

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.

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 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](#).

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

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

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

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Recall and Precision Metrics

- **Recall:** ability of a classification model to identify all relevant instances
- **Precision:** ability of a classification model to return only relevant instances
- **F1 score:** single metric that combines recall and precision using the harmonic mean

We tend to use metrics as our intuition tells us we should maximize is known in statistics as **recall**, or the ability of a model to find all the relevant cases within a dataset. The precise 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 data point classified as positive by the model that actually are positive (meaning they are correct), and false negatives are data points the model identifies as negative that actually are positive (incorrect).

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', 'InternetService',
 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
 '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 forming
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

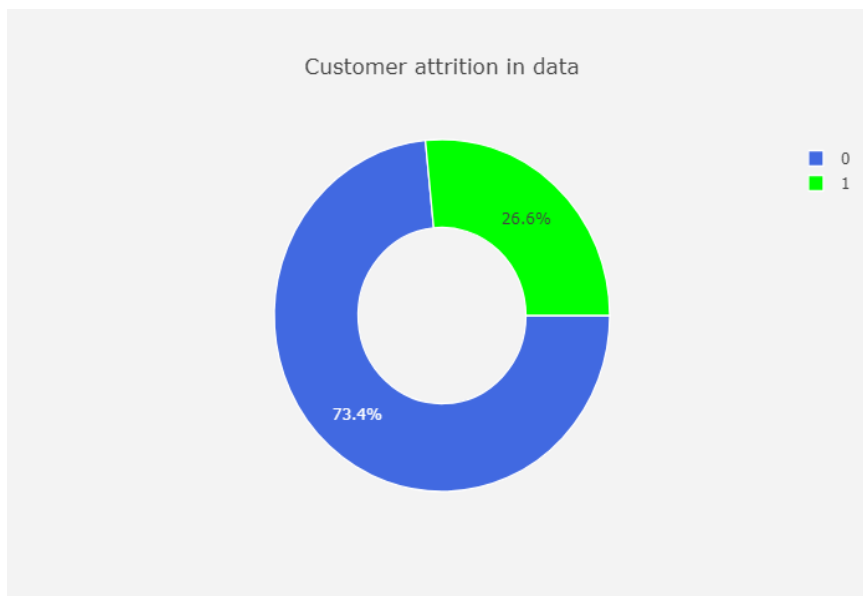
```
In [407]: #Descriptive analysis  
data.describe()
```

```
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

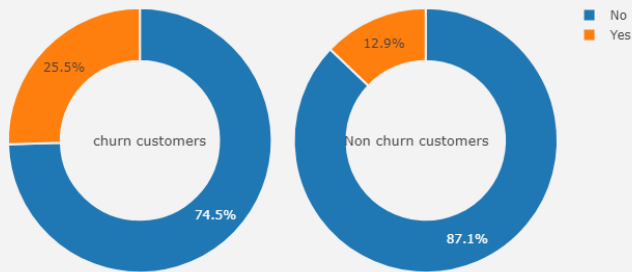
1.Churn to Non-Churn Proportion



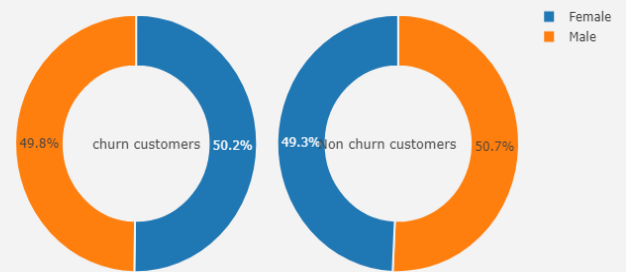
- The data is mostly unbalanced, Evaluate modeling on unbalanced data is a terrible mistake. Will see later why and how to work with unbalanced data.

2.Statistical Analysis in Customer Attrition

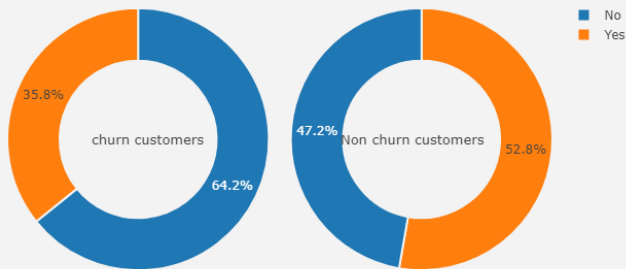
SeniorCitizen distribution in customer attrition



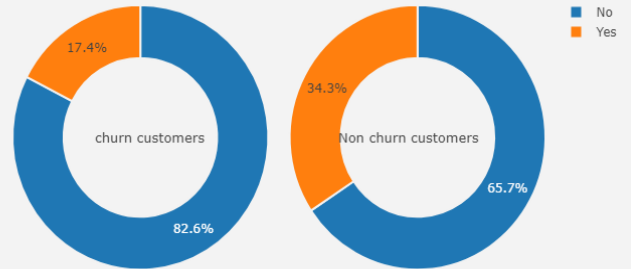
gender distribution in customer attrition



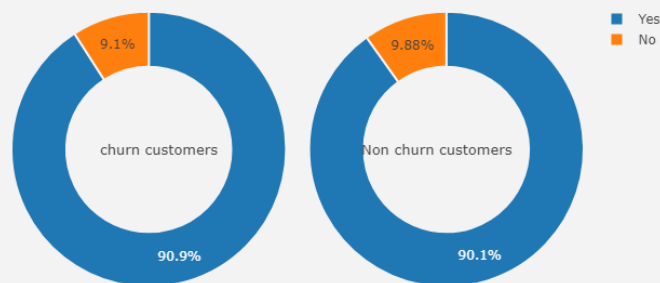
Partner distribution in customer attrition



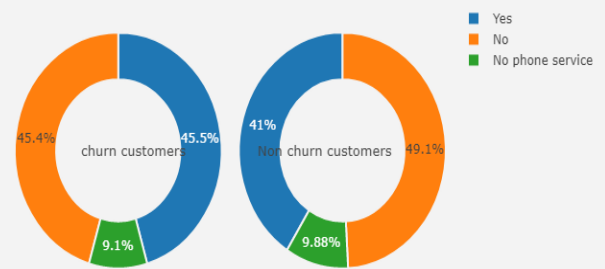
Dependents distribution in customer attrition



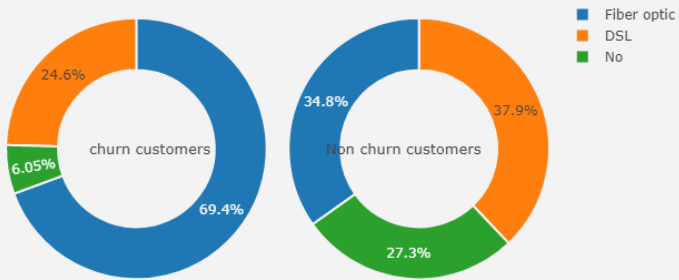
PhoneService distribution in customer attrition



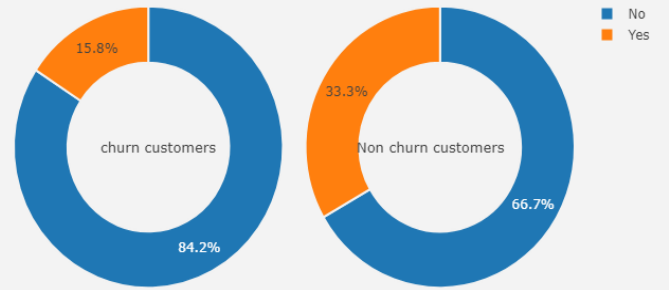
MultipleLines distribution in customer attrition



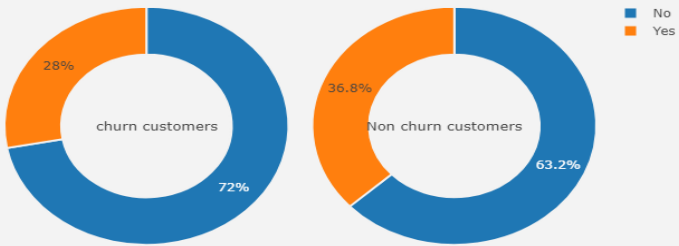
InternetService distribution in customer attrition



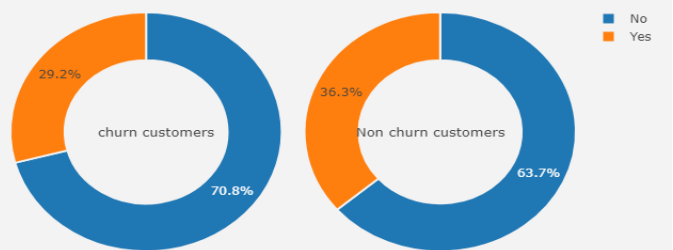
OnlineSecurity distribution in customer attrition



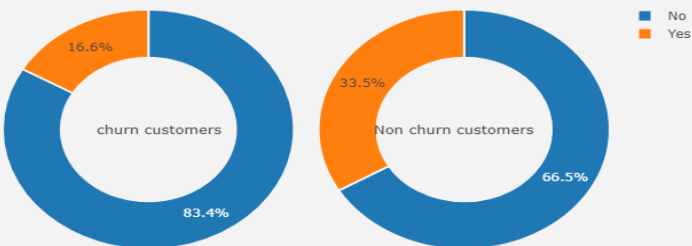
OnlineBackup distribution in customer attrition



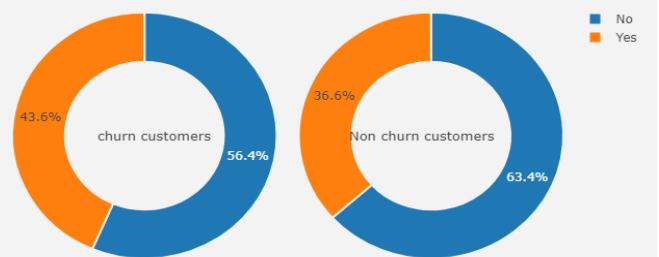
DeviceProtection distribution in customer attrition



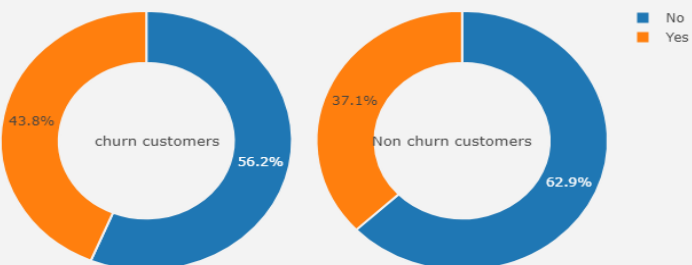
TechSupport distribution in customer attrition



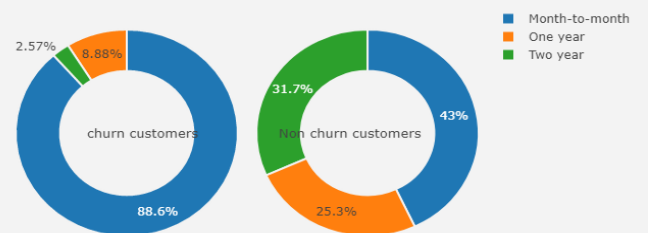
StreamingTV distribution in customer attrition



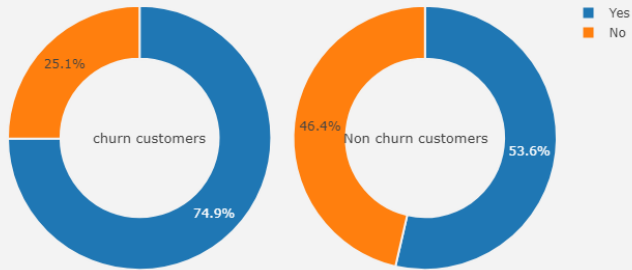
StreamingMovies distribution in customer attrition



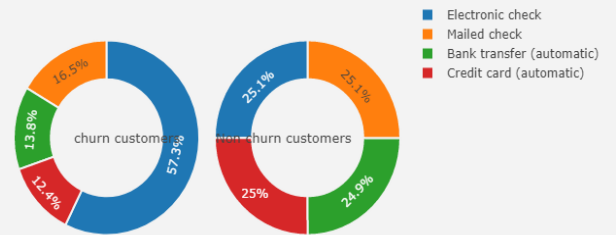
Contract distribution in customer attrition



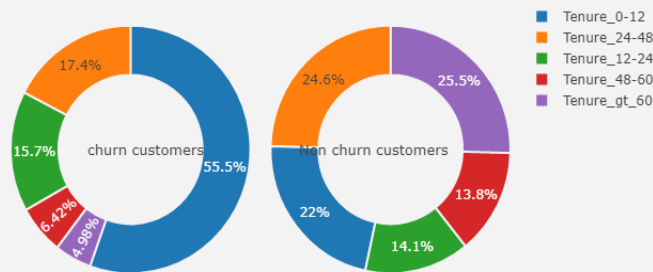
PaperlessBilling distribution in customer attrition



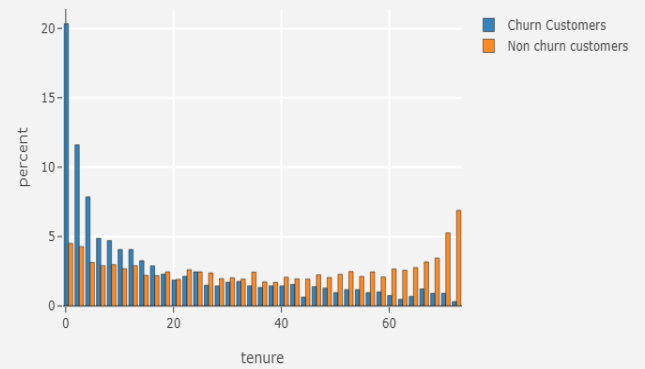
PaymentMethod distribution in customer attrition



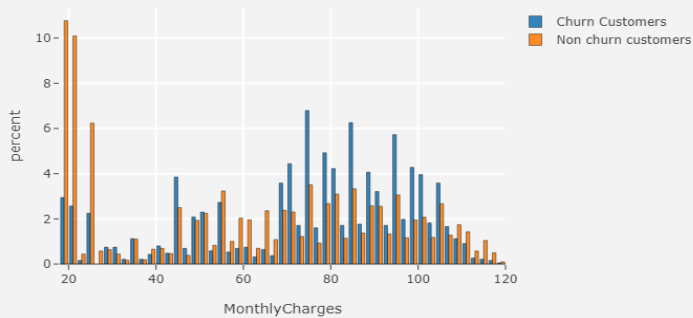
tenure_group distribution in customer attrition



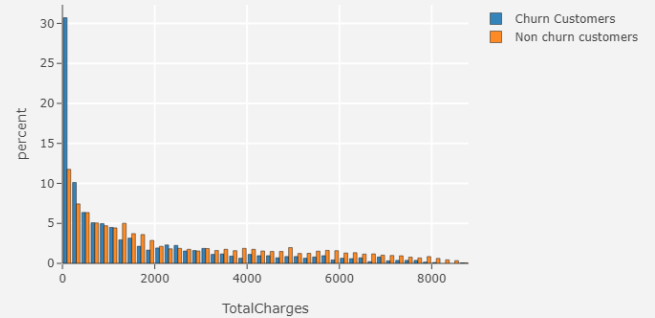
tenure distribution in customer attrition



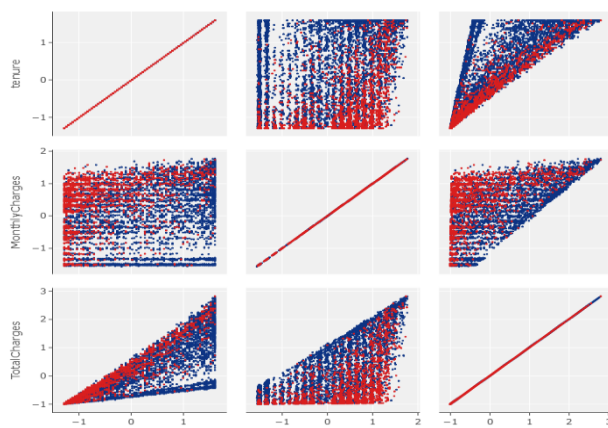
MonthlyCharges distribution in customer attrition



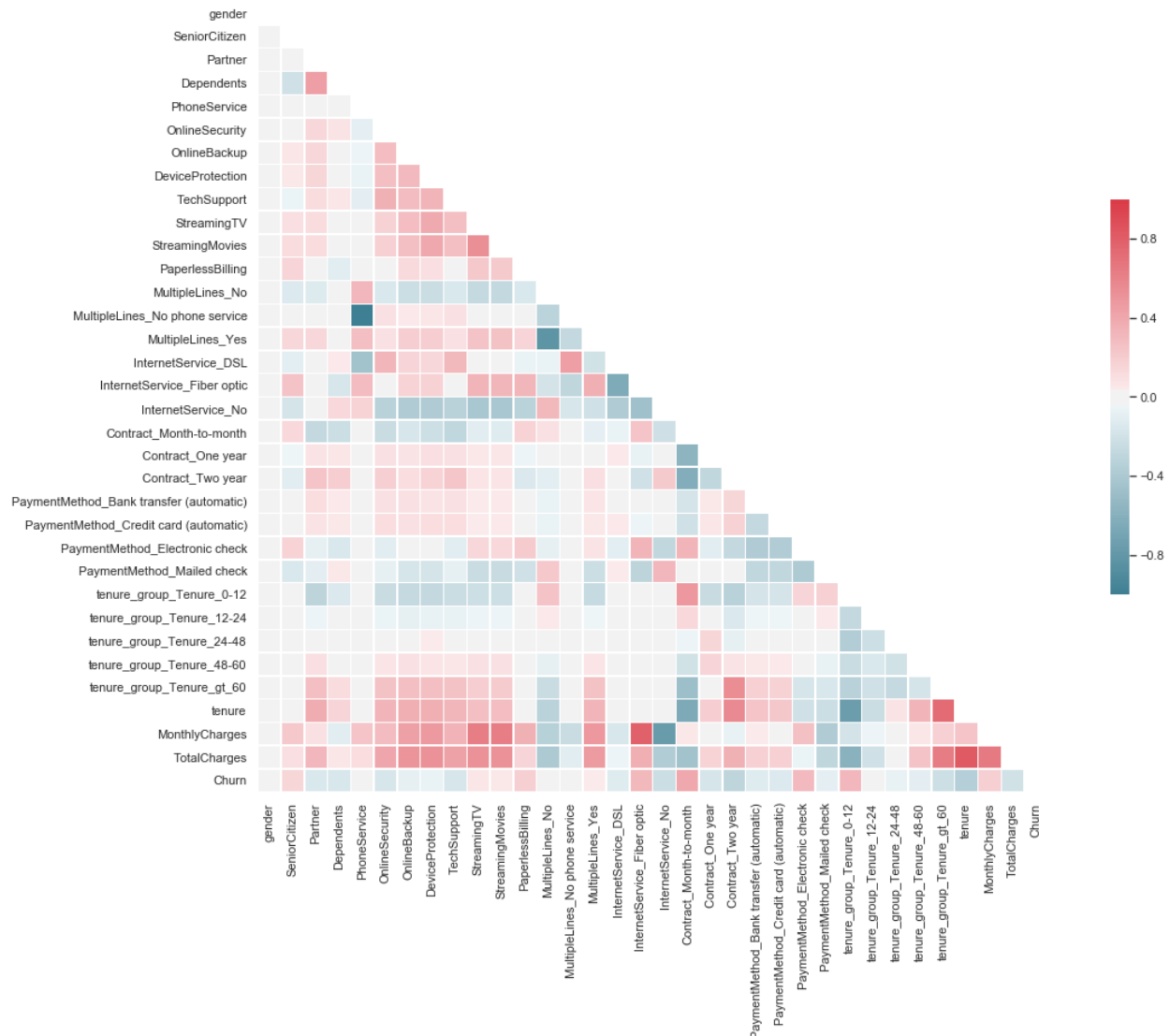
TotalCharges distribution in customer attrition



Scatter plot matrix for Numerical columns for customer attrition



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.



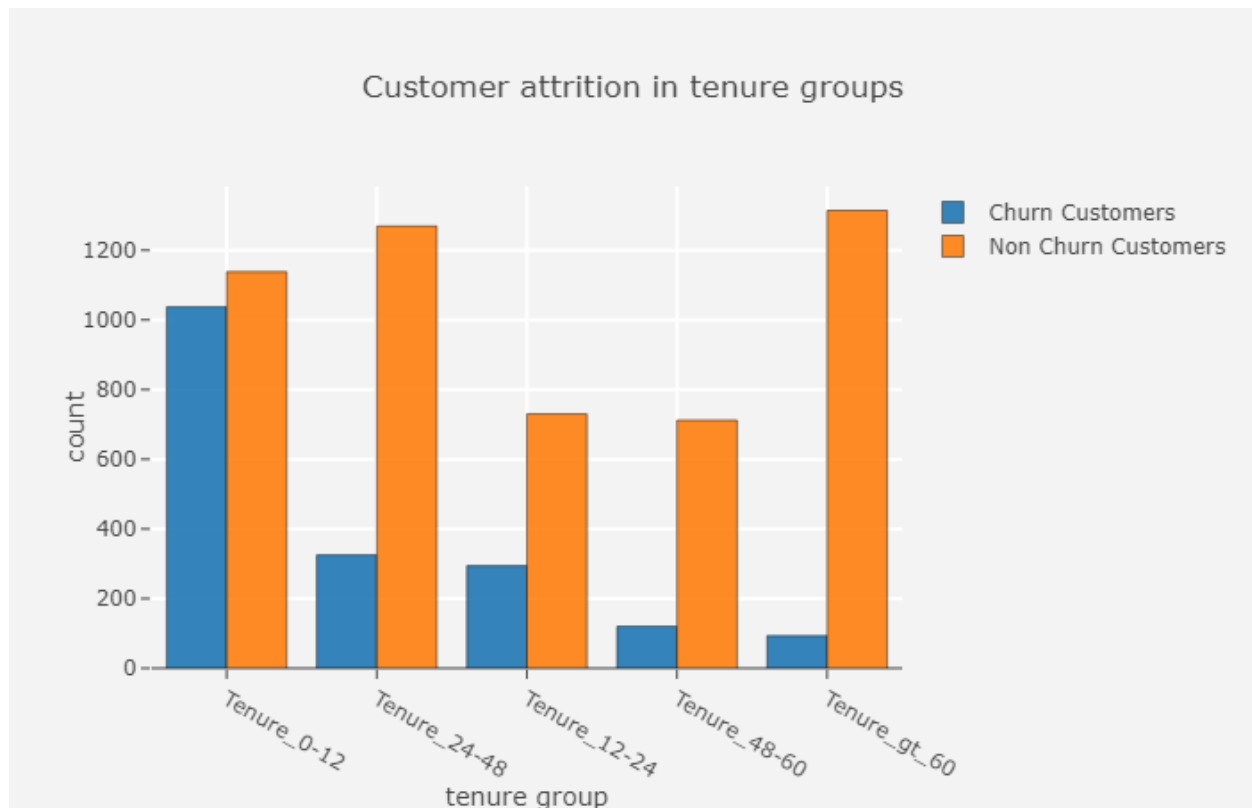
As we see that the gender has almost None correlation to our target Churn

- 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.

Tuning parameters: Kernel, Regularization, Gamma and Margin.

Kernel

The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role.

For **linear kernel** the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

$$f(x) = B(o) + \text{sum}(a_i * (x, x_i))$$

This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B_o and a_i (for each input) must be estimated from the training data by the learning algorithm.

The **polynomial kernel** can be written as $K(x, x_i) = 1 + \sum (x * x_i)^d$ and **exponential** as $K(x, x_i) = \exp(-\gamma \sum ((x - x_i)^2))$.
[Source for this excerpt : <http://machinelearningmastery.com/>].

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

Logistic Regression:

Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical.

For example,

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

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:

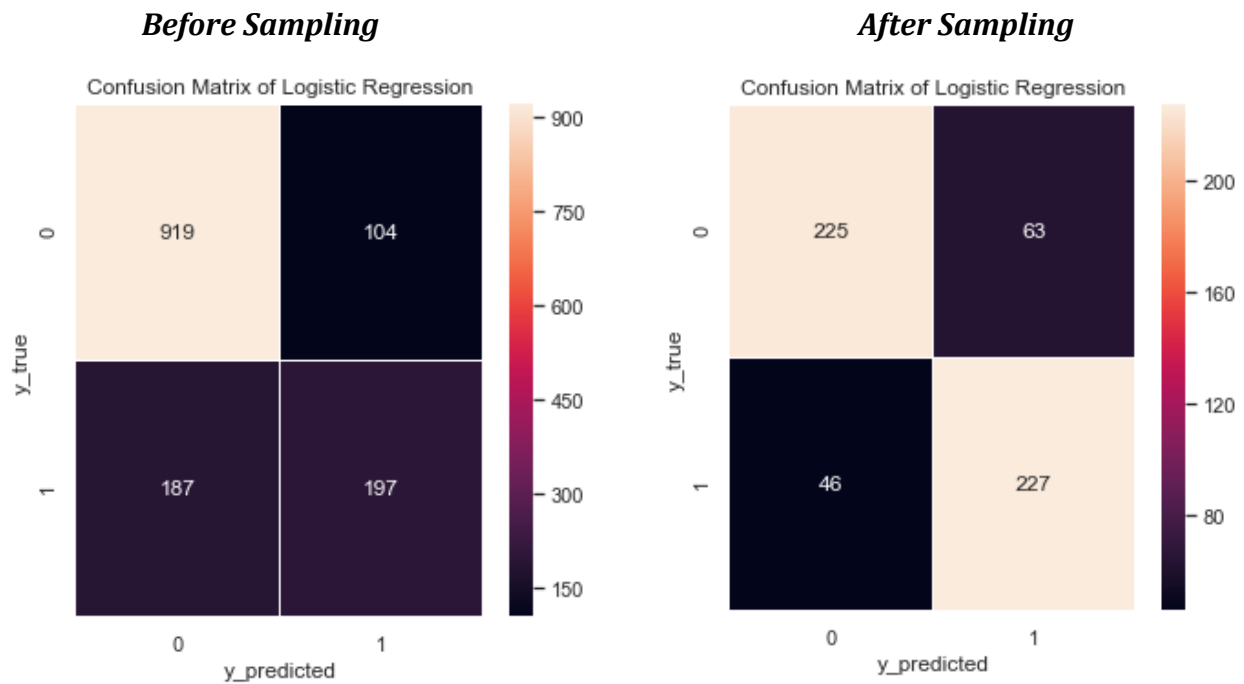
*Resampling techniques are a set of methods to either repeat sampling from a given **sample** or **population**, or a way to estimate the **precision** of a statistic. Although the method sounds daunting, the math involved is relatively simple and only requires a high school level understanding of algebra. 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 be calculated based on the Confusion Matrix Scores.



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

```
telcom = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")
#first few rows
telcom.head()

#Data Manipulation

#Replacing spaces with null values in total charges column
telcom['TotalCharges'] = telcom["TotalCharges"].replace(" ",np.nan)

#Dropping null values from total charges column which contain .15% missing data
telcom = telcom[telcom["TotalCharges"].notnull()]
telcom = telcom.reset_index()[telcom.columns]

#convert to float type
telcom["TotalCharges"] = telcom["TotalCharges"].astype(float)

#replace 'No internet service' to No for the following columns
replace_cols = [ 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
                  'TechSupport', 'StreamingTV', 'StreamingMovies']
for i in replace_cols :
    telcom[i] = telcom[i].replace({'No internet service' : 'No'})

#replace values
telcom["SeniorCitizen"] = telcom["SeniorCitizen"].replace({1:"Yes",0:"No"})

#Tenure to categorical column
def tenure_lab(telcom) :

    if telcom["tenure"] <= 12 :
        return "Tenure_0-12"
    elif (telcom["tenure"] > 12) & (telcom["tenure"] <= 24) :
        return "Tenure_12-24"
    elif (telcom["tenure"] > 24) & (telcom["tenure"] <= 48) :
        return "Tenure_24-48"
    elif (telcom["tenure"] > 48) & (telcom["tenure"] <= 60) :
        return "Tenure_48-60"
    elif telcom["tenure"] > 60 :
        return "Tenure_gt_60"
telcom["tenure_group"] = telcom.apply(lambda telcom:tenure_lab(telcom),
                                     axis = 1)

#Separating churn and non churn customers
churn = telcom[telcom["Churn"] == "Yes"]
not_churn = telcom[telcom["Churn"] == "No"]

#Separating catagorical and numerical columns
Id_col = ['customerID']
target_col = ["Churn"]
cat_cols = telcom.nunique()[telcom.nunique() < 6].keys().tolist()
cat_cols = [x for x in cat_cols if x not in target_col]
num_cols = [x for x in telcom.columns if x not in cat_cols + target_col + Id_col]
```

2.2 Load Dataset after Manipulation

```
#Load the dataset
data = telcom

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

	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



3. Analysis

3.1 Data Exploration

3.1.1 Load DataSet

```
#Load the dataset
data = telcom
data2 = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")

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

	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



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 formatting
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
```

3.1.4 Data Descriptive Analysis

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

	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

3.1.5 Data Unique Values Calculation

```
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', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', '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 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

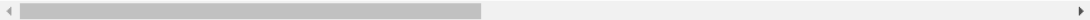
```
In [410]: #Load the dataset after normalization
data = telcom

#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 attributes

```
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	...	Paym
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

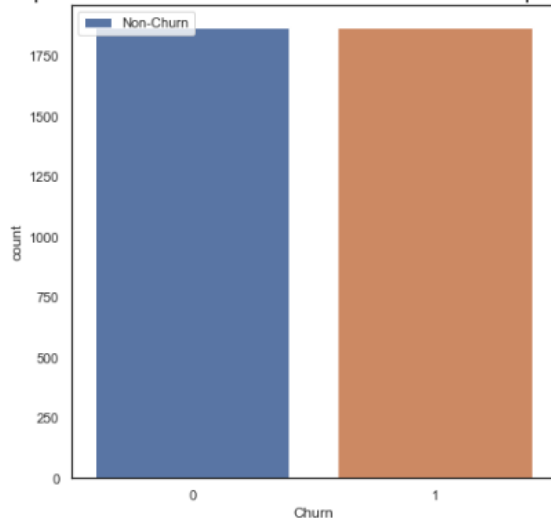
7.1.2 Data Plotting after Resampling

```
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>

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_undersample' 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 gamma 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

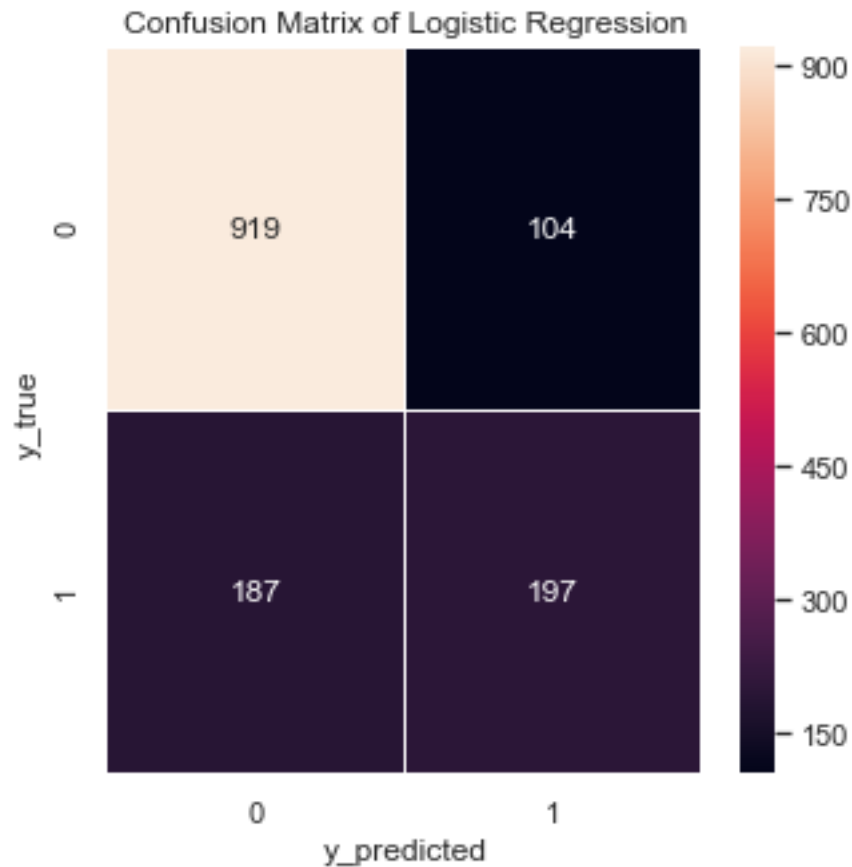
precision score on training set: 0.6844
precision score on testing set: 0.6723

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

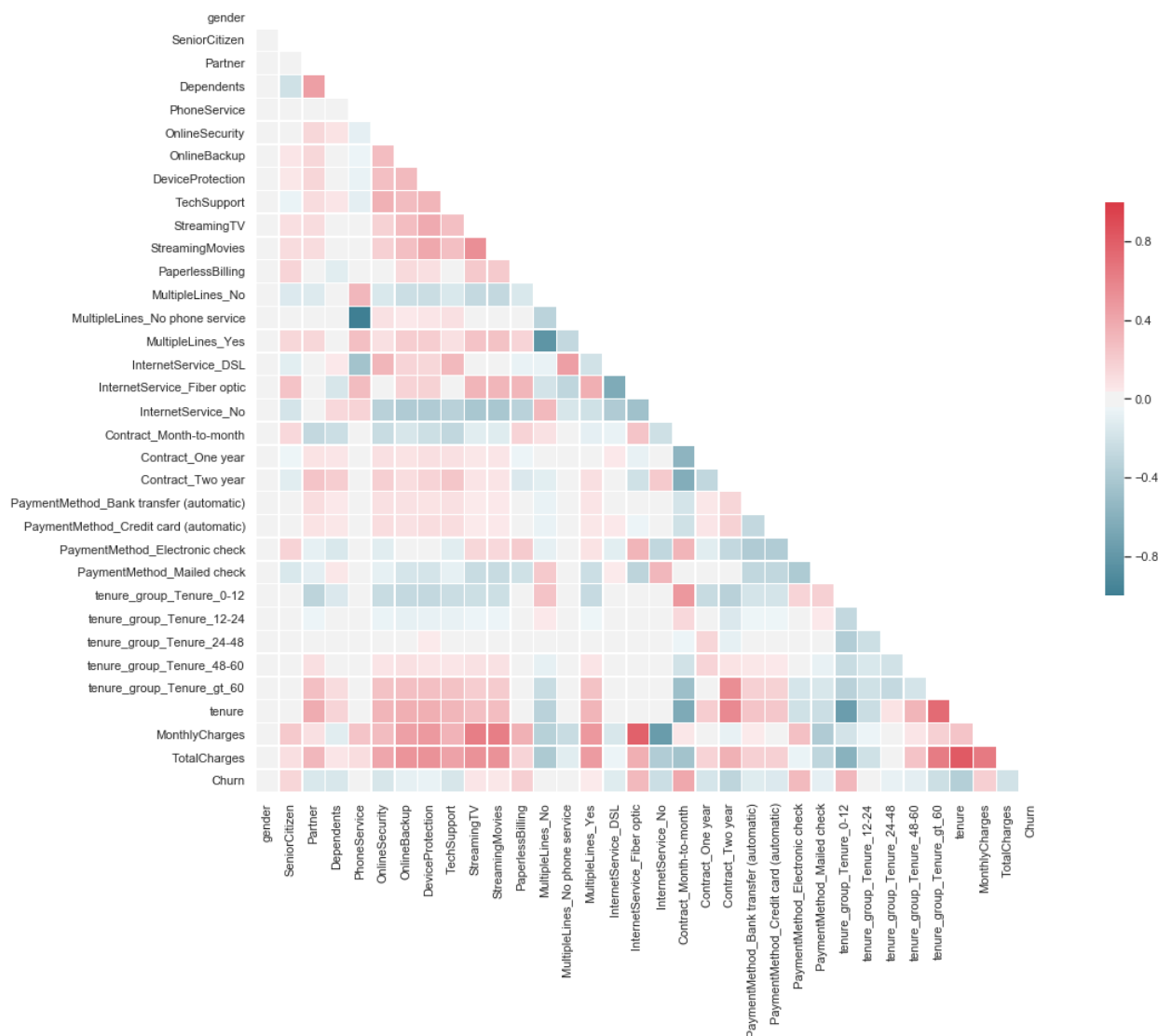
```
report = classification_report(y_test, lr_model.predict(X_test))
print(report)
```

	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.



Then after finding that features like gender, customer id, tenure group 12-24 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 attributes

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

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_undersample' 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

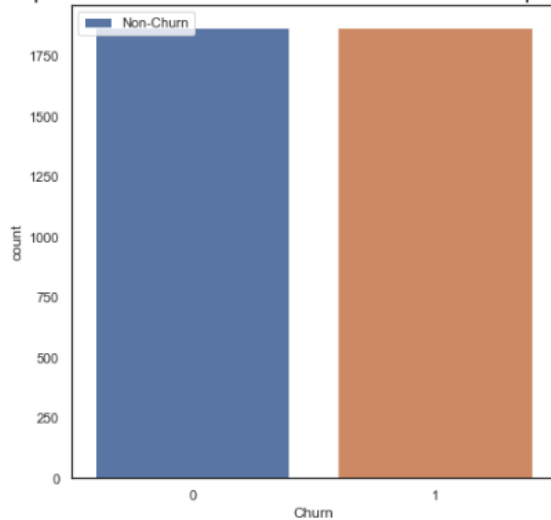
7.1.2 Data Plotting after Resampling

```
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>

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_undersample' 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

- **1. Data & DataSet Import**
 - **1.1 Importing Dataset from Kaggle If Using Google Colab**
 - **1.1.1 Install Kaggle Packages**
 - **1.1.2 Importing Kaggle Api & Creating Kaggle Directory**
 - **1.1.3 Download Dataset**
 - **1.1.4 Unzip Dataset Package**
 - **1.1.5 Function For Running Plotly Graphs on Google Colab**
 - **1.2 Import required Libraries**
- **2. Data Manipulation**
 - **2.1 Creating Cluster groups of Tenures**
 - **2.2 Load Dataset after Manipulation**
- **3. Analysis**
 - **3.1 Data Exploration**
 - **3.1.1 Load DataSet**

- 3.1.2 Checking for Missing Data
 - 3.1.3 Data Format Description
 - 3.1.4 Data Descriptive Analysis
 - 3.1.5 Data Unique Values Calculation
- **4. Methodology**
 - **4.1 Data Preprocessing**
 - 4.1.1 Data Normalization
 - 4.1.2 Loading Data After Normalization
 - 4.1.3 Moving Churn Column to Last for Correlation Matrix Readness Ease
- **5. Analysis - 2**
 - **5.1 Data Visualization**
 - 5.1.1 Churn to Non-Churn Proportion
 - 5.1.2 Statistical Analysis in Customer Attrition
 - 5.1.3 Correlation Matrix
 - 5.1.4 Customer Churn to Non-Churn in Tenure Groups
 - **5.2 Algorithms & Techniques**
 - **5.3 Benchmark**
- **6. Methodology - 2**
 - **6.1 Data Preprocessing - 2**
 - **6.2 Data Cleaning & Refinement of Un-necessary attributes**
 - **6.3 Data Shuffling & Splitting**
 - **6.4 Implementation**
 - 6.4.1 Fit & Train SVM
 - 6.4.2 SVM Scoring on Whole Dataset
 - 6.4.3 Fit & Train Logistic Regression with Accuracy Score
 - 6.4.4 Calculate Confusion Matrix for Log Regression
 - 6.4.5 Description of Confusion Matrix Values
 - 6.4.6 Descriptive Scoring for Logistic Regression
- **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 your data sample.

A poorly chosen value for k may result in a mis-representative idea of the skill of the model, such as a score with a high variance (that may change a lot based on the data used to fit the model), or a high bias, (such as an overestimate of the skill of the model).

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.