Machine Learning Engineer Nanodegree

Capstone Project

Mohammed Ashraf Farouq Ali
June 11th, 2019

I. Definition

Project Overview

Context

Customer churn is defined by losing customers or clients.

Which as in overall companies like mention here telephone companies pay numerous amount of money to explore the possible reasons and causes to customers churning from the company as for voluntary causes such as customer decides to leave or go for another service provider or involuntary cause which can be due to natural causes that prevent the customer from continuing his/her subscriptions, as in overall spending capital over how to keep customers satisfied and committing to the company is often less costly than exploring new ways to get new customers to replace old ones.

What we'll be Discussing through this project is the various type of customers and how the process of their subscription went through out and how that affects the future state of customers on the long term and understand analysis of the problem.

Acknowledgements

The dataset has been collected from an **[IBM Sample Data Sets]**

- *Check more about the dataset here.*
- *Related research paper here.*

DataSet & Input:

As for our input we'll be using the Churn variables to compare results, and as well we'll split data and shuffle it for training purposes after analysis of what features are more relevant to our process.

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 imbalanced whereas, In information retrieval, precision is a measure of result relevancy that helps define correlation proportions , while recall is a measure of how many truly relevant results are returned moreover a model with high recall but low precision returns many results, which most of its predicted labels are incorrect results when compared to the training labels. In more contrast way a model with high precision but low recall is just the opposite, returning very few results of labels, but most of its predicted labels are correct of the comparison of the training labels. A mostly perfect model where with high Precision & Recall would return as much results as possible with mostly all of them are correct compared to the Labels.

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 as It takes into consideration both the Recall and Precision metrics that makes it check the credibility of both metrics results.

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

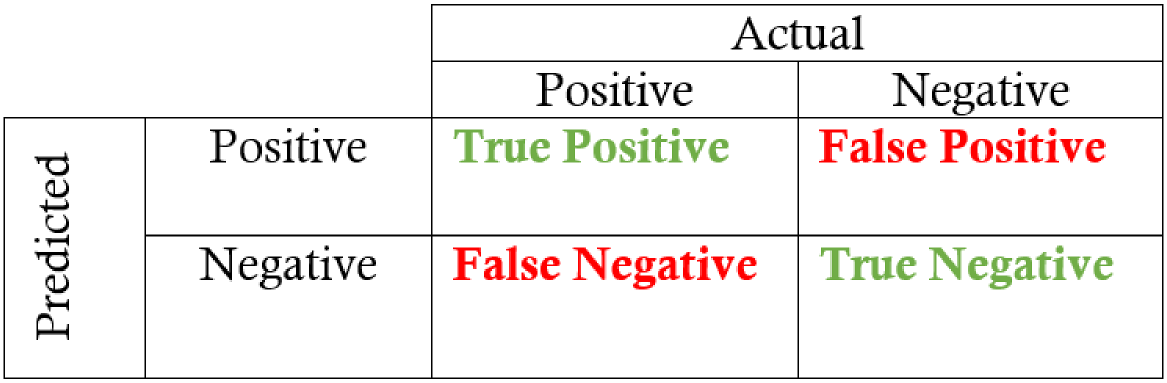
Precision: high precision means that an algorithm returned mostly more relevant results to the labels than irrelevant

Recall: high recall means that an algorithm has returned almost of the relevant results possible.

Good article and info here.

Confusion Matrix for calculating Logistic Regression Scores: A confusion matrix is a table that is used to define the performance of a model or a classifier on a data that is mostly known. While it can somehow confuse readers it can be easy to understand them, where it has main aspects which divide into True Positives, True Negative, False Positive, False Negative. Quick easy article here.

F-1 Score: is a measure of a test's accuracy, as it takes into consideration both precision and the recall scores of a test to determine the accuracy whereas, P is the amount of correct positives divided by amount of all positives returned by the classifier or model, moreover R is amount of relevant results divided over the amount of all relevant results. More Info here.



		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

$$recall = \frac{true\ positives}{true\ positives\ + false\ negatives} \qquad precision = \frac{true\ positives}{true\ positives + false\ positives}$$

We tend to use metrics as it helps tell us to try maximize our recall and precision moreover the accuracy scores, or the ability of a model to find all the relevant cases within a dataset. Whereas a simple definition of recall is the number of true positives divided by the number of true positives plus the number of false negatives. True positives are labels that are classified as positive by the model that are actually positive (which is actually right), and false negatives are labels that the model identifies as negative that actually are positive (which means the model understood them to be wrong while they are right).

As in over all the use of the fore mentioned metrics is to find all possible cases of the predictions and validate the overall probability of getting the probable right prediction.

II. Analysis

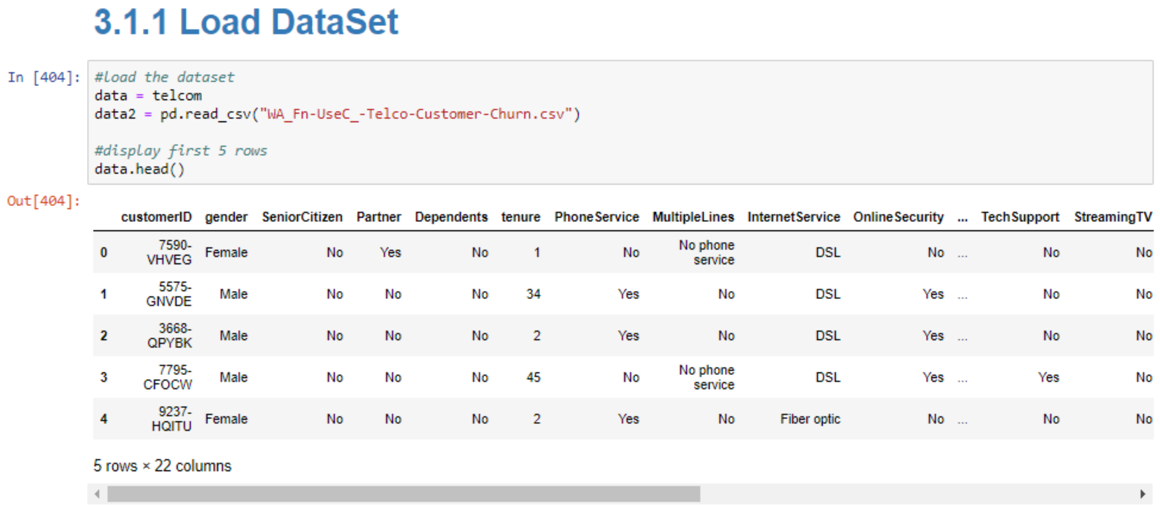
Data Exploration

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

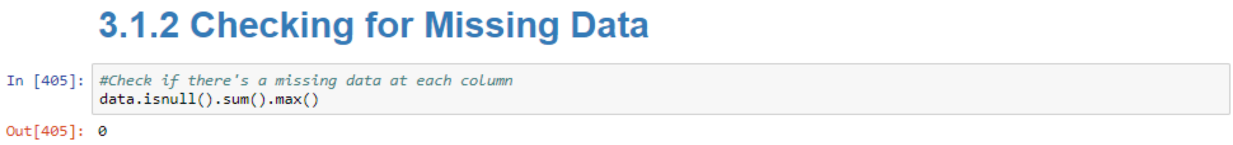
```
In [404]: #load the dataset
          data = telcom
          data2 = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")

          #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

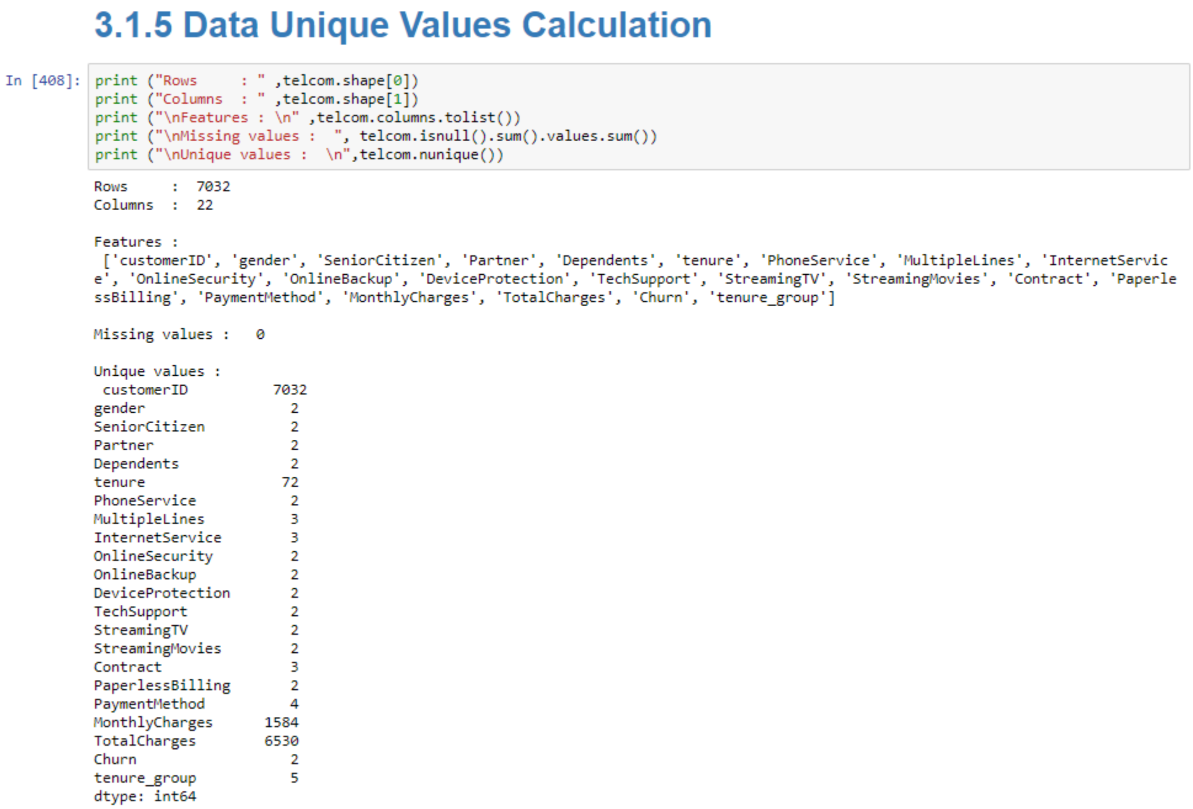
5 rows × 22 columns

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
```

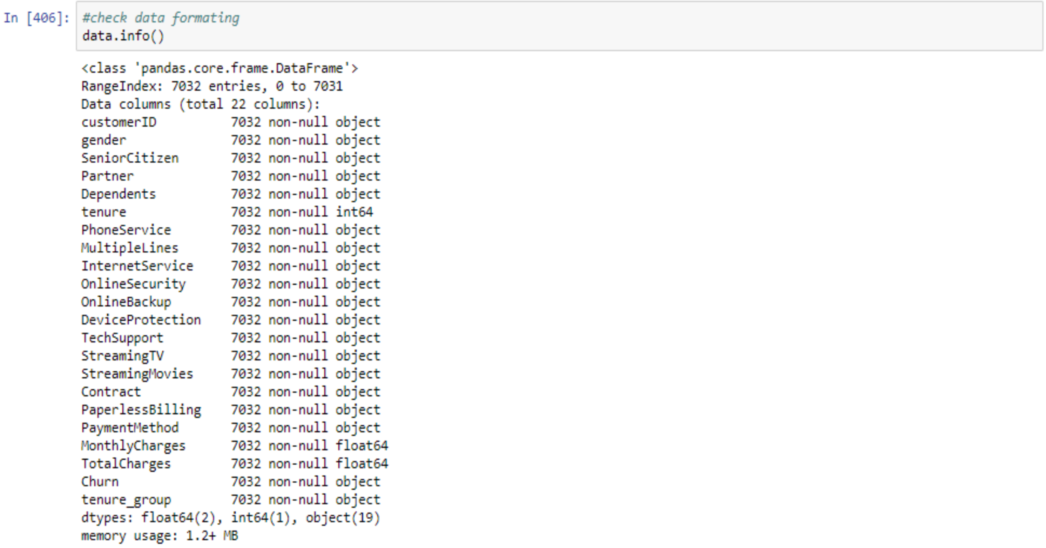


3.1.5 Data Unique Values Calculation

```
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

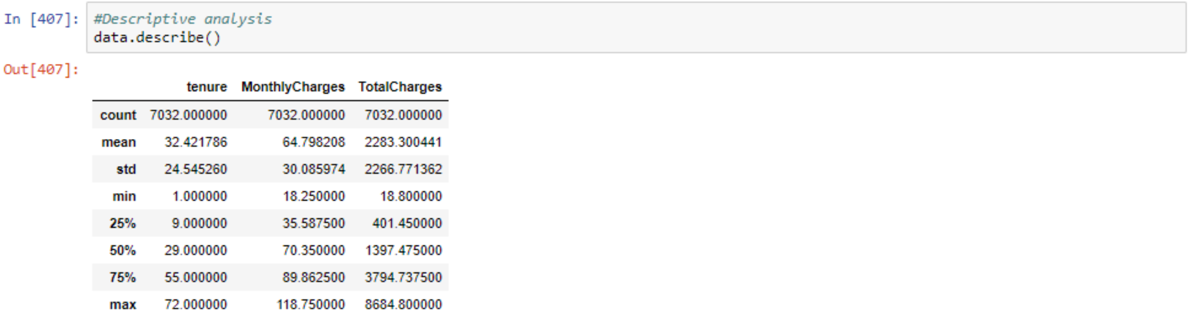
3.1.3 Data Format Description



```
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



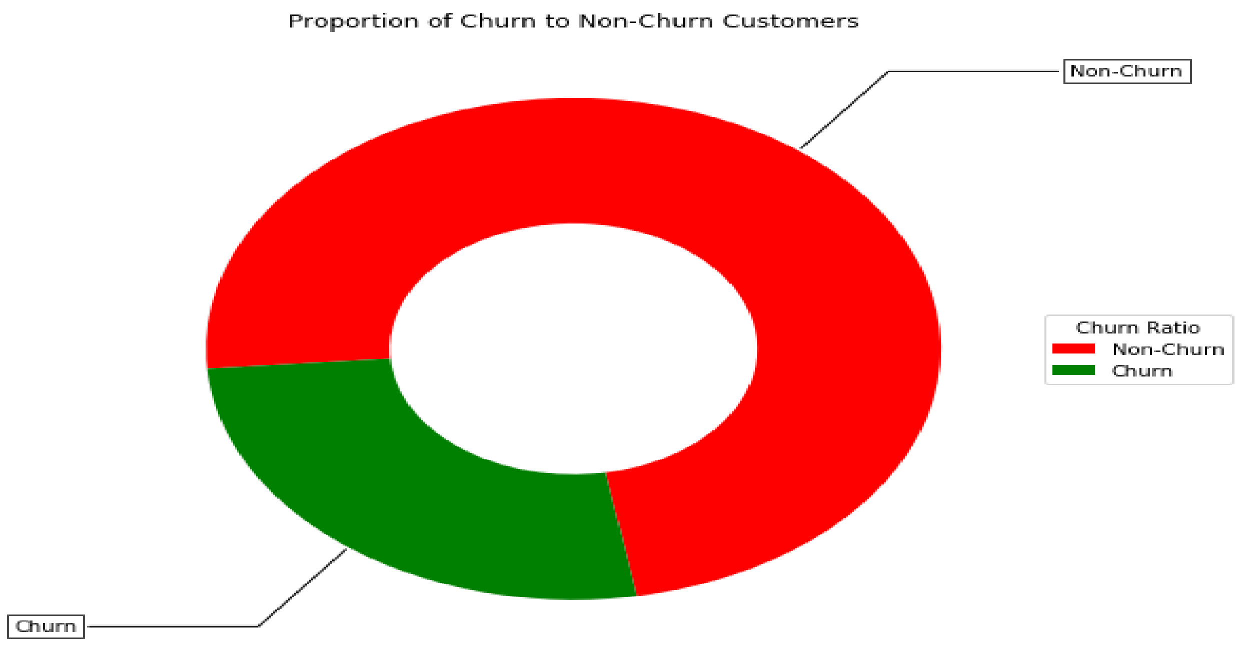
```
In [407]: #Descriptive analysis
          data.describe()
```

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

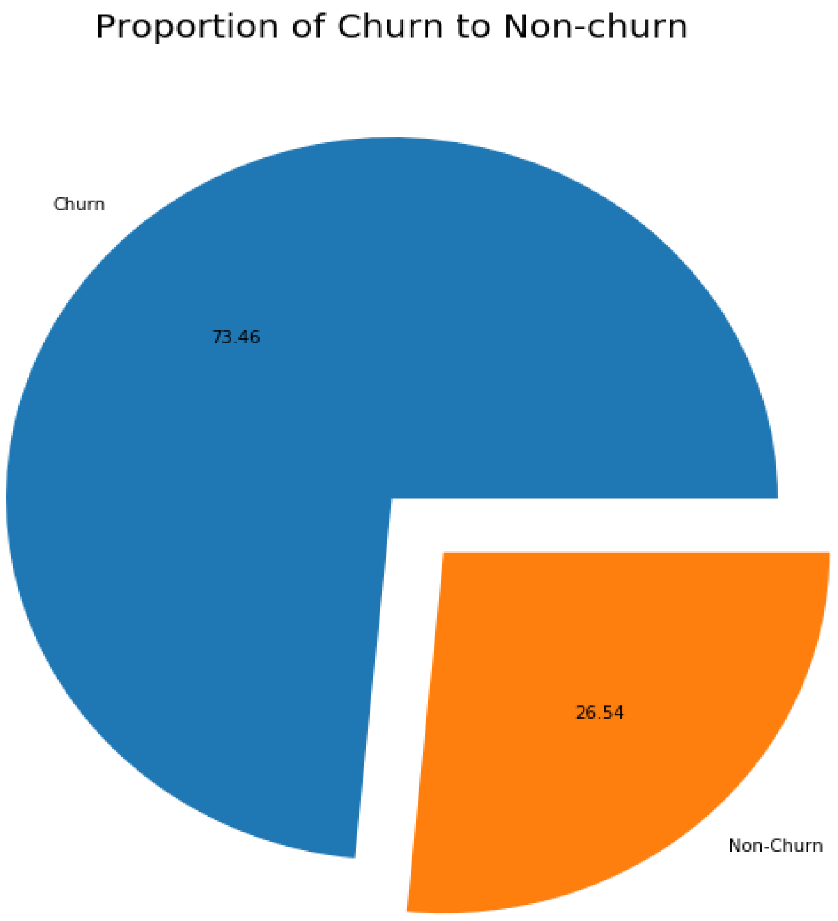
Exploratory Visualization

1.Churn to Non-Churn Proportion

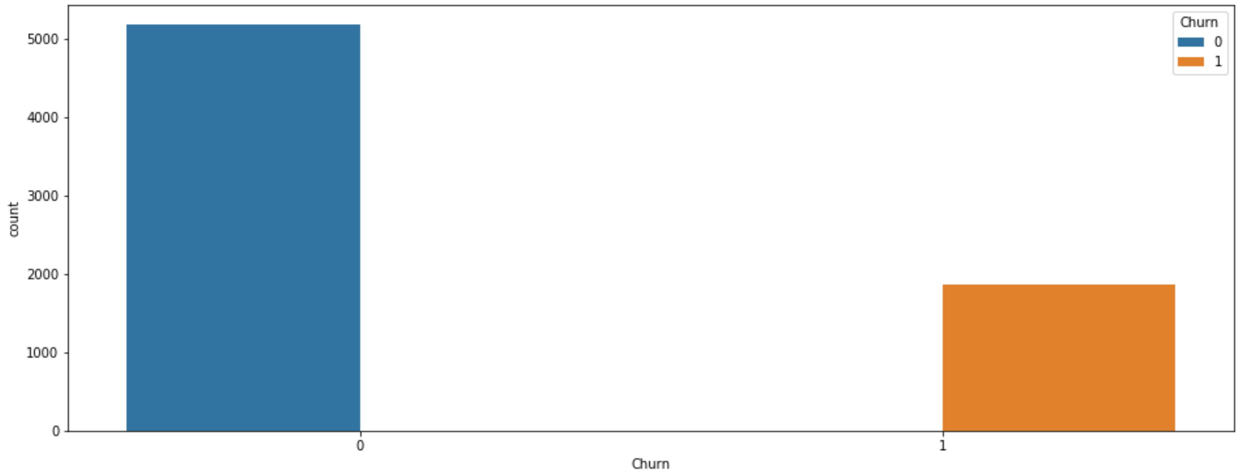


- The data is mostly unbalanced, Evaluate modeling on unbalanced data is a terrible mistake. Will see later why and how to work with unbalanced data.

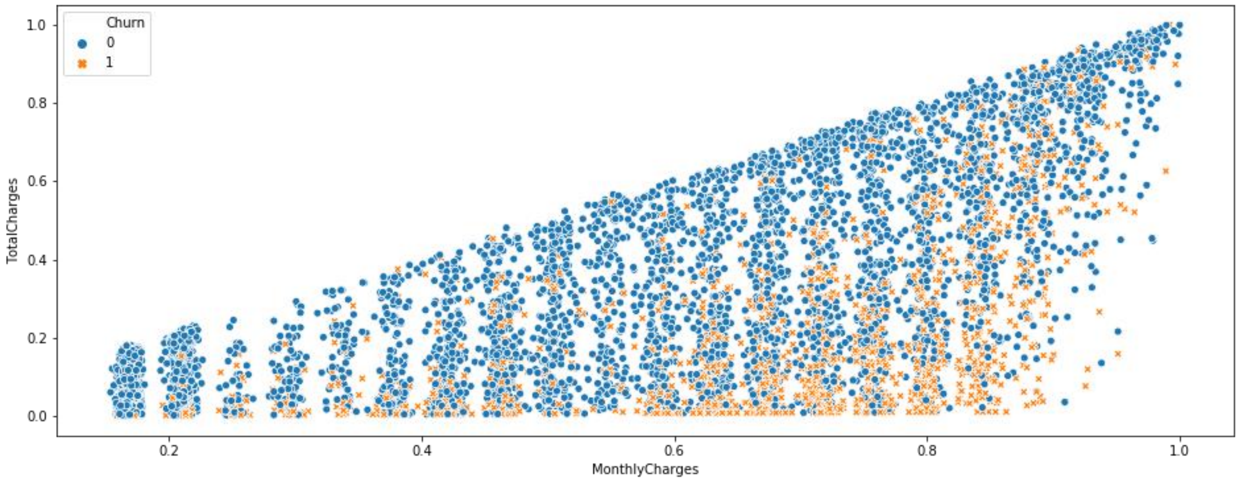
2.Statistical Analysis in Customer Churning

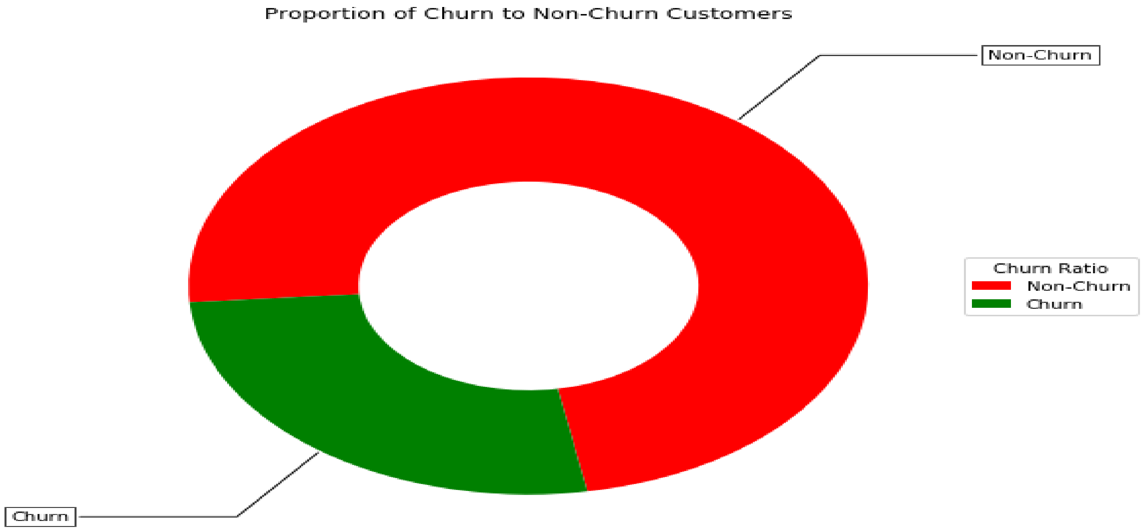


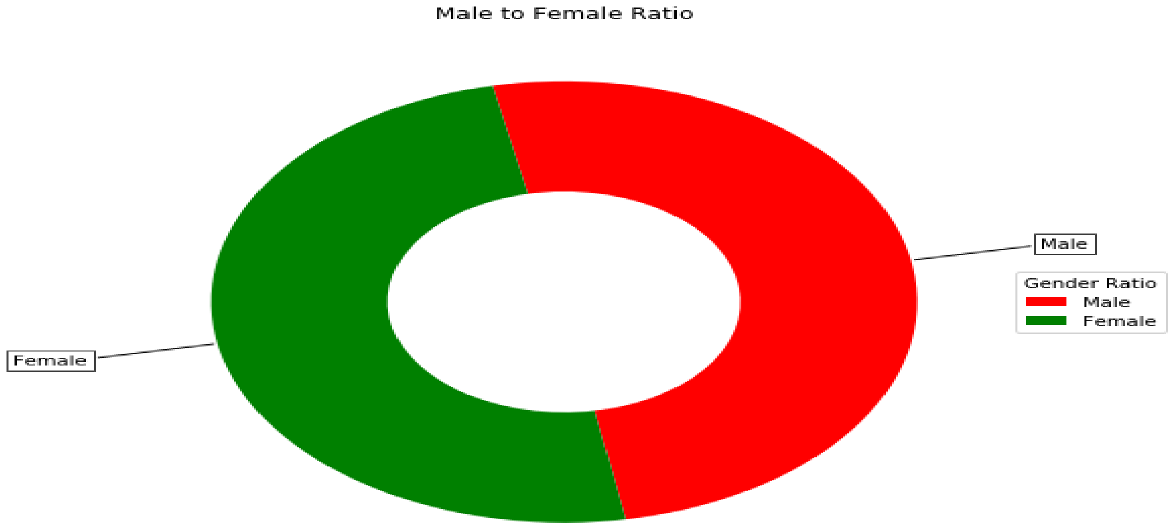
Bar Chart description of Non-Churn to Churn Customers

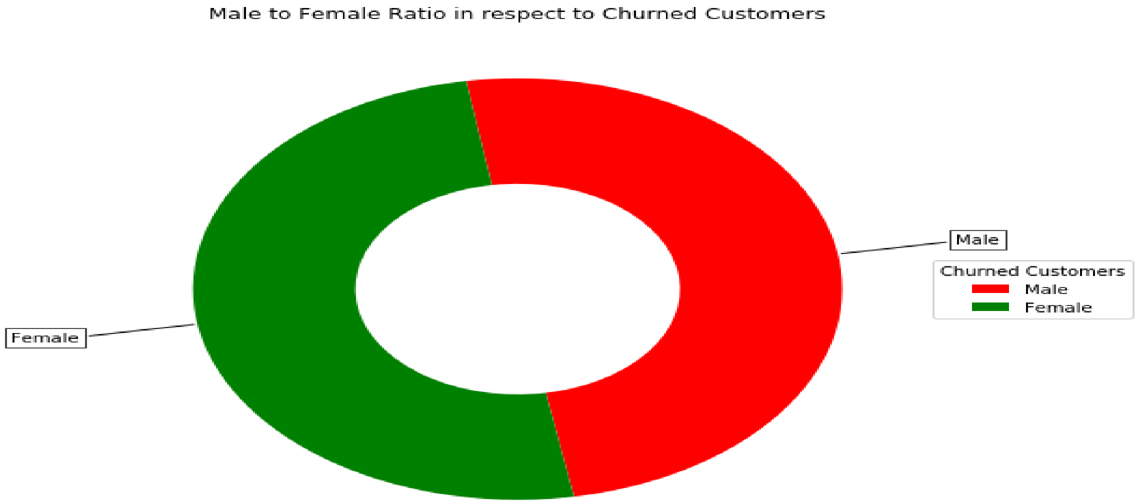


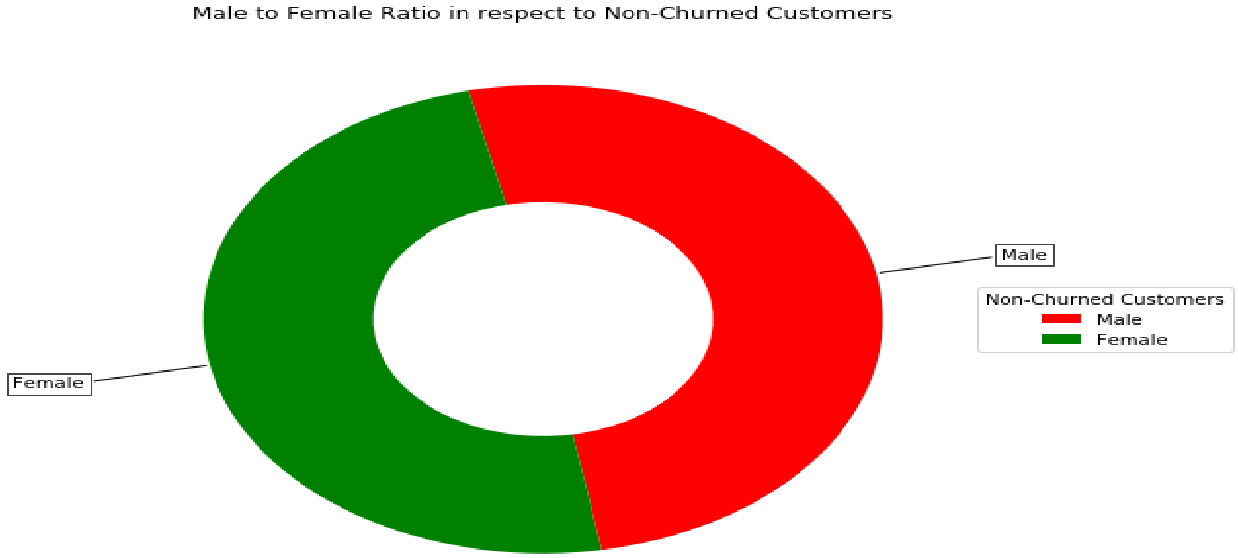
Scatter Plotting ratio of Churn/Non-Churn in respect Monthly & Total Charges

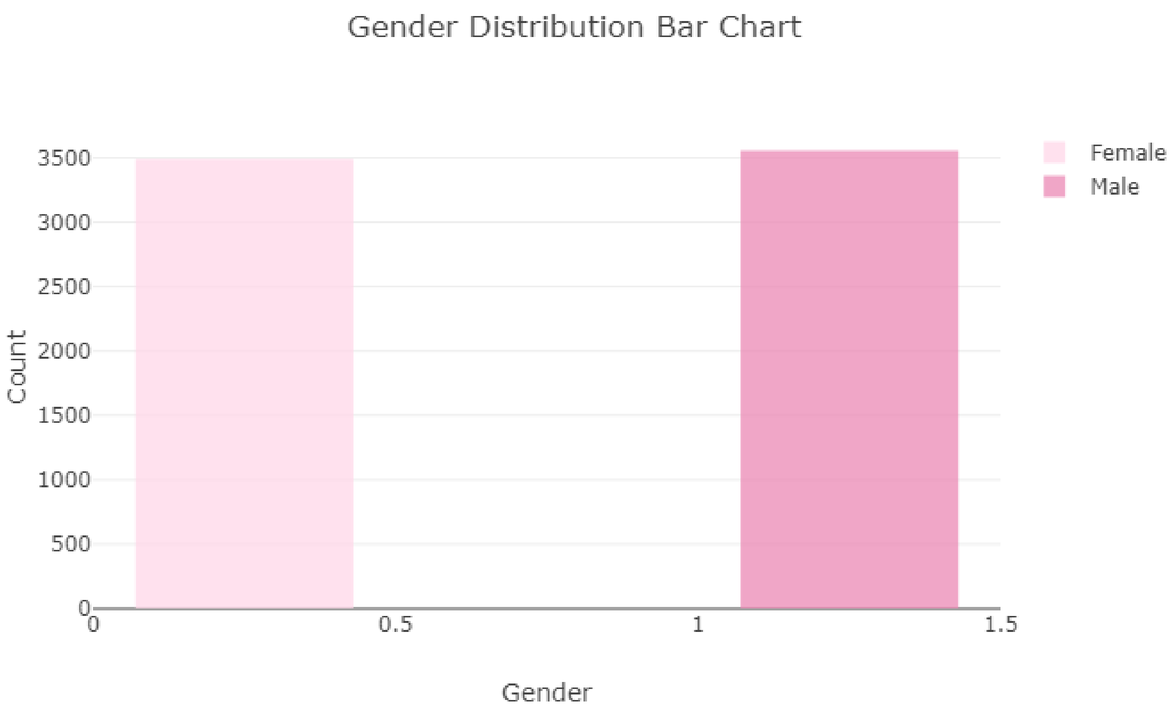


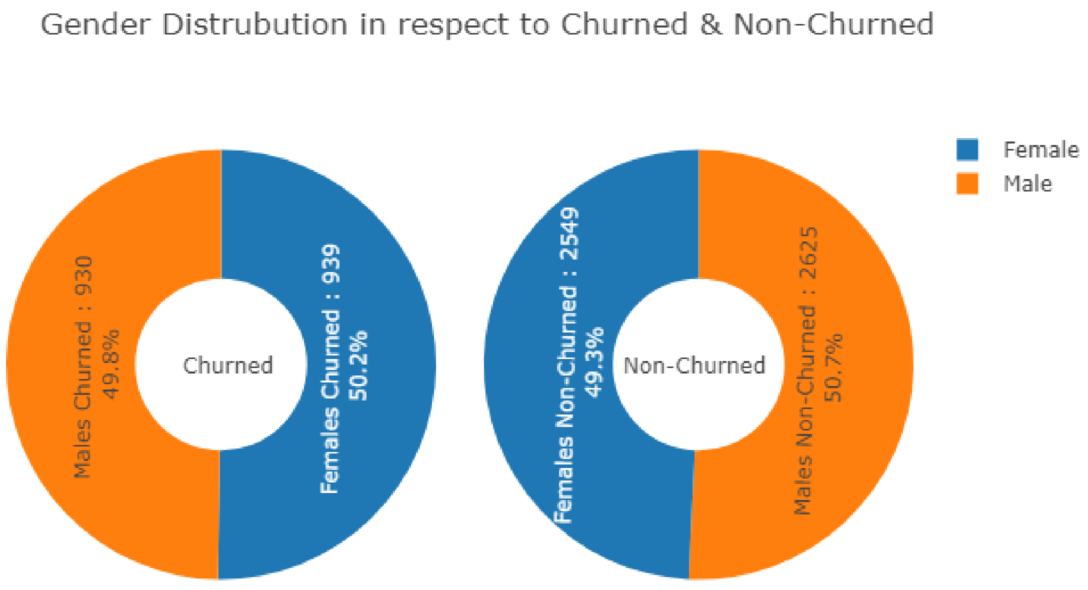



Male to Female Ratio
Male
Female
Gender Ratio
🟥 Male
🟩 Female

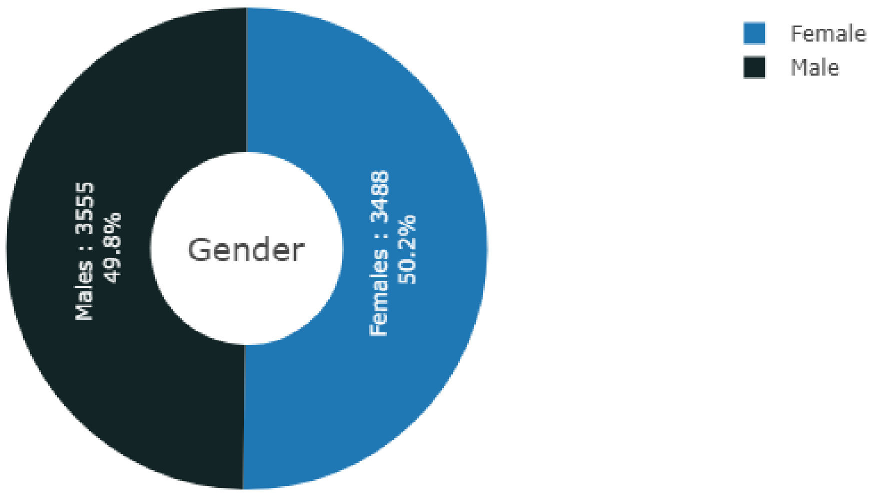

Male to Female Ratio in respect to Churned Customers
Male
Female
Churned Customers
🟥 Male
🟩 Female


Male to Female Ratio in respect to Non-Churned Customers
Male
Female
Non-Churned Customers
🟥 Male
🟩 Female

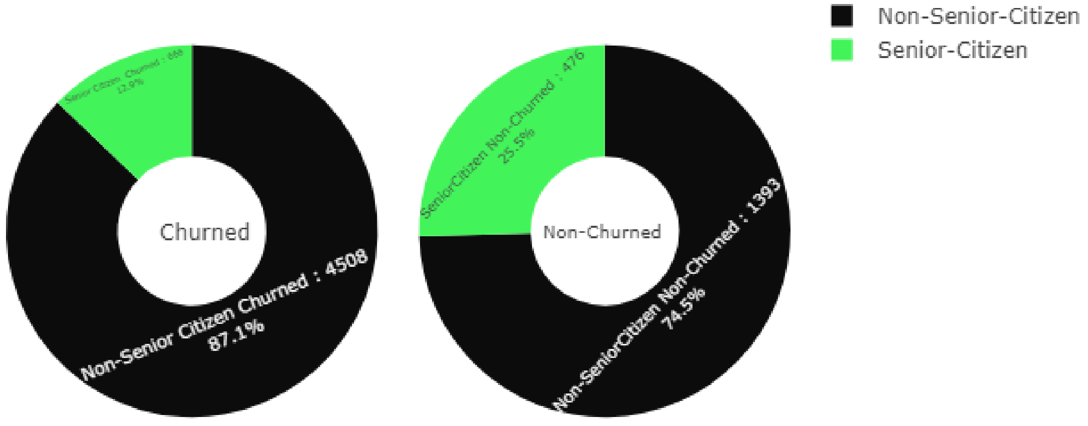

Gender Distribution Bar Chart


Gender Distrubution in respect to Churned & Non-Churned

Pie Plot of Gender ratio With Labeling

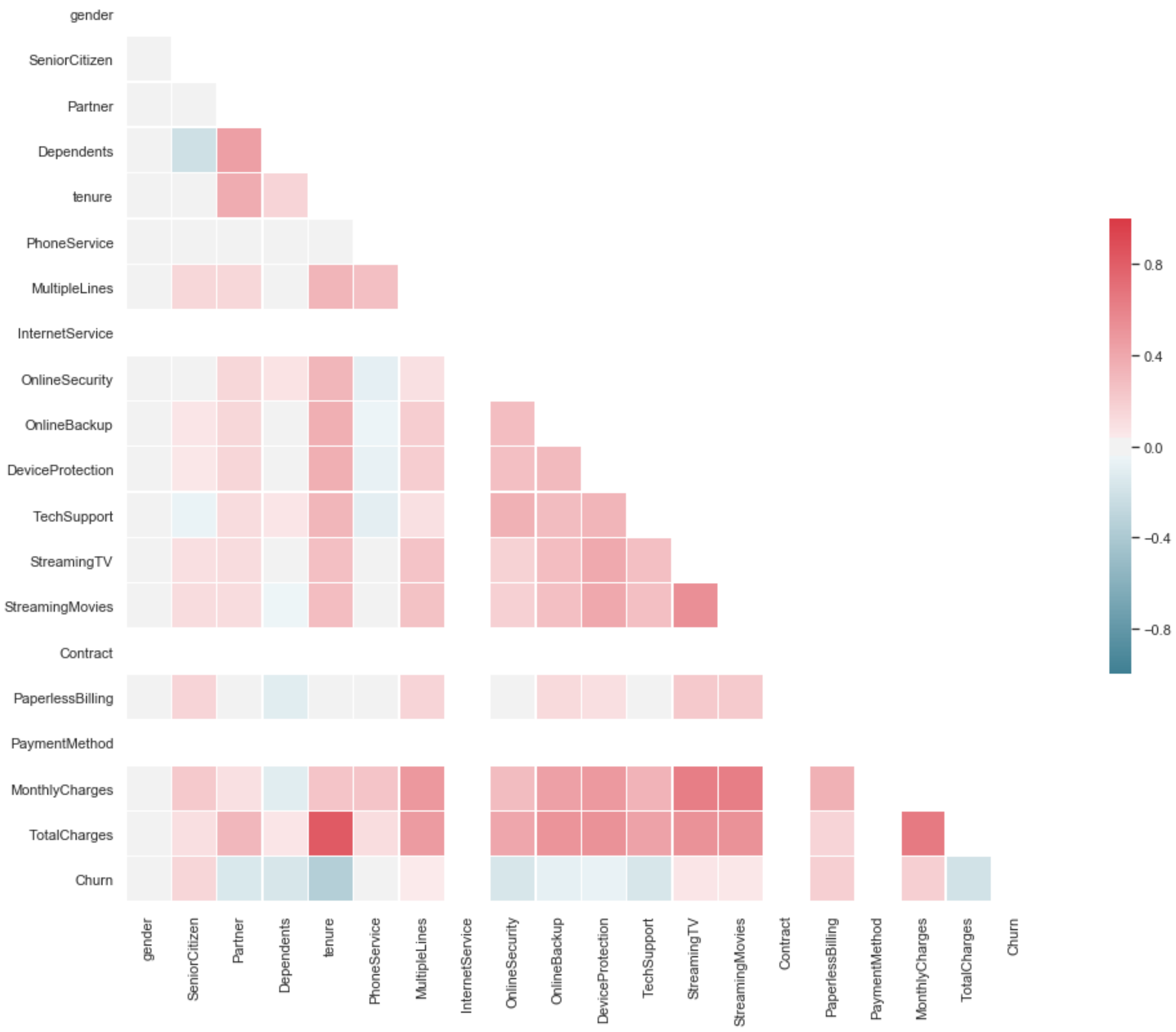


Senior Citizen Distrubution in respect to Churned & Non-Churned



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

- 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 InternetService.
- Almost there's no correlation between our target "Churn" and Contract.
- Almost there's no correlation between our target "Churn" and PaymentMethod.
- There's no correlation between gender and the other features.

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

Algorithms and Techniques

SVM:

A Support Vector Machine (SVM) is a type of classifier that goes by separating hyperplane. Moreover, uses labeled training which in our definition (*supervised learning*), the algorithm outputs an optimal hyperplane which categorizes new labels that is (Test data). In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

Tuning parameters: Kernel, Regularization, Gamma and Margin.

Kernel

The learning rate of a SVM model goes by transforming any problem into a linear mode solution where it goes as.

The Linear Solution goes by predicting a new input by applying the Cross do Product between X which is the input and a support vector which is Xi:

$f(x) = B(0) + sum(ai * (x,xi))$

The equation involves calculating inner product of a new input vector of X with all available support vectors. Whereas coefficients BO and ai for all possible inputs must get estimated by a learning algorithm from the training data.

The **polynomial kernel** can be written as $K(x,xi) = 1 + sum(x * xi)^d$ and **exponential** as $K(x,xi) = exp(-gamma * sum((x—xi^2))$.

*Polynomial and exponential kernels calculates separation line in higher dimension. This is called **kernel trick***

Logistic Regression:

Logistic regression is quite simple to understand, as it has historical applications in biological theorems where it has then evolved from to be applied to today’s continuous problems. In overall it’s a Special case of Linear Regression, where it’s variable is categorial dependent. In other words too it divides it’s data into binary classes where it’s outcome come in two results either yes or No. It’s dependent variable uses the Bernoulli Distribution method, moreover It’s Estimation goes by the most likely result,

For example,

- To predict whether an email is spam (1) or (0)
- Whether the tumor is malignant (1) or not (0)

Linear Regression Equation:

$$y = \beta 0 + \beta 1X1 + \beta 2X2 + \ldots\ldots + \beta nXn$$

Sigmoid Function:

$$p = 1/1 + e^{-y}$$

Application of Sigmoid function on linear regression:

$$p = 1/1 + e^{-(\beta 0 + \beta 1X1 + \beta 2X2 \ldots\ldots \beta nXn)}$$

In here we'll be using this Algorithm to divide our cases into four which are the TN, FP, FN, TP to cover our Churn And non-Churn cases and see what turns out in the end.

Resampling Data Techniques:

Resampling techniques are a set of methods that would be used to repeat a sampling process from
a certain sample or population slice, or define a precision of a state of a model from.

The mathematics behind it could be complex in contrast but comes easier more in digging deeper into.
And check more about resampling techniques here.

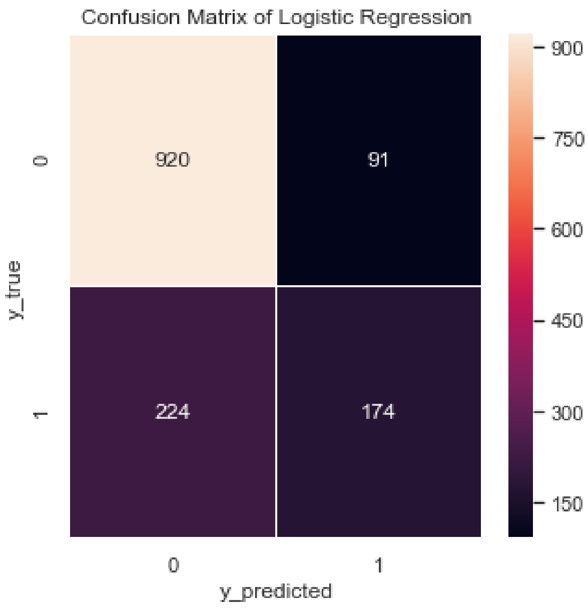
Recall and precision:

Recall and precision: Precision-Recall is a useful measure of success of prediction when the classes are imbalanced whereas, In information retrieval, precision is a measure of result relevancy that helps define correlation proportions , while recall is a measure of how many truly relevant results are returned moreover a model with high recall but low precision returns many results, which most of its predicted labels are incorrect results when compared to the training labels. In more contrast way a model with high precision but low recall is just the opposite, returning very few results of labels, but most of its predicted labels are correct of the comparison of the training labels. A mostly perfect model where with high Precision & Recall would return as much results as possible with mostly all of them are correct compared to the Labels.

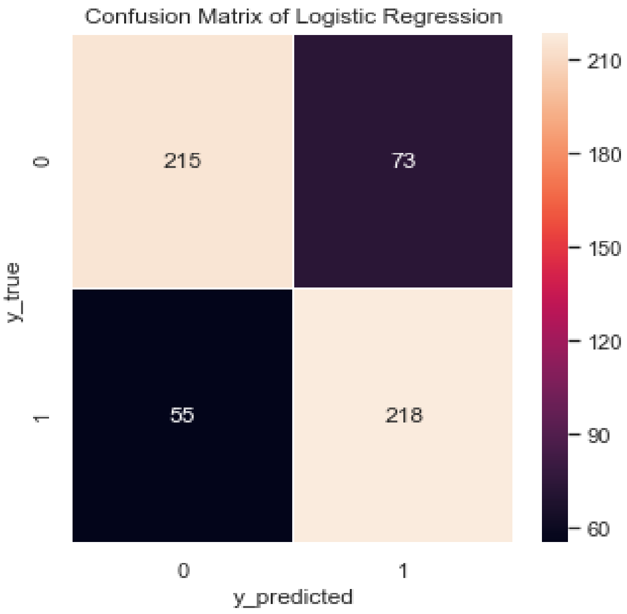
Benchmark :

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

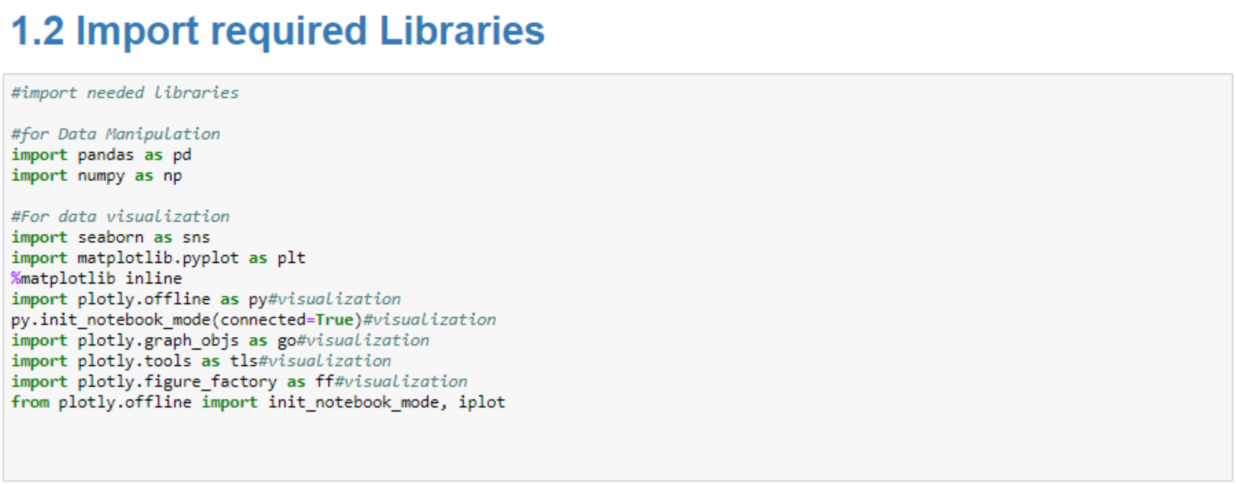


After Sampling



III. Methodology

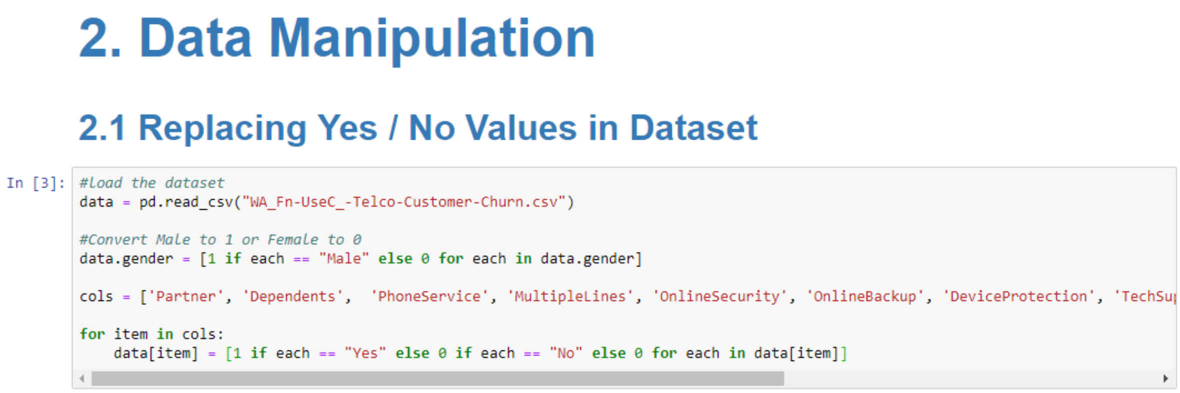
Importing Libraries



1.2 Import required Libraries

```
#import needed libraries

#for Data Manipulation
import pandas as pd
import numpy as np

#For data visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import plotly.offline as py#visualization
py.init_notebook_mode(connected=True)#visualization
import plotly.graph_objs as go#visualization
import plotly.tools as tls#visualization
import plotly.figure_factory as ff#visualization
from plotly.offline import init_notebook_mode, iplot
```

Data Manipulation



2. Data Manipulation

2.1 Replacing Yes / No Values in Dataset

In [3]:
```
#load the dataset
data = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")

#Convert Male to 1 or Female to 0
data.gender = [1 if each == "Male" else 0 for each in data.gender]

cols = ['Partner', 'Dependents',  'PhoneService', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSup

for item in cols:
    data[item] = [1 if each == "Yes" else 0 if each == "No" else 0 for each in data[item]]
```

2.2 Display Data After Manipulation ¶

```
data.head()
```

	Partner	Dependents	tenure	MultipleLines	OnlineSecurity	TechSupport	StreamingTV	StreamingMovies	PaperlessBilling	MonthlyCharges	Churn
0	1	0	1	0	0	0	0	0	1	0.251368	0
1	0	0	34	0	1	0	0	0	0	0.479579	0
2	0	0	2	0	1	0	0	0	1	0.453474	1
3	0	0	45	0	1	1	0	0	0	0.356211	0
4	0	0	2	0	0	0	0	0	1	0.595368	1

3. Analysis

3.1 Data Exploration

3.1.1 Load and Display DataSet

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

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSup
0	7590-VHVEG	0	0	1	0	1	0	0	DSL	0	...	0	
1	5575-GNVDE	1	0	0	0	34	1	0	DSL	1	...	1	
2	3668-QPYBK	1	0	0	0	2	1	0	DSL	1	...	0	
3	7795-CFOCW	1	0	0	0	45	0	0	DSL	1	...	1	
4	9237-HQITU	0	0	0	0	2	1	0	Fiber optic	0	...	0	

5 rows × 21 columns

3.1.2 Checking for Missing Data

```
#Check if there's a missing data at each column
data.isnull().sum().max()
```

```
0
```

3.1.3 Data Format Description

```
#check data formating
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
customerID          7043 non-null object
gender              7043 non-null int64
SeniorCitizen       7043 non-null int64
Partner             7043 non-null int64
Dependents          7043 non-null int64
tenure              7043 non-null int64
PhoneService        7043 non-null int64
MultipleLines       7043 non-null int64
InternetService     7043 non-null object
OnlineSecurity      7043 non-null int64
OnlineBackup        7043 non-null int64
DeviceProtection    7043 non-null int64
TechSupport         7043 non-null int64
StreamingTV         7043 non-null int64
StreamingMovies     7043 non-null int64
Contract            7043 non-null object
PaperlessBilling    7043 non-null int64
PaymentMethod       7043 non-null object
MonthlyCharges      7043 non-null float64
TotalCharges        7043 non-null object
Churn               7043 non-null int64
dtypes: float64(1), int64(15), object(5)
memory usage: 1.1+ MB
```

3.1.4 Data Descriptive Analysis

```
#Descriptive analysis
data.describe()
```

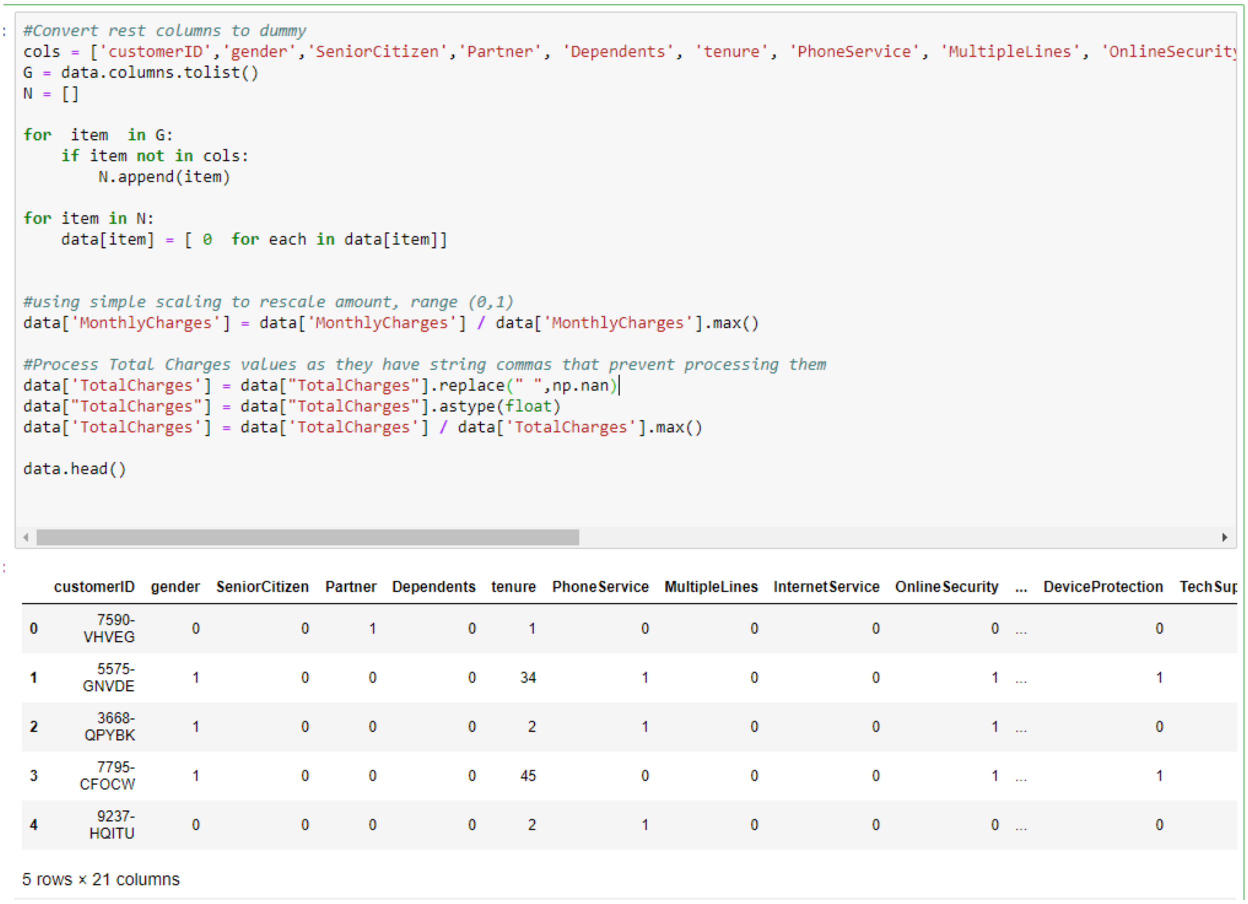
	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechS
count	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.
mean	0.504756	0.162147	0.483033	0.299588	32.371149	0.903166	0.421837	0.286668	0.344881	0.343888	0.
std	0.500013	0.368612	0.499748	0.458110	24.559481	0.295752	0.493888	0.452237	0.475363	0.475038	0.
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
25%	0.000000	0.000000	0.000000	0.000000	9.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.
50%	1.000000	0.000000	0.000000	0.000000	29.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.
75%	1.000000	0.000000	1.000000	1.000000	55.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.
max	1.000000	1.000000	1.000000	1.000000	72.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.

Data Preprocessing

Data Normalization

4.1 Data Preprocessing

4.1.1 Data Normalization

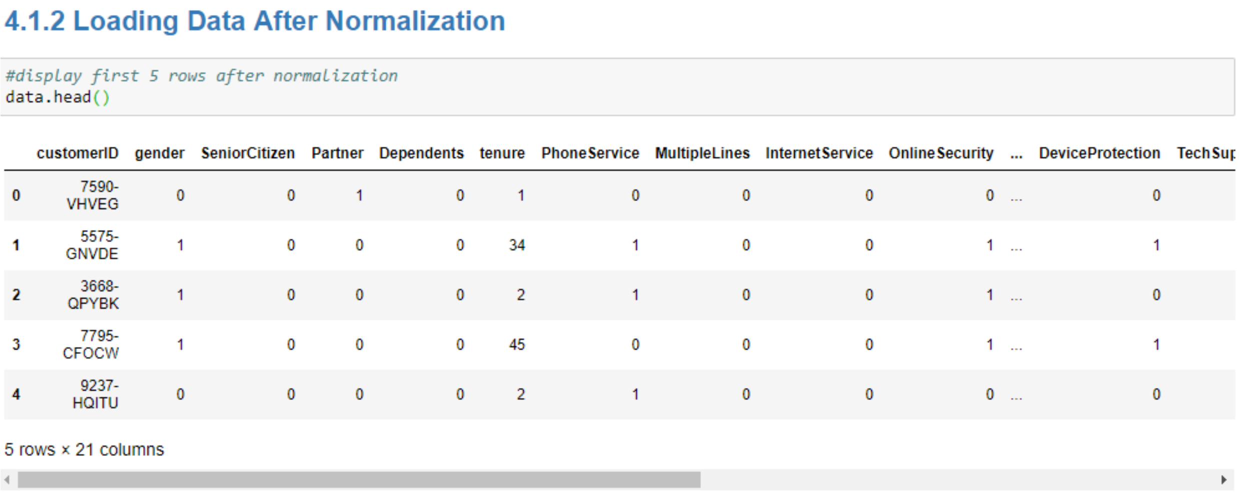


```
#Convert rest columns to dummy
cols = ['customerID','gender','SeniorCitizen','Partner', 'Dependents', 'tenure', 'PhoneService', 'MultipleLines', 'OnlineSecurit
G = data.columns.tolist()
N = []

for  item  in G:
    if item not in cols:
        N.append(item)

for item in N:
    data[item] = [ 0  for each in data[item]]


#using simple scaling to rescale amount, range (0,1)
data['MonthlyCharges'] = data['MonthlyCharges'] / data['MonthlyCharges'].max()

#Process Total Charges values as they have string commas that prevent processing them
data['TotalCharges'] = data["TotalCharges"].replace(" ",np.nan)
data["TotalCharges"] = data["TotalCharges"].astype(float)
data['TotalCharges'] = data['TotalCharges'] / data['TotalCharges'].max()

data.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSup
0	7590-VHVEG	0	0	1	0	1	0	0	0	0	...	0	
1	5575-GNVDE	1	0	0	0	34	1	0	0	1	...	1	
2	3668-QPYBK	1	0	0	0	2	1	0	0	1	...	0	
3	7795-CFOCW	1	0	0	0	45	0	0	0	1	...	1	
4	9237-HQITU	0	0	0	0	2	1	0	0	0	...	0	

5 rows × 21 columns

Loading Data After Normalization



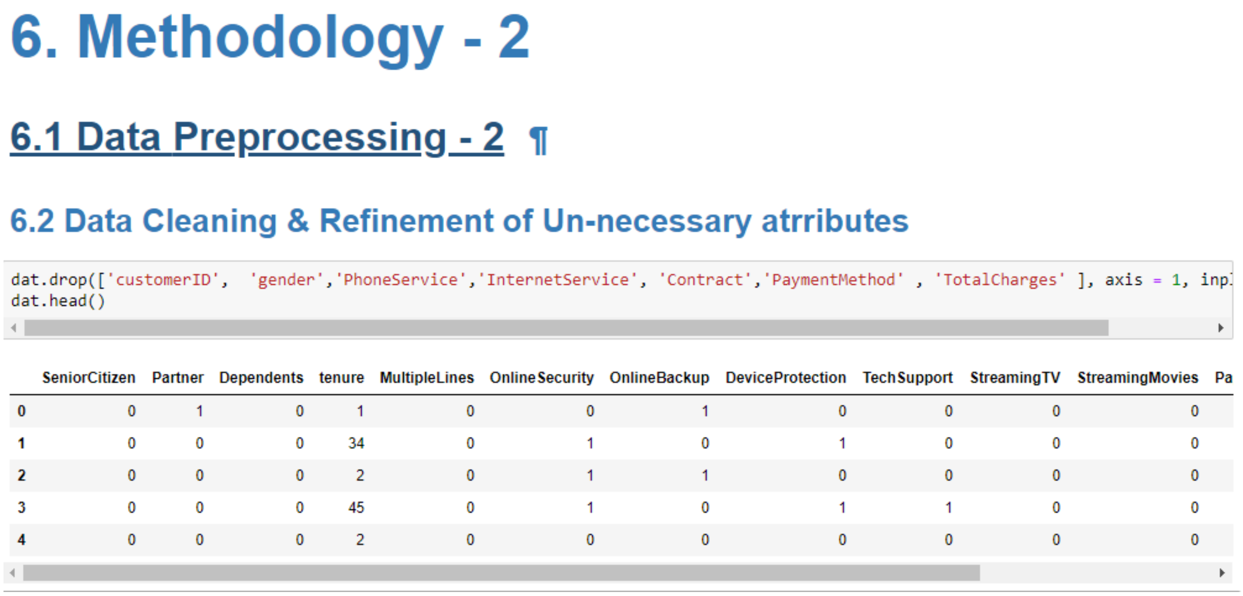
4.1.2 Loading Data After Normalization

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

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSup
0	7590-VHVEG	0	0	1	0	1	0	0	0	0	...	0	
1	5575-GNVDE	1	0	0	0	34	1	0	0	1	...	1	
2	3668-QPYBK	1	0	0	0	2	1	0	0	1	...	0	
3	7795-CFOCW	1	0	0	0	45	0	0	0	1	...	1	
4	9237-HQITU	0	0	0	0	2	1	0	0	0	...	0	

5 rows × 21 columns

Data Cleaning & Refinement of Un-necessary attributes



6. Methodology - 2

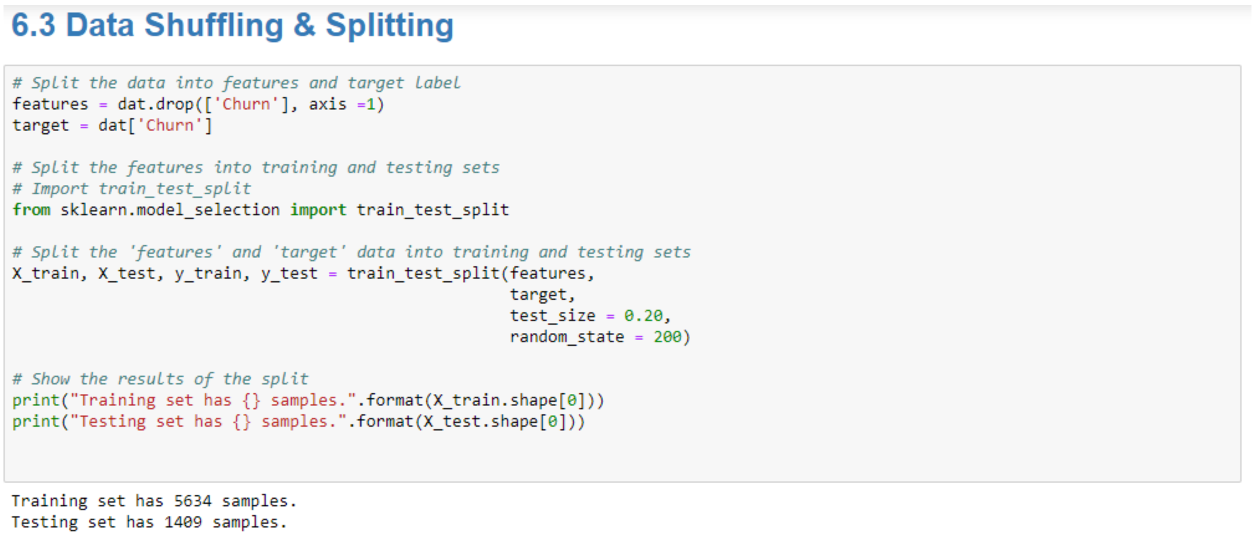
6.1 Data Preprocessing - 2 ¶

6.2 Data Cleaning & Refinement of Un-necessary atrributes

```
dat.drop(['customerID',  'gender','PhoneService','InternetService', 'Contract','PaymentMethod' , 'TotalCharges' ], axis = 1, inp
dat.head()
```

	SeniorCitizen	Partner	Dependents	tenure	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Pa
0	0	1	0	1	0	0	1	0	0	0	0	
1	0	0	0	34	0	1	0	1	0	0	0	
2	0	0	0	2	0	1	1	0	0	0	0	
3	0	0	0	45	0	1	0	1	1	0	0	
4	0	0	0	2	0	0	0	0	0	0	0	

Data Shuffling & Splitting

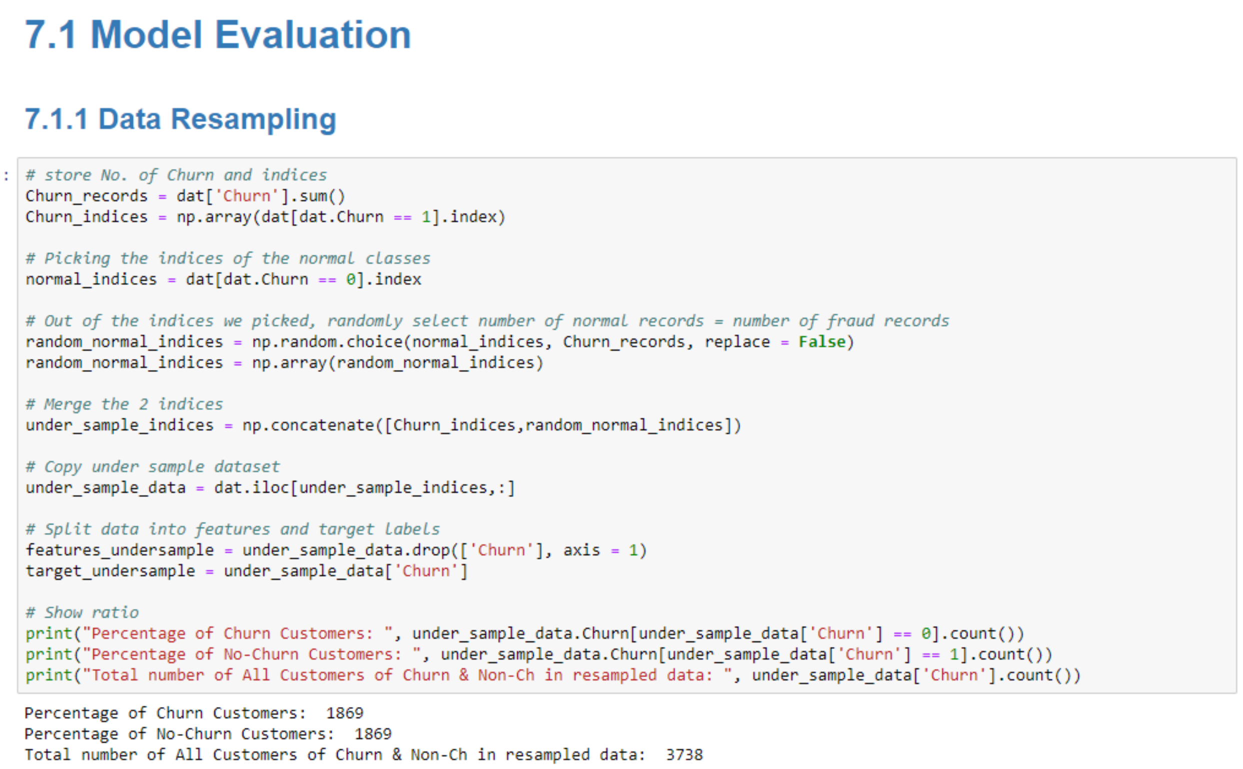


6.3 Data Shuffling & Splitting

```
# 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 5634 samples.
Testing set has 1409 samples.
```

Data Resampling



7.1 Model Evaluation

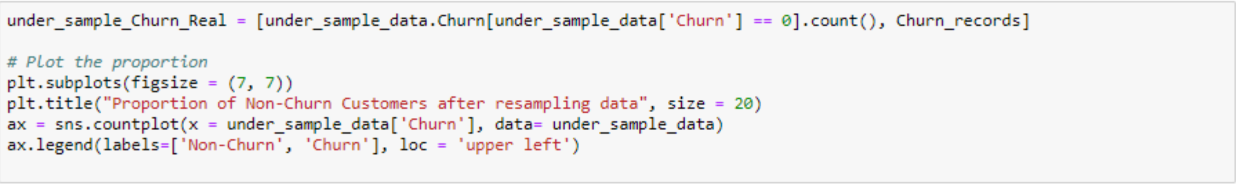
7.1.1 Data Resampling

```
# 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
```

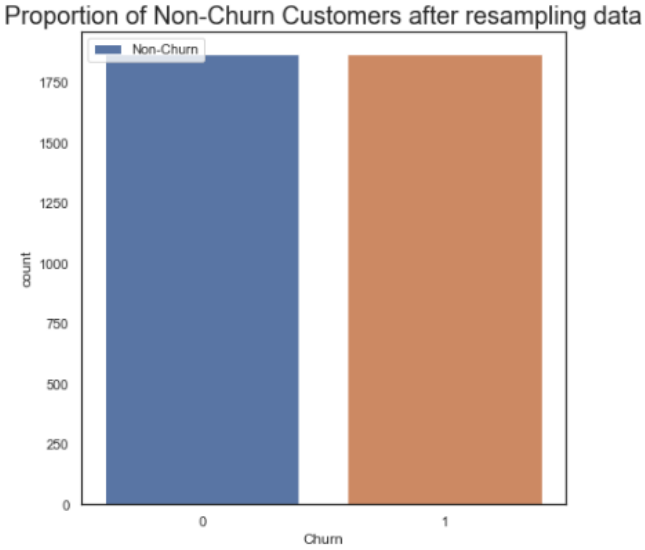
Data Plotting after Resampling

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')
```

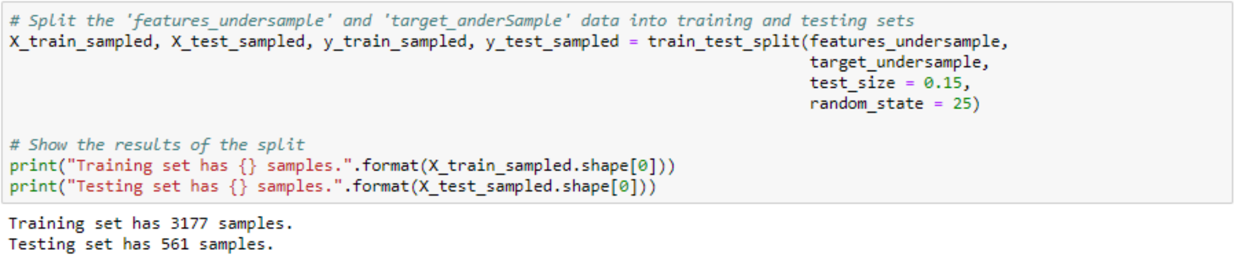
```
<matplotlib.legend.Legend at 0x27c930c0630>
```



Proportion of Non-Churn Customers after resampling data

Shuffle and Split Data after Resampling

7.1.3 Shuffle and Split Data after Resampling



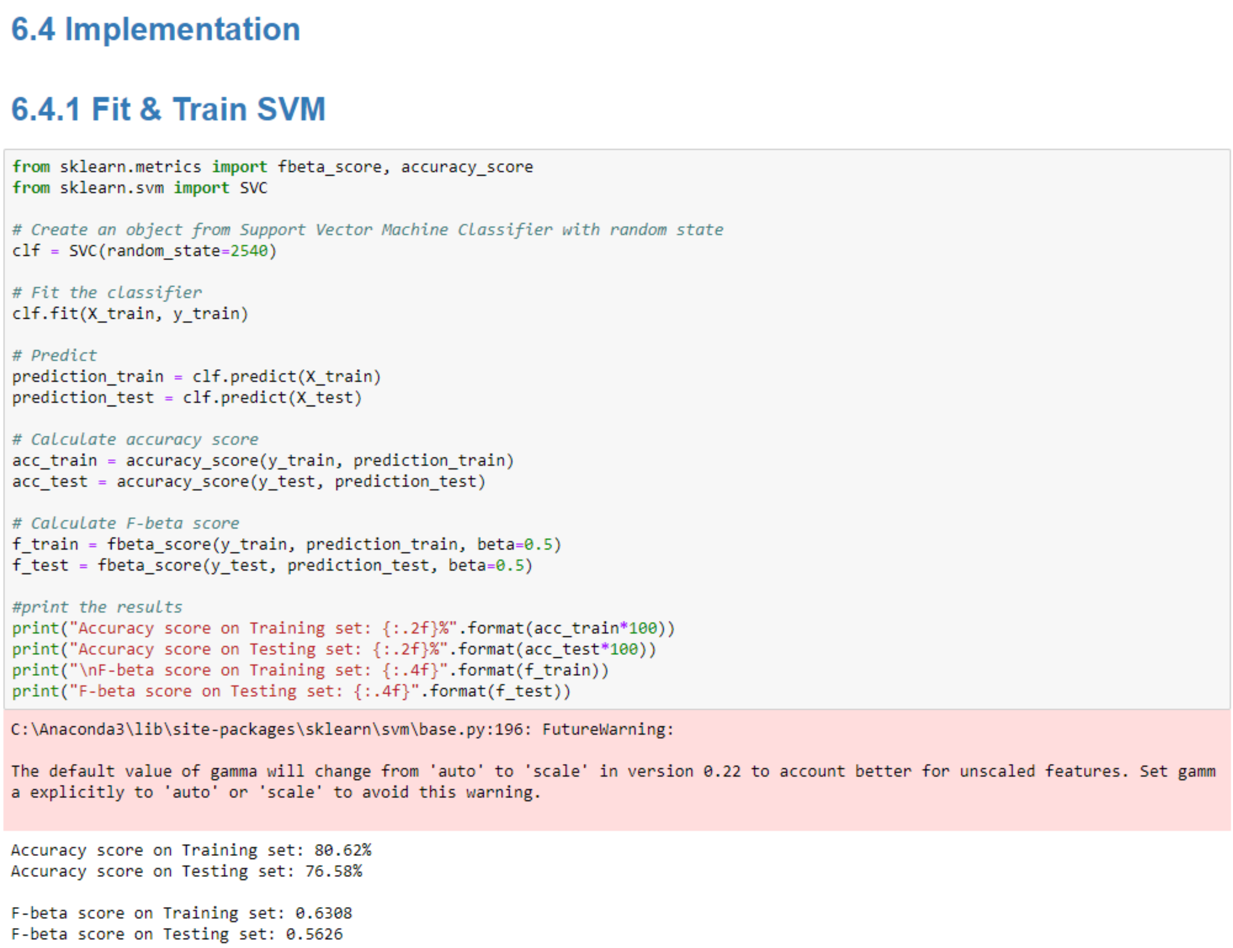
```
# Split the 'features_undersample' and 'target_anderSample' data into training and testing sets
X_train_sampled, X_test_sampled, y_train_sampled, y_test_sampled = train_test_split(features_undersample,
                                                                                    target_undersample,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

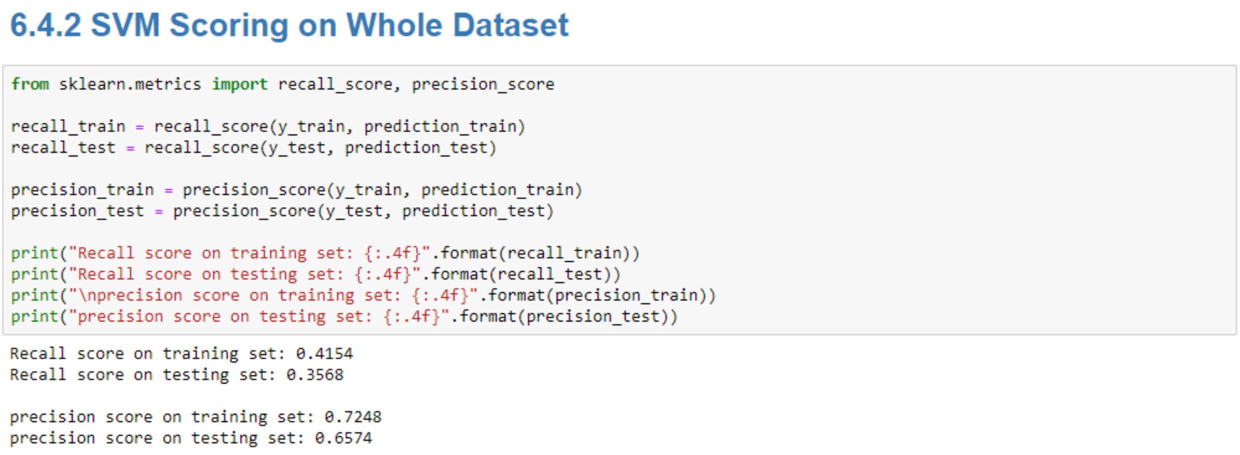
```python
from sklearn.metrics import fbeta_score, accuracy_score
from sklearn.svm import SVC

# 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: 80.62%
Accuracy score on Testing set: 76.58%

F-beta score on Training set: 0.6308
F-beta score on Testing set: 0.5626
```

1-SVM Scoring on Whole Dataset

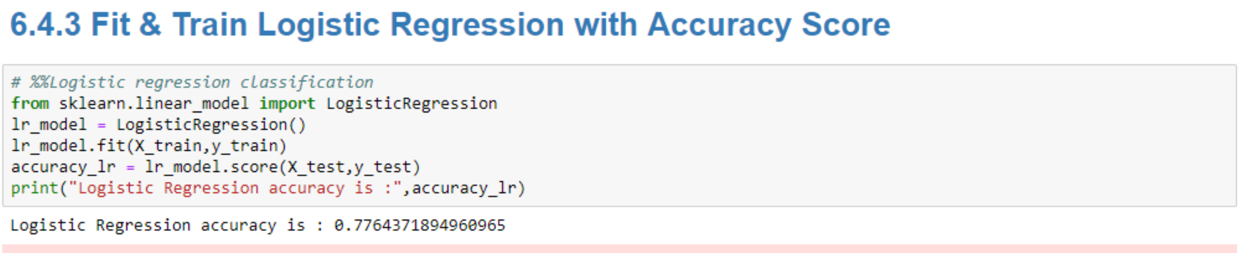
2-Fit & Train Logistic Regression with Accuracy Score



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: 0.4154
Recall score on testing set: 0.3568

precision score on training set: 0.7248
precision score on testing set: 0.6574
```

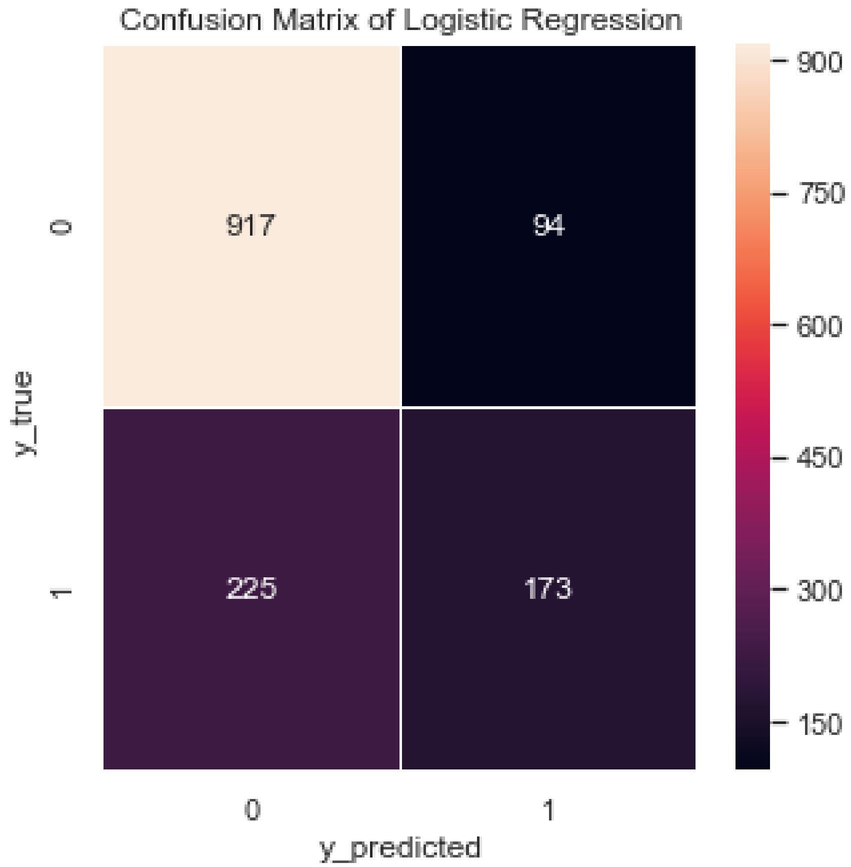


6.4.3 Fit & Train Logistic Regression with Accuracy Score

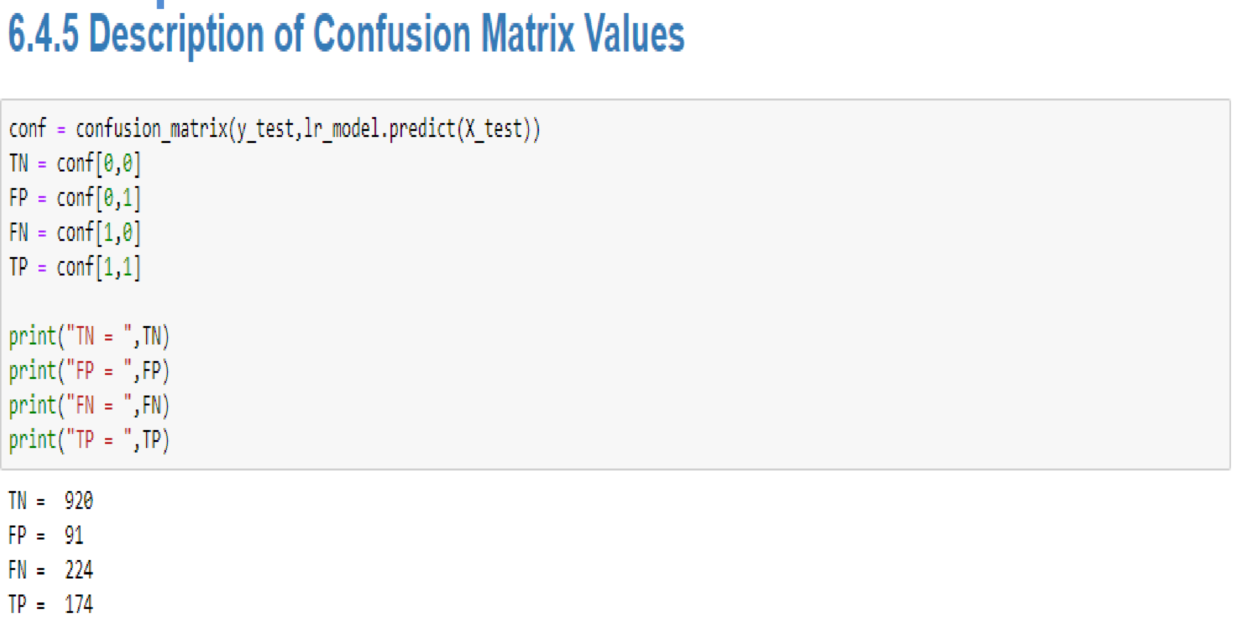
```
# %%Logistic regression classification
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression()
lr_model.fit(X_train,y_train)
accuracy_lr = lr_model.score(X_test,y_test)
print("Logistic Regression accuracy is :",accuracy_lr)
```

```
Logistic Regression accuracy is : 0.7764371894960965
```

Calculate Confusion Matrix for Log Regression



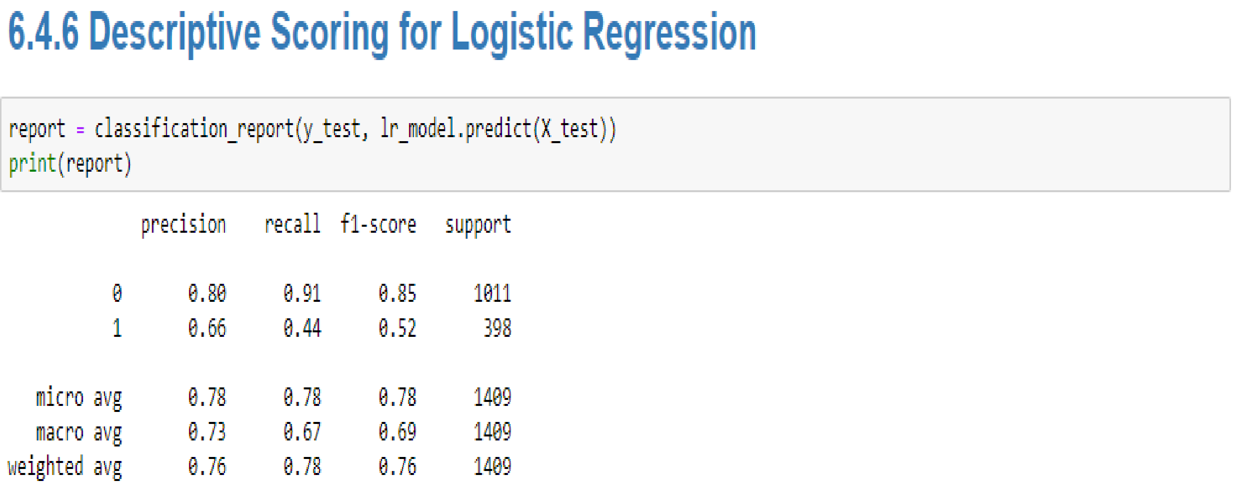
Description of Confusion Matrix Values



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 =  920
FP =  91
FN =  224
TP =  174
```



6.4.6 Descriptive Scoring for Logistic Regression

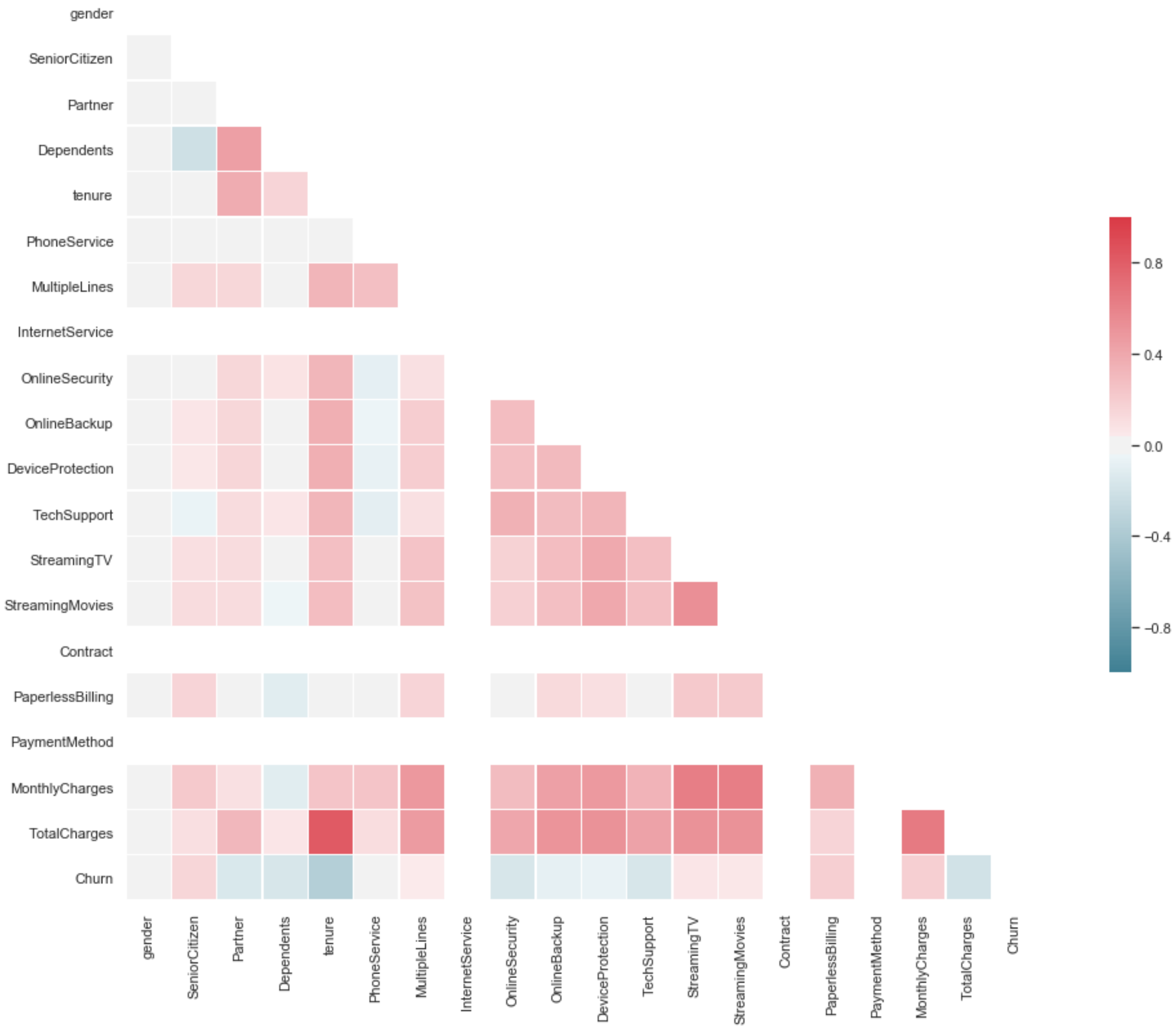
```
report = classification_report(y_test, lr_model.predict(X_test))
print(report)
```

```
              precision    recall  f1-score   support

           0       0.80      0.91      0.85      1011
           1       0.66      0.44      0.52       398

   micro avg       0.78      0.78      0.78      1409
   macro avg       0.73      0.67      0.69      1409
weighted avg       0.76      0.78      0.76      1409
```

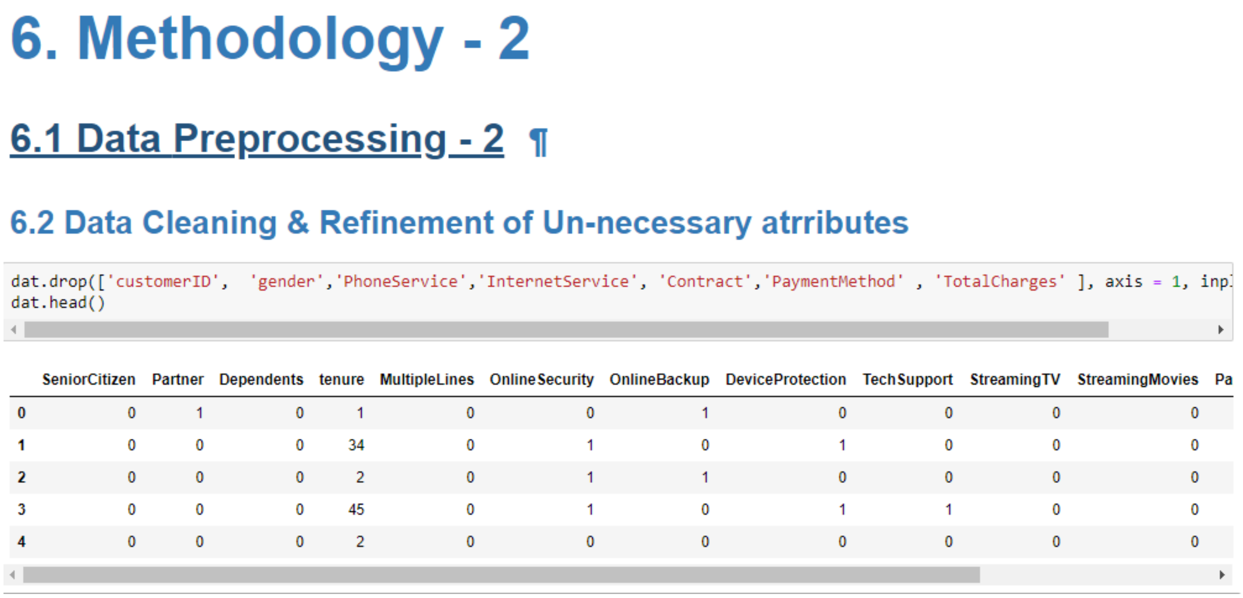
Refinement

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.

We use first the correlation matrix to find the less relevant features that we'll exclude later



Then after finding that features like gender, internet Service,Contract, Payment method are less relevant to our target churn class we exclude them from our data



6. Methodology - 2

6.1 Data Preprocessing - 2 ¶

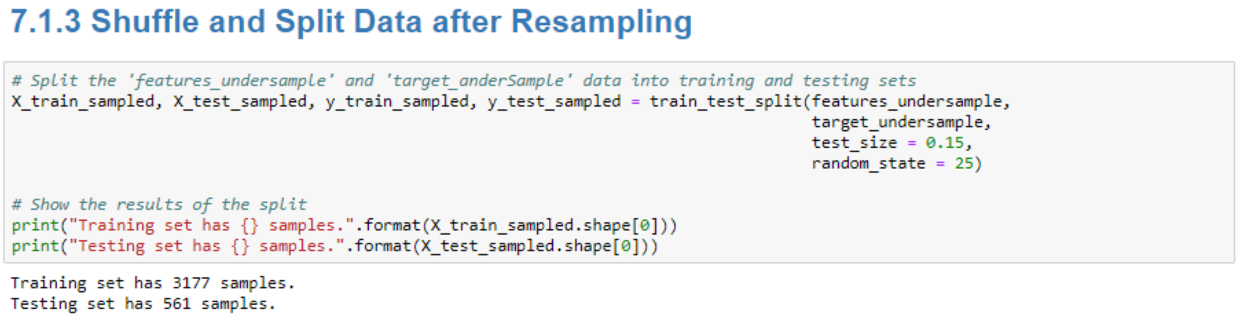
6.2 Data Cleaning & Refinement of Un-necessary atrributes

```
dat.drop(['customerID',  'gender','PhoneService','InternetService', 'Contract','PaymentMethod' , 'TotalCharges' ], axis = 1, inpl
dat.head()
```

	SeniorCitizen	Partner	Dependents	tenure	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Pa
0	0	1	0	1	0	0	1	0	0	0	0	
1	0	0	0	34	0	1	0	1	0	0	0	
2	0	0	0	2	0	1	1	0	0	0	0	
3	0	0	0	45	0	1	0	1	1	0	0	
4	0	0	0	2	0	0	0	0	0	0	0	

Which leaves us later ready for Shuffling and splitting data.

As well we can tune test size and random state for shuffling and splitting which can help us generate new states of the data thus improving future qualities of the scores.



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]))
```

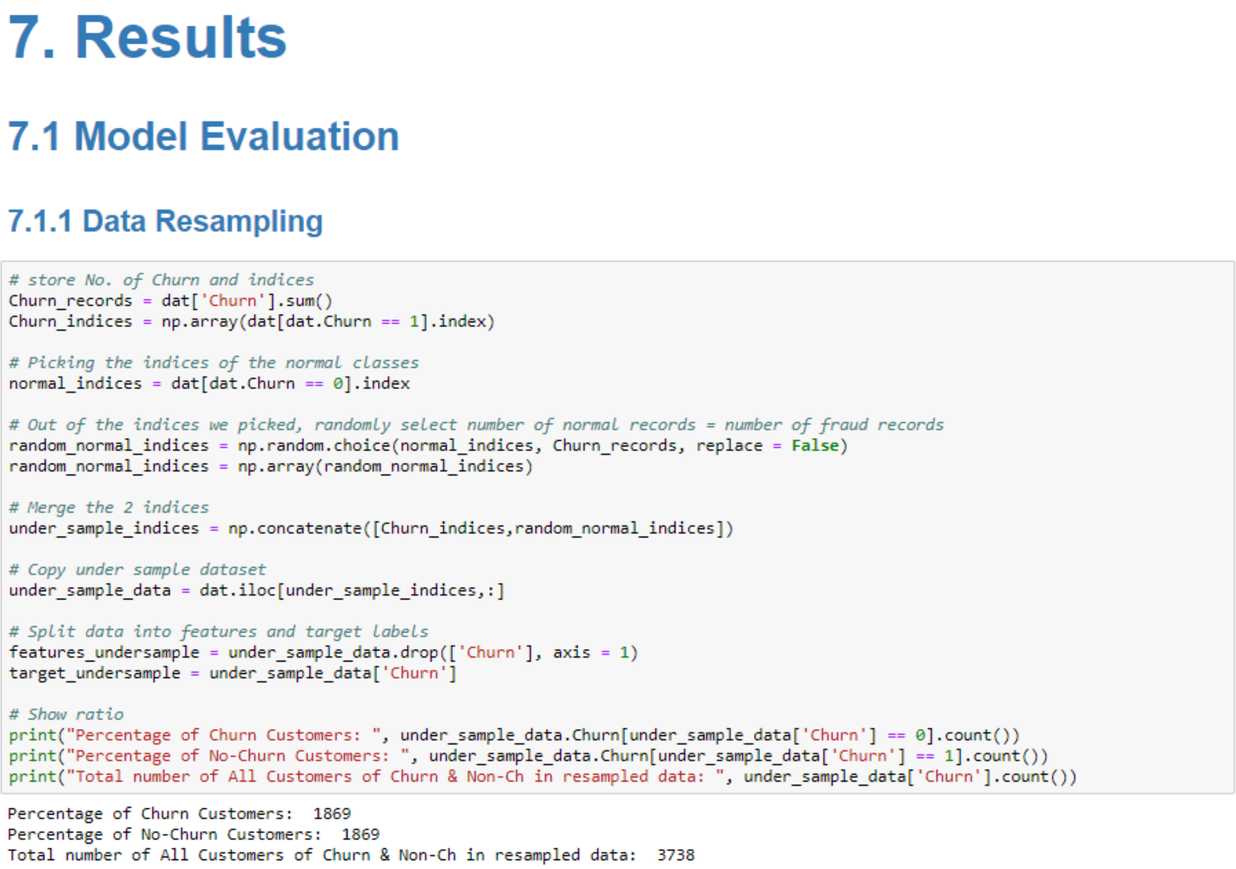
```
Training set has 3177 samples.
Testing set has 561 samples.
```

IV. Results

As we will see the model reacts better after the sampling deals well with unseen data as it targets only its main goal the churn as well , though it needs more data to be adjusted well for the model to start grasping more to the definition of the data and classifying it.

Model Evaluation and Validation

Data Resampling



7. Results

7.1 Model Evaluation

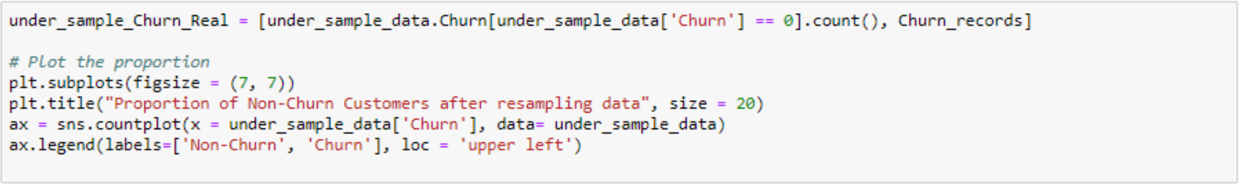
7.1.1 Data Resampling

```
# 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
```

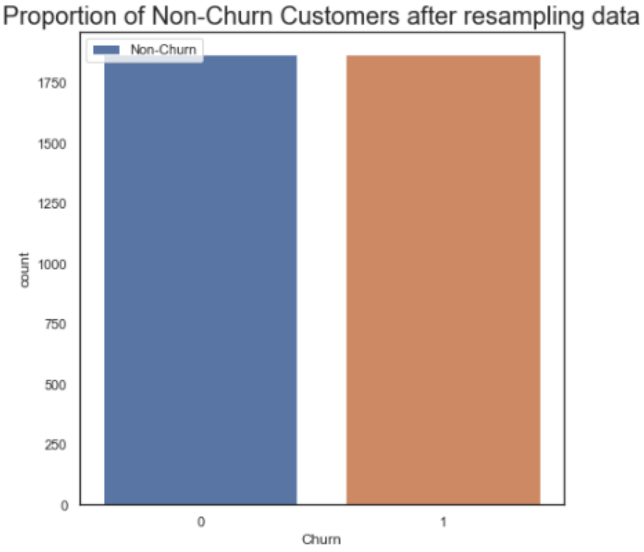
Data Plotting after Resampling

7.1.2 Data Plotting after Resampling



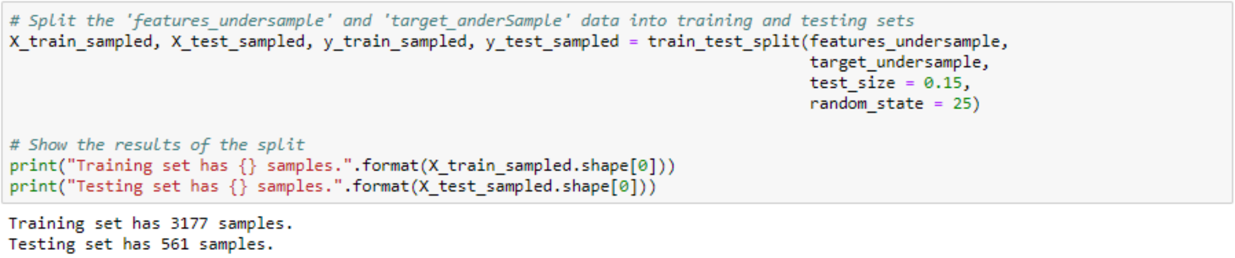
```
under_sample_Churn_Real = [under_sample_data.Churn[under_sample_data['Churn'] == 0].count(), Churn_records]

# 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

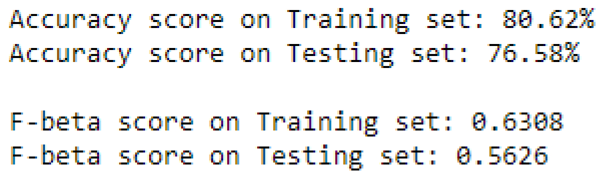
As we see there is a slight much improvement in the SVM Scores after Sampling & Shuffling

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.

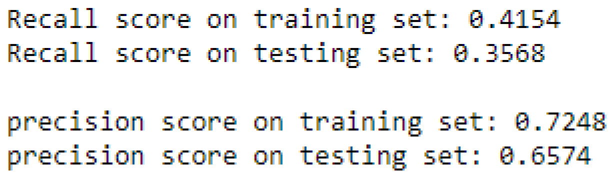
As for the Logistic Regression we can see that the sampling has helped improve a distant large amount in the following precision, F-1 Score, and recall Scores especially in the prediction section of the Churned Customers, as for the accuracy doesn't seem to be affected much, with overall the scores could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

Bench Mark Models

SVM:

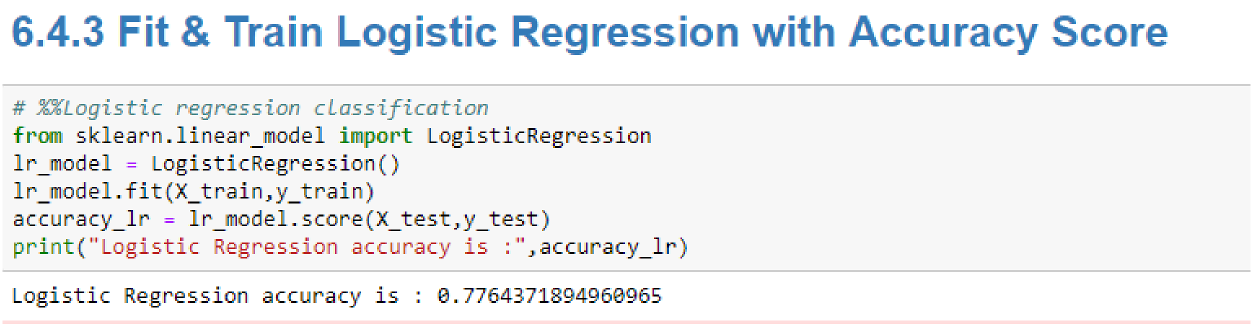


```
Accuracy score on Training set: 80.62%
Accuracy score on Testing set: 76.58%

F-beta score on Training set: 0.6308
F-beta score on Testing set: 0.5626
```



```
Recall score on training set: 0.4154
Recall score on testing set: 0.3568

precision score on training set: 0.7248
precision score on testing set: 0.6574
```

Logistic Regression:

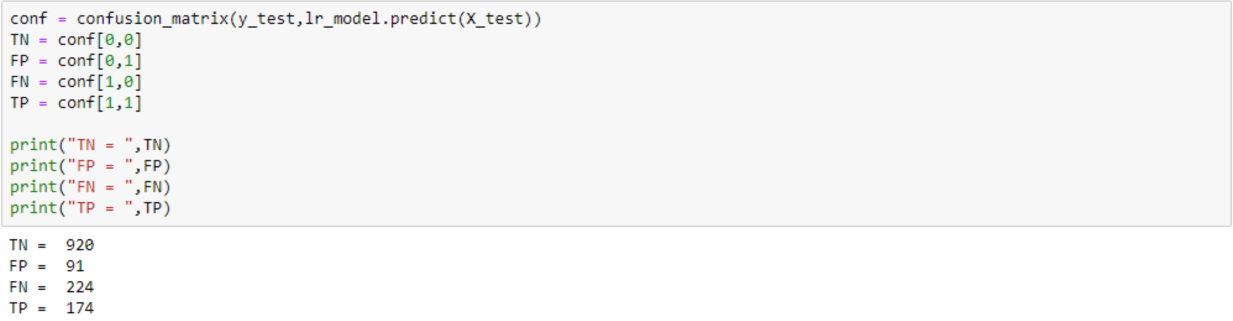


6.4.3 Fit & Train Logistic Regression with Accuracy Score

```
# %%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 : 0.7764371894960965
```

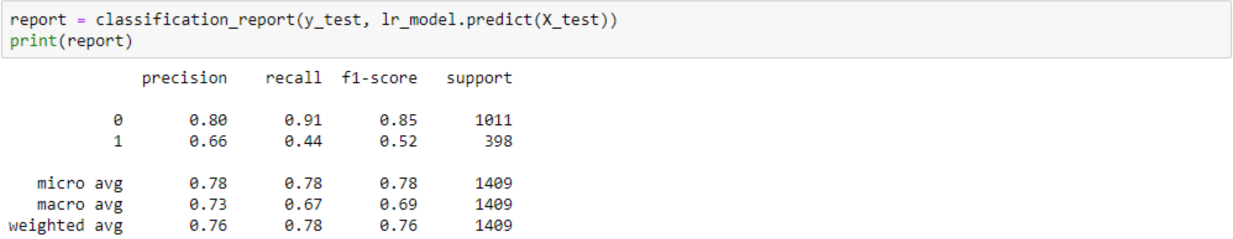
6.4.5 Description of Confusion Matrix Values



```
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 =  920
FP =  91
FN =  224
TP =  174
```

6.4.6 Descriptive Scoring for Logistic Regression

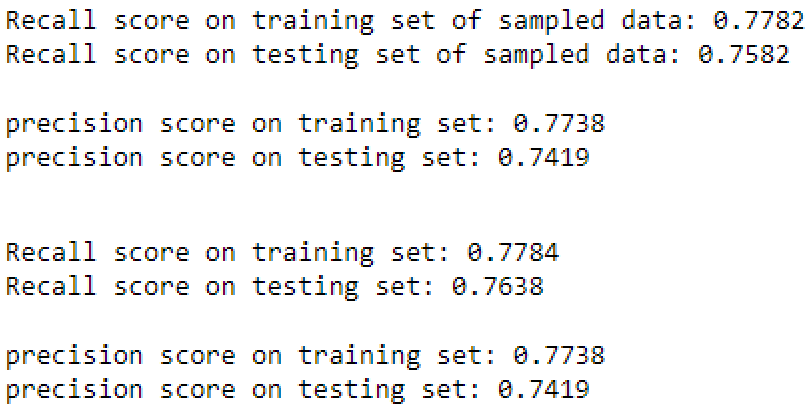


```
report = classification_report(y_test, lr_model.predict(X_test))
print(report)
```

```
              precision    recall  f1-score   support

           0       0.80      0.91      0.85      1011
           1       0.66      0.44      0.52       398

   micro avg       0.78      0.78      0.78      1409
   macro avg       0.73      0.67      0.69      1409
weighted avg       0.76      0.78      0.76      1409
```

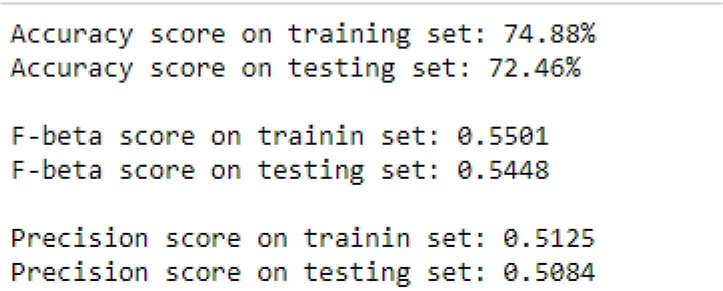
Final Models

SVM:



```
Recall score on training set of sampled data: 0.7782
Recall score on testing set of sampled data: 0.7582

precision score on training set: 0.7738
precision score on testing set: 0.7419


Recall score on training set: 0.7784
Recall score on testing set: 0.7638

precision score on training set: 0.7738
precision score on testing set: 0.7419
```

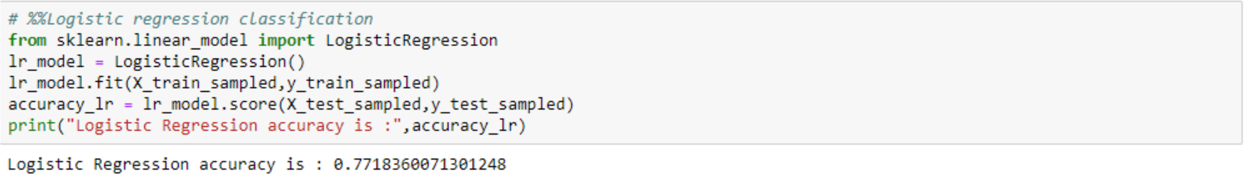
Scoring on Whole Dataset:



```
Accuracy score on training set: 74.88%
Accuracy score on testing set: 72.46%

F-beta score on trainin set: 0.5501
F-beta score on testing set: 0.5448

Precision score on trainin set: 0.5125
Precision score on testing set: 0.5084
```

Logistic Regression

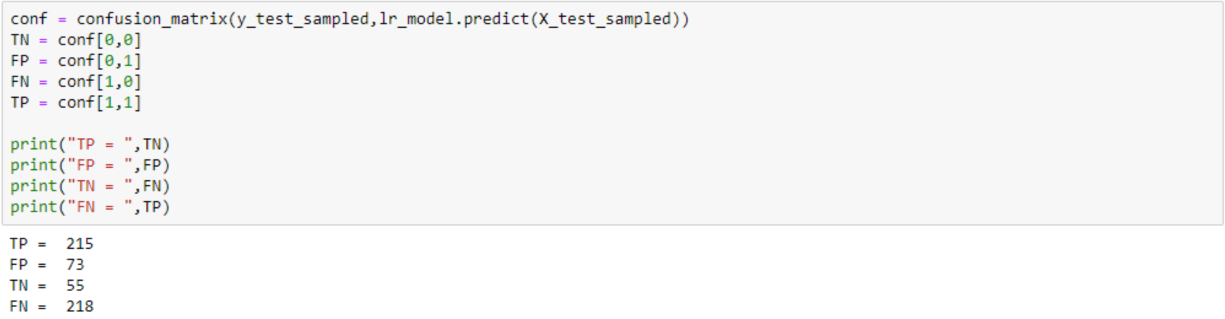
7.2.4 Fit & Train Log Reg after Resampling with Accuracy Score



```
# %%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 : 0.7718360071301248
```

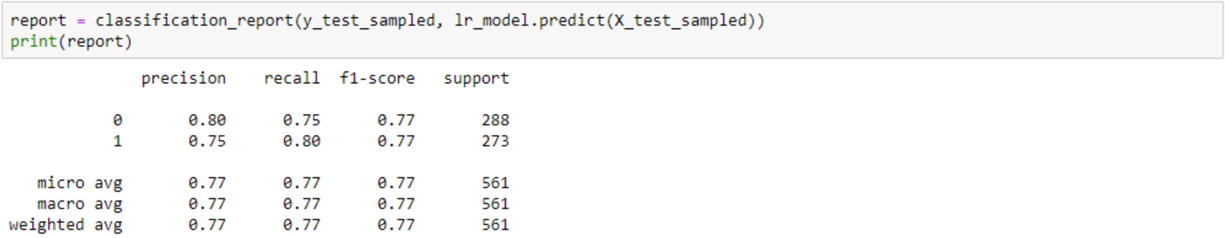
7.2.6 Description of Confusion Matrix Values



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

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

```
TP =  215
FP =  73
TN =  55
FN =  218
```

7.2.7 Descriptive Scoring for Logistic Regression



```
report = classification_report(y_test_sampled, lr_model.predict(X_test_sampled))
print(report)
```

```
              precision    recall  f1-score   support

           0       0.80      0.75      0.77       288
           1       0.75      0.80      0.77       273

   micro avg       0.77      0.77      0.77       561
   macro avg       0.77      0.77      0.77       561
weighted avg       0.77      0.77      0.77       561
```

V. Conclusion

Free-Form Visualization

All needed visualization is attached in above sections

Reflection

- **7. Results**
    - **7.1 Model Evaluation**
        - **7.1.1 Data Resampling**
        - **7.1.2 Data Plotting after Resampling**
        - **7.1.3 Shuffle and Split Data after Resampling**
    - **7.2 Models Justification & Comparison after Improvement**
        - **7.2.1 Fit & Train SVM after Resampling**
        - **7.2.2 SVM Scoring after Resampling on Whole Dataset**
        - **7.2.3 More SVM Scoring with F-Beta on Sampled Dataset**
        - **7.2.4 Fit & Train Log Reg after Resampling with Accuracy Score**
        - **7.2.5 Calculate Confusion Matrix for Log Regression**
        - **7.2.6 Description of Confusion Matrix Values**
        - **7.2.7 Descriptive Scoring for Logistic Regression**

    - **As in overall the challenging part of the whole process is dealing with the dataset and preprocessing it to suit the implementation part of the procedure as well the breaking down of the data and splitting it required quiet tuning to retain some results which had different results, as well it could achieve different various conclusions and scores which contains the most aggressive part of the project.**

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.

For K-fold we can go for the following :

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
1. Take the group as a hold out or test data set
2. Take the remaining groups as a training data set
3. Fit a model on the training set and evaluate it on the test set
4. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores

Configuration of k

The k value must be chosen carefully for the data set.

A unwisely chosen value for k may result in a mis-performing of the model over the data, moreover a score with a high variance (that can change a lot of outcomes based on the data used to fit the model which may result on a lot of possible scores), or a high bias, (where overestimation of the outcomes or overfitting in our words).

Three common tactics for choosing a value for k are as follows:

- **Representative**: The value for k is chosen such that each train/test group of data samples is large enough to be statistically representative of the broader dataset.
- **k=10**: The value for k is fixed to 10, a value that has been found through experimentation to generally result in a model skill estimate with low bias a modest variance.
- **k=n**: The value for k is fixed to n, where n is the size of the dataset to give each test sample an opportunity to be used in the hold out dataset. This approach is called leave-one-out cross-validation.
- **A good Article features the following can be found here.**

As for Grid Search we can go for following :

What is grid search?

Grid search is the process of performing hyper parameter tuning in order to determine the optimal values for a given model. This is significant as the performance of the entire model is based on the hyper parameter values specified.

In contrast, a ***parameter*** is an internal characteristic of the model and its value can be estimated from data. Example, beta coefficients of linear/logistic regression or support vectors in Support Vector Machines.

Grid-search is used to find the optimal hyperparameters *of a model which results in the most 'accurate' predictions.*

A good well documented reference can be found here on this topic.