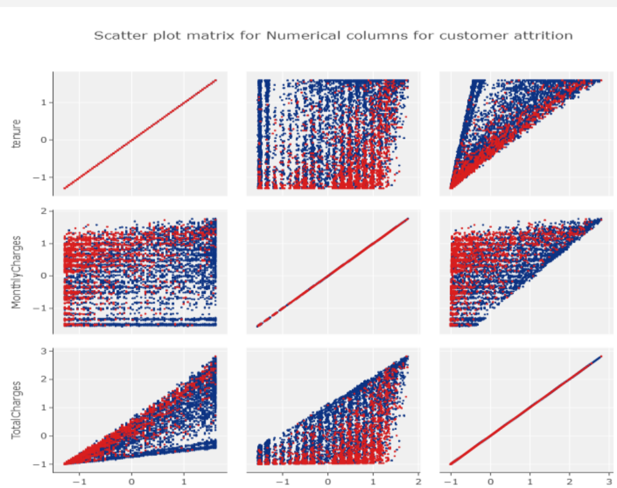

PaymentMethod distribution in customer attrition
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
churn customers
non churn customers
16.5%
13.8%
12.4%
57.3%
25.1%
25.1%
24.9%
25%


tenure_group distribution in customer attrition
Tenure_0-12
Tenure_24-48
Tenure_12-24
Tenure_48-60
Tenure_gt_60
churn customers
Non churn customers
17.4%
15.7%
6.42%
4.98%
55.5%
24.6%
25.5%
13.8%
14.1%
22%


tenure distribution in customer attrition
Churn Customers
Non churn customers
percent
tenure
0
5
10
15
20
40
60

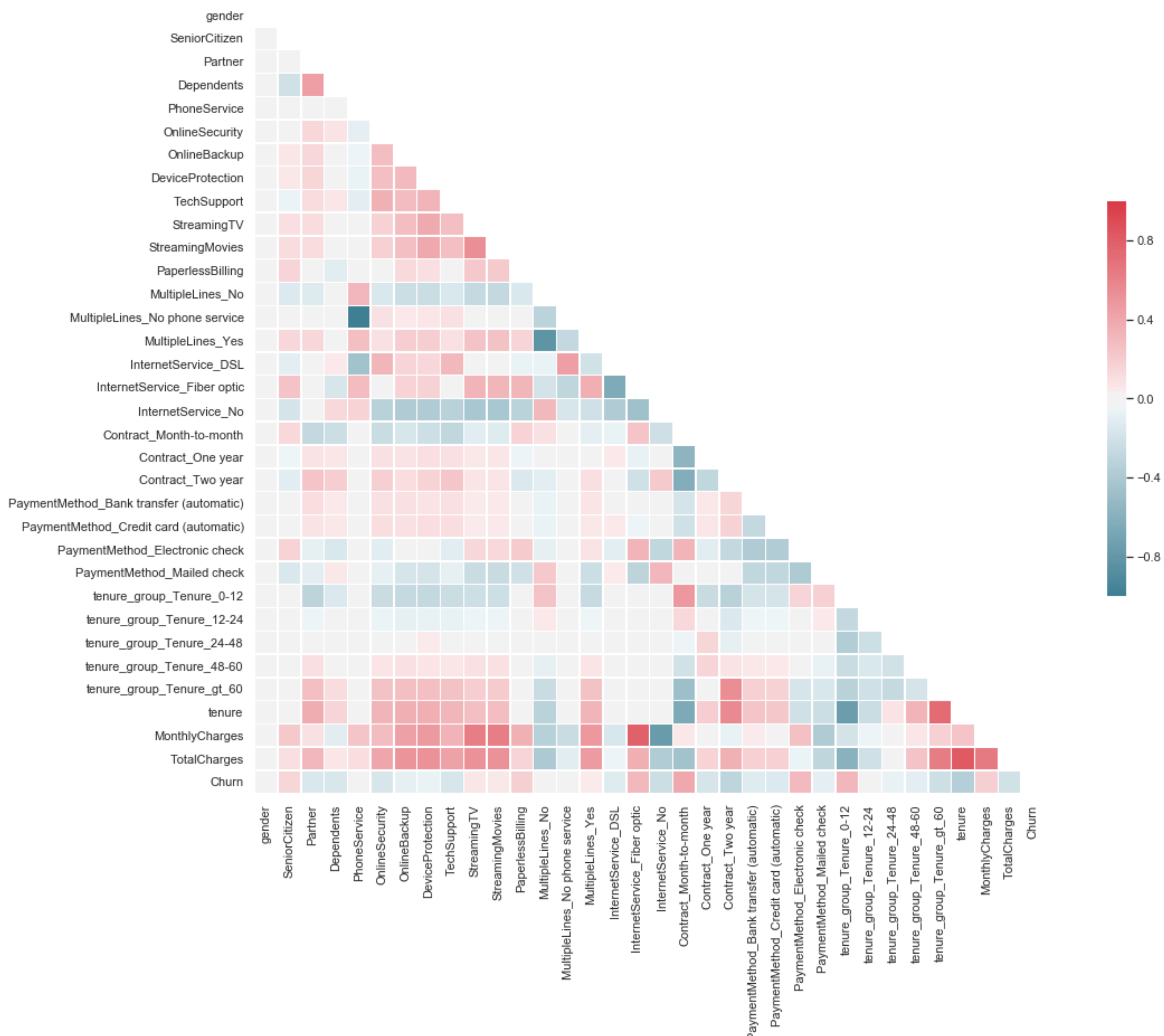

MonthlyCharges distribution in customer attrition
Churn Customers
Non churn customers
percent
MonthlyCharges
0
2
4
6
8
10
20
40
60
80
100
120


TotalCharges distribution in customer attrition
Churn Customers
Non churn customers
percent
TotalCharges
0
5
10
15
20
25
30
2000
4000
6000
8000


Scatter plot matrix for Numerical columns for customer attrition
tenure
MonthlyCharges
TotalCharges

3. Correlation between the features

First let me define the correlation matrix. A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used as a way to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses. Correlation coefficient = 1 means there's a large positive correlation, -1 means a large negative correlation and 0 means there's no correlation between the 2 features or variables.



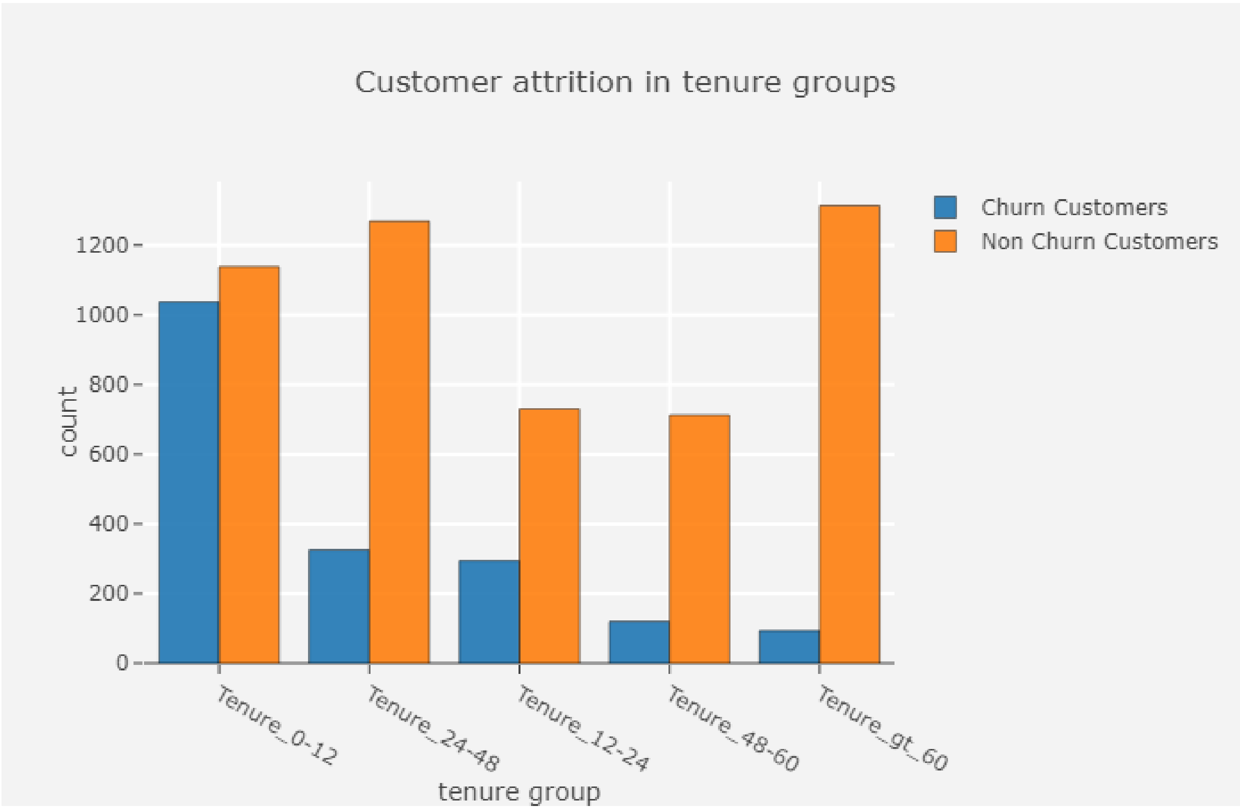
As we see that the gender has almost None correlation to our target Churn

- Almost there's no correlation between our target "Churn" and gender.
- Almost there's no correlation between our target "Churn" and Phone Service.

- Almost there's no correlation between our target "Churn" and Multiple Lines_No.
- Almost there's no correlation between our target "Churn" and Multiple Lines_No_Phone Service.
- Almost there's no correlation between our target "Churn" and Tenure_group_Tenure_12-24.
- There's correlation between gender and the other features.

As a result we'll be Dropping them in the preprocessing

4. Customer Churn to Non-Churn in Tenure Groups



Algorithms and Techniques

SVM:

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (*supervised learning*), the
algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class
lay in either side.

Resampling Data Techniques:

Resampling techniques are a set of methods to either repeat sampling from a given sample or population, or a way to estimate the precision of a statistic. Although the
method sounds daunting, the math involved is relatively simple and only requires a high
school level understanding of algebra.
And check more about resampling techniques here.

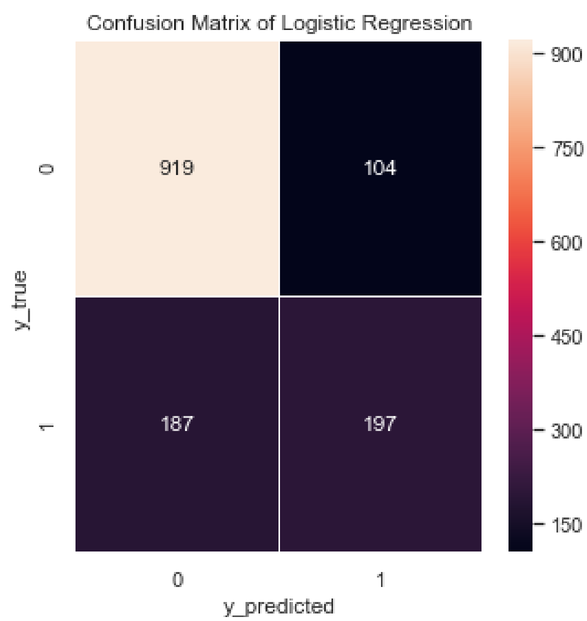
Recall and precision:

Precision-Recall is a useful measure of success of prediction when the classes are very imbalanced. In information retrieval, precision is a measure of result relevancy, while recall is
a measure of how many truly relevant results are returned.
A system with high recall but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels. A system with high
precision but low recall is just the opposite, returning very few results, but most of its predicted labels are correct when compared to the training labels. An ideal system with high
precision and high recall will return many results, with all results labeled correctly.
And here a very good article about Recall and precision score.

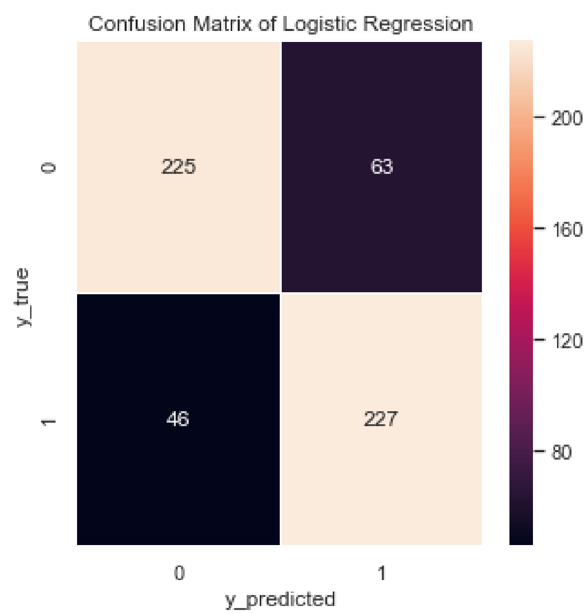
Benchmark：

In This case we'll be using SVM(Support Vector Machine) & Logistic Regression Algorithms for the Classification and we'll be calculating Scores to test for Improvement before and after Sampling & Refinement, as they are to be the best models for benchmarking and training for this case here, we'll be calculating the precision , recall, accuracy & F-beta score for SVM, and Accuracy , F-1, Precision, Recall Score for Logistic Regression that will calculated based on the Confusion Matrix Scores.

Before Sampling



After Sampling



III. Methodology

Data Preprocessing

Data Normalization

4.Methodology

4.1 Data Preprocessing

4.1.1 Data Normalization

In [409]:
```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler

#customer id col
Id_col     = ['customerID']
#Target columns
target_col = ["Churn"]
#categorical columns
cat_cols   = telcom.nunique()[telcom.nunique() < 6].keys().tolist()
cat_cols   = [x for x in cat_cols if x not in target_col]
#numerical columns
num_cols   = [x for x in telcom.columns if x not in cat_cols + target_col + Id_col]
#Binary columns with 2 values
bin_cols   = telcom.nunique()[telcom.nunique() == 2].keys().tolist()
#Columns more than 2 values
multi_cols = [i for i in cat_cols if i not in bin_cols]

#Label encoding Binary columns
le = LabelEncoder()
for i in bin_cols :
    telcom[i] = le.fit_transform(telcom[i])

#Duplicating columns for multi value columns
telcom = pd.get_dummies(data = telcom,columns = multi_cols )

#Scaling Numerical columns
std = StandardScaler()
scaled = std.fit_transform(telcom[num_cols])
scaled = pd.DataFrame(scaled,columns=num_cols)

#dropping original values merging scaled values for numerical columns
df_telcom_og = telcom.copy()
telcom = telcom.drop(columns = num_cols,axis = 1)
telcom = telcom.merge(scaled,left_index=True,right_index=True,how = "left")
```

Loading Data After Normalization

4.1.2 Loading Data After Normalization

In [410]:
```
#load the dataset after normalization
data = telcom

#display first 5 rows after normalization
data.head()
```

Out[410]:

	customerID	gender	SeniorCitizen	Partner	Dependents	PhoneService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	...	PaymentMethod_E
0	7590-VHVEG	0	0	1	0	0	0	1	0	0	...	
1	5575-GNVDE	1	0	0	0	1	1	0	1	0	...	
2	3668-QPYBK	1	0	0	0	1	1	1	0	0	...	
3	7795-CFOCW	1	0	0	0	0	1	0	1	1	...	
4	9237-HQITU	0	0	0	0	1	0	0	0	0	...	

5 rows × 35 columns

Moving Churn Column to Last for Correlation Matrix Readness Ease

4.1.3 Moving Churn Column to Last for Correlation Matrix Readness Ease

In [411]:
```
#data3 = data.columns.tolist()
#print(data3)

data = data[['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'OnlineSecurity', 'OnlineBackup',
dat = data
```

Data Cleaning & Refinement of Un-necessary attributes

6. Methodology - 2

6.1 Data Preprocessing - 2

6.2 Data Cleaning & Refinement of Un-necessary atrributes

In [416]:
```
dat.drop(['customerID','gender','PhoneService','MultipleLines_No', 'MultipleLines_No phone service','tenure_group_Tenure_12-24'].
dat.head()
```

Out[416]:

	SeniorCitizen	Partner	Dependents	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	PaperlessBilling	...	Payn
0	0	1	0	0	1	0	0	0	0	1	...	
1	0	0	0	1	0	1	0	0	0	0	...	
2	0	0	0	1	1	0	0	0	0	1	...	
3	0	0	0	1	0	1	1	0	0	0	...	
4	0	0	0	0	0	0	0	0	0	1	...	

5 rows × 29 columns

Data Shuffling & Splitting

6.3 Data Shuffling & Splitting

```
# Split the data into features and target label
features = dat.drop(['Churn'], axis =1)
target = dat['Churn']

# Split the features into training and testing sets
# Import train_test_split
from sklearn.model_selection import train_test_split

# Split the 'features' and 'target' data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features,
                                                    target,
                                                    test_size = 0.20,
                                                    random_state = 200)

# Show the results of the split
print("Training set has {} samples.".format(X_train.shape[0]))
print("Testing set has {} samples.".format(X_test.shape[0]))
```

```
Training set has 5625 samples.
Testing set has 1407 samples.
```

Data Resampling

7. Results

7.1 Model Evaluation

7.1.1 Data Resampling

```
# store No. of Churn and indices
Churn_records = dat['Churn'].sum()
Churn_indices = np.array(dat[dat.Churn == 1].index)

# Picking the indices of the normal classes
normal_indices = dat[dat.Churn == 0].index

# Out of the indices we picked, randomly select number of normal records = number of fraud records
random_normal_indices = np.random.choice(normal_indices, Churn_records, replace = False)
random_normal_indices = np.array(random_normal_indices)

# Merge the 2 indices
under_sample_indices = np.concatenate([Churn_indices,random_normal_indices])

# Copy under sample dataset
under_sample_data = dat.iloc[under_sample_indices,:]

# Split data into features and target labels
features_undersample = under_sample_data.drop(['Churn'], axis = 1)
target_undersample = under_sample_data['Churn']

# Show ratio
print("Percentage of Churn Customers: ", under_sample_data.Churn[under_sample_data['Churn'] == 0].count())
print("Percentage of No-Churn Customers: ", under_sample_data.Churn[under_sample_data['Churn'] == 1].count())
print("Total number of All Customers of Churn & Non-Ch in resampled data: ", under_sample_data['Churn'].count())
```

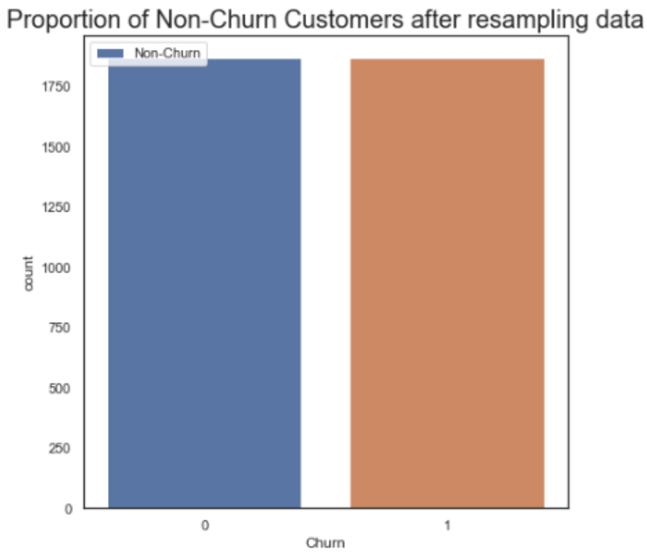
```
Percentage of Churn Customers:  1869
Percentage of No-Churn Customers:  1869
Total number of All Customers of Churn & Non-Ch in resampled data:  3738
```

Data Plotting after Resampling

7.1.2 Data Plotting after Resampling

```
under_sample_Churn_Real = [under_sample_data.Churn[under_sample_data['Churn'] == 0].count(), Churn_records]

# Plot the proportion
plt.subplots(figsize = (7, 7))
plt.title("Proportion of Non-Churn Customers after resampling data", size = 20)
ax = sns.countplot(x = under_sample_data['Churn'], data= under_sample_data)
ax.legend(labels=['Non-Churn', 'Churn'], loc = 'upper left')
```

```
<matplotlib.legend.Legend at 0x27c930c0630>
```



Shuffle and Split Data after Resampling

7.1.3 Shuffle and Split Data after Resampling

```
# Split the 'features_undersample' and 'target_anderSample' data into training and testing sets
X_train_sampled, X_test_sampled, y_train_sampled, y_test_sampled = train_test_split(features_undersample,
                                                                                    target_undersample,
                                                                                    test_size = 0.15,
                                                                                    random_state = 25)

# Show the results of the split
print("Training set has {} samples.".format(X_train_sampled.shape[0]))
print("Testing set has {} samples.".format(X_test_sampled.shape[0]))
```

```
Training set has 3177 samples.
Testing set has 561 samples.
```

Implementation

Before Sampling & Reshuffling

Fit & Train SVM

6.4 Implementation

6.4.1 Fit & Train SVM

```
from sklearn.metrics import fbeta_score, accuracy_score
from sklearn.svm import SVC

# Create an object from Support Vector Machine Classifier with random state
clf = SVC(random_state=2540)

# Fit the classifier
clf.fit(X_train, y_train)

# Predict
prediction_train = clf.predict(X_train)
prediction_test = clf.predict(X_test)

# Calculate accuracy score
acc_train = accuracy_score(y_train, prediction_train)
acc_test = accuracy_score(y_test, prediction_test)

# Calculate F-beta score
f_train = fbeta_score(y_train, prediction_train, beta=0.5)
f_test = fbeta_score(y_test, prediction_test, beta=0.5)

#print the results
print("Accuracy score on Training set: {:.2f}%".format(acc_train*100))
print("Accuracy score on Testing set: {:.2f}%".format(acc_test*100))
print("\nF-beta score on Training set: {:.4f}".format(f_train))
print("F-beta score on Testing set: {:.4f}".format(f_test))
```

```
C:\Anaconda3\lib\site-packages\sklearn\svm\base.py:196: FutureWarning:

The default value of gamma will change from 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamm
a explicitly to 'auto' or 'scale' to avoid this warning.
```

```
Accuracy score on Training set: 79.89%
Accuracy score on Testing set: 78.54%

F-beta score on Training set: 0.6169
F-beta score on Testing set: 0.5988
```

1-SVM Scoring on Whole Dataset

2-Fit & Train Logistic Regression with Accuracy Score

6.4.2 SVM Scoring on Whole Dataset

```
from sklearn.metrics import recall_score, precision_score

recall_train = recall_score(y_train, prediction_train)
recall_test = recall_score(y_test, prediction_test)

precision_train = precision_score(y_train, prediction_train)
precision_test = precision_score(y_test, prediction_test)

print("Recall score on training set: {:.4f}".format(recall_train))
print("Recall score on testing set: {:.4f}".format(recall_test))
print("\nprecision score on training set: {:.4f}".format(precision_train))
print("precision score on testing set: {:.4f}".format(precision_test))
```

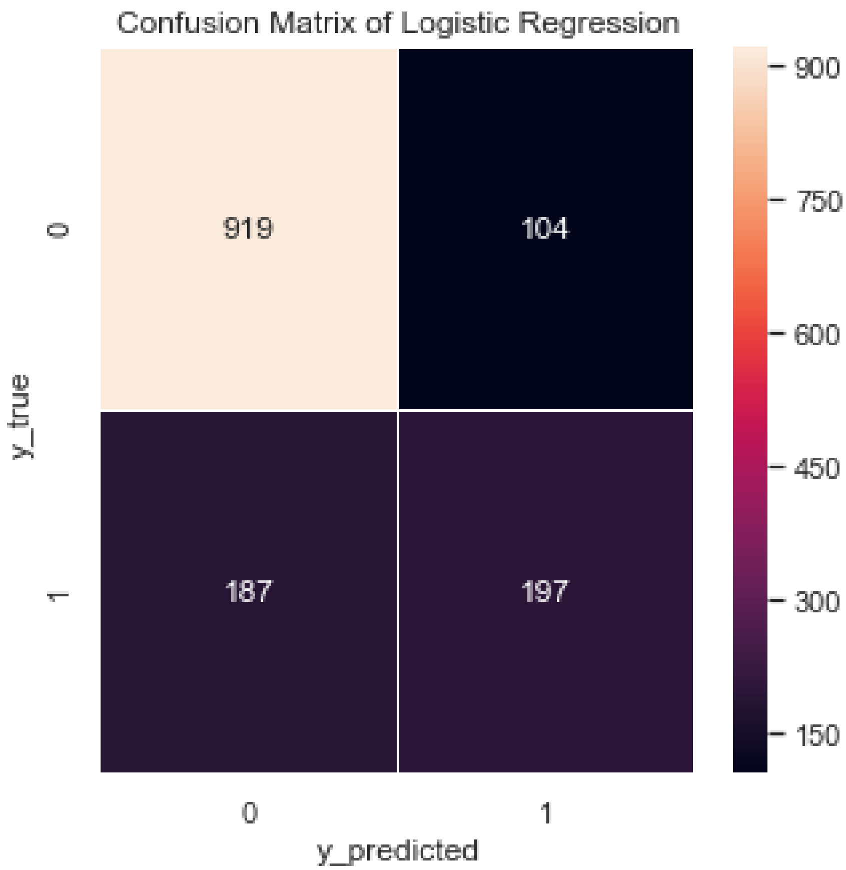
```
Recall score on training set: 0.4424
Recall score on testing set: 0.4167

precision score on training set: 0.6844
precision score on testing set: 0.6723
```

6.4.3 Fit & Train Logistic Regression with Accuracy Score

```
# %%Logistic regression classification
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression()
lr_model.fit(X_train,y_train)
accuracy_lr = lr_model.score(X_test,y_test)
print("Logistic Regression accuracy is :",accuracy_lr)
```

```
Logistic Regression accuracy is : 0.7931769722814499
```

Calculate Confusion Matrix for Log Regression



Description of Confusion Matrix Values

6.4.5 Description of Confusion Matrix Values

```
conf = confusion_matrix(y_test,lr_model.predict(X_test))
TN = conf[0,0]
FP = conf[0,1]
FN = conf[1,0]
TP = conf[1,1]

print("TN = ",TN)
print("FP = ",FP)
print("FN = ",FN)
print("TP = ",TP)
```

```
TN =  919
FP =  104
FN =  187
TP =  197
```

Descriptive Scoring for Logistic Regression

6.4.6 Descriptive Scoring for Logistic Regression

```
report = classification_report(y_test, lr_model.predict(X_test))
print(report)
```

```
              precision    recall  f1-score   support

           0       0.83      0.90      0.86      1023
           1       0.65      0.51      0.58       384

   micro avg       0.79      0.79      0.79      1407
   macro avg       0.74      0.71      0.72      1407
weighted avg       0.78      0.79      0.78      1407
```

Refinement

As I describe above working on imbalanced data is a huge mistake and the results of evaluating metrics of benchmark shows that, after resampling the data using undersampling the results improved as I show above Recall score is above 70%.
There're other techniques to improve the result like using K-fold and Grid-Search to pick the best hyper-parameters.

IV. Results

Model Evaluation and Validation

Data Resampling

7. Results

7.1 Model Evaluation

7.1.1 Data Resampling

```
# store No. of Churn and indices
Churn_records = dat['Churn'].sum()
Churn_indices = np.array(dat[dat.Churn == 1].index)

# Picking the indices of the normal classes
normal_indices = dat[dat.Churn == 0].index

# Out of the indices we picked, randomly select number of normal records = number of fraud records
random_normal_indices = np.random.choice(normal_indices, Churn_records, replace = False)
random_normal_indices = np.array(random_normal_indices)

# Merge the 2 indices
under_sample_indices = np.concatenate([Churn_indices,random_normal_indices])

# Copy under sample dataset
under_sample_data = dat.iloc[under_sample_indices,:]

# Split data into features and target labels
features_undersample = under_sample_data.drop(['Churn'], axis = 1)
target_undersample = under_sample_data['Churn']

# Show ratio
print("Percentage of Churn Customers: ", under_sample_data.Churn[under_sample_data['Churn'] == 0].count())
print("Percentage of No-Churn Customers: ", under_sample_data.Churn[under_sample_data['Churn'] == 1].count())
print("Total number of All Customers of Churn & Non-Ch in resampled data: ", under_sample_data['Churn'].count())
```

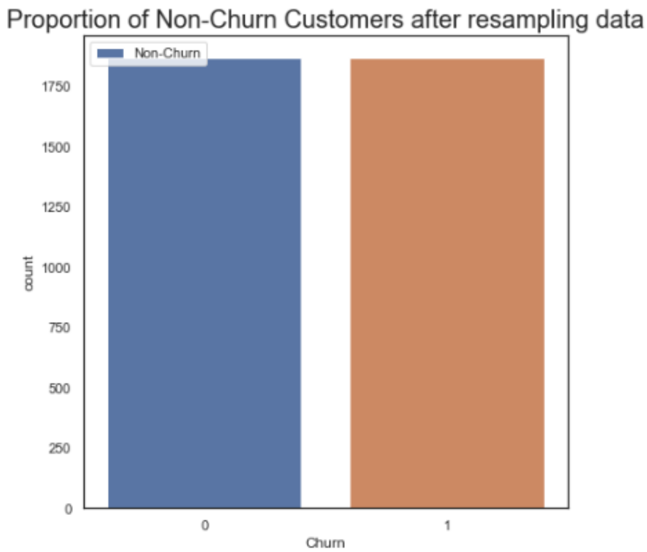
```
Percentage of Churn Customers:  1869
Percentage of No-Churn Customers:  1869
Total number of All Customers of Churn & Non-Ch in resampled data:  3738
```

Data Plotting after Resampling

7.1.2 Data Plotting after Resampling

```
under_sample_Churn_Real = [under_sample_data.Churn[under_sample_data['Churn'] == 0].count(), Churn_records]

# Plot the proportion
plt.subplots(figsize = (7, 7))
plt.title("Proportion of Non-Churn Customers after resampling data", size = 20)
ax = sns.countplot(x = under_sample_data['Churn'], data= under_sample_data)
ax.legend(labels=['Non-Churn', 'Churn'], loc = 'upper left')
```

```
<matplotlib.legend.Legend at 0x27c930c0630>
```



Shuffle and Split Data after Resampling

7.1.3 Shuffle and Split Data after Resampling

```
# Split the 'features_undersample' and 'target_anderSample' data into training and testing sets
X_train_sampled, X_test_sampled, y_train_sampled, y_test_sampled = train_test_split(features_undersample,
                                                                                     target_undersample,
                                                                                     test_size = 0.15,
                                                                                     random_state = 25)

# Show the results of the split
print("Training set has {} samples.".format(X_train_sampled.shape[0]))
print("Testing set has {} samples.".format(X_test_sampled.shape[0]))
```

```
Training set has 3177 samples.
Testing set has 561 samples.
```

Justification

As we see there is a slight much improvement in the SVM Scores after Sampling & Shuffling

As F-beta doesn't seem to be affected much which in overall shows large improvement that could be further increased with methods as fore mentioned like K-fold Cross Validation & Grid Search.

As for the Logistic Regression we can see that the sampling has helped improve a distant large amount in the following precision, F-1 Score, and recall Scores especially in the prediction section of the Churned Customers, as for the accuracy doesn't seem to be affected much, with overall the scores could be increased further bit with the same fore mentioned methods.

Bench Mark Models

SVM:

```
Accuracy score on Training set: 79.89%
Accuracy score on Testing set: 78.54%

F-beta score on Training set: 0.6169
F-beta score on Testing set: 0.5988


Recall score on training set: 0.4424
Recall score on testing set: 0.4167

precision score on training set: 0.6844
precision score on testing set: 0.6723
```

Logistic Regression:

6.4.3 Fit & Train Logistic Regression with Accuracy Score

```
# %%Logistic regression classification
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression()
lr_model.fit(X_train,y_train)
accuracy_lr = lr_model.score(X_test,y_test)
print("Logistic Regression accuracy is :",accuracy_lr)
```

```
Logistic Regression accuracy is : 0.7931769722814499
```

```
TN =  919
FP =  104
FN =  187
TP =  197
```

```
              precision    recall  f1-score   support

           0       0.83      0.90      0.86      1023
           1       0.65      0.51      0.58       384

   micro avg       0.79      0.79      0.79      1407
   macro avg       0.74      0.71      0.72      1407
weighted avg       0.78      0.79      0.78      1407
```

Final Models

SVM:

```
Recall score on training set of sampled data: 0.8133
Recall score on testing set of sampled data: 0.8242

precision score on training set: 0.7451
precision score on testing set: 0.7679
```

```
Recall score on training set: 0.8189
Recall score on testing set: 0.7995

precision score on training set: 0.7451
precision score on testing set: 0.7679
```

Scoring on Whole Dataset:

```
Accuracy score on training set: 73.76%
Accuracy score on testing set: 73.70%

F-beta score on trainin set: 0.5440
F-beta score on testing set: 0.5514

Precision score on trainin set: 0.5019
Precision score on testing set: 0.5117
```

Logistic Regression

7.2.4 Fit & Train Log Reg after Resampling with Accuracy Score

```
# %%Logistic regression classification
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression()
lr_model.fit(X_train_sampled,y_train_sampled)
accuracy_lr = lr_model.score(X_test_sampled,y_test_sampled)
print("Logistic Regression accuracy is :",accuracy_lr)
```

```
Logistic Regression accuracy is : 0.8057040998217468
```

```
TP =  225
FP =  63
TN =  46
FN =  227
```

	precision	recall	f1-score	support
0	0.83	0.78	0.81	288
1	0.78	0.83	0.81	273
micro avg	0.81	0.81	0.81	561
macro avg	0.81	0.81	0.81	561
weighted avg	0.81	0.81	0.81	561

V. Conclusion

Free-Form Visualization

All needed visualization is attached in above sections

Reflection

Improvement

Well as pre-Discussed there are couple of improvement types such as K-fold & Grid-Search that suit up to improve both models and as well help in pushing up the recall score but since we are working on imbalanced data we focus mostly on the F-1 Score which as far we got we achieved some high improvements.