Telco Customer Churn

A report submitted in partial fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

SAI PRANAY GANTA

Regd. No.: 20B91A05Q8

Under Supervision of Mr. Gundala Nagaraju

Henotic Technology Pvt Ltd, Hyderabad

(Duration: 7th July, 2022 to 6th September, 2022)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE

(An Autonomous Institution)

Approved by AICTE, NEW DELHI and Affiliated to JNTUK, Kakinada

CHINNA AMIRAM, BHIMAVARAM,

ANDHRA PRADESH

SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE
(Autonomous)
Chinna Amiram, Bhimavaram

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the "**Summer Internship Report**" submitted by **Sai Pranay Ganta, 20B91A05Q8** is work done by him and submitted during 2021 - 2022 academic year, in partial fulfillment of the requirements for the award of the Summer Internship Program for **Bachelor of Technology in Computer Science and Engineering,** at **Henotic Technology Private Ltd** from *07.07.2022 to 06.09.2022 .*

Department Internship Coordinator **Dean -T & P Cell** **Head of the Department**

Table of Contents

Abstract

Predicting customer churn is critical for telecommunication companies to be able to effectively retain customers. It is more costly to acquire new customers than to retain existing ones. For this reason, large telecommunications corporations are seeking to develop models to predict which customers are more likely to change and take actions accordingly.

Customer retention and customer satisfaction are essential for a business to succeed.

1. Customer satisfaction is improved by repeating businesses, brand loyalty, and positive word of mouth.

2. Consumers prefer to stay with their current providers due to quality and price. Therefore, new anti-churn strategies must be constantly developed.

3. Data processing automates analytical model building. Machine learning algorithms improve the data set iteratively to find hidden patterns.

4. Several studies show that machine learning can predict churn and severe problems in competitive service sectors. Predicting churning customers early on can be a valuable revenue source.

The data set used in this article is available in the Kaggle and contains nineteen columns (independent variables) that indicate the characteristics of the clients of a fictional telecommunications corporation.The churn column (response variable) indicates whether the customer departed within the last month or not. The class NO includes the clients that did not leave the company last month, while the class YES contains the clients that decided to terminate their relations with the company. The objective of the analysis is to obtain the relation between the customer’s characteristics and the churn.

1.0 Introduction

With the increasing power of computer technology, companies and institutions can nowadays store large amounts of data at reduced cost. The amount of available data is increasing exponentially and cheap disk storage makes it easy to store data that previously was thrown away. There is a huge amount of information locked up in databases that is potentially important but has not yet been explored. The growing size and complexity of the databases makes it hard to analyse the data manually, so it is important to have automated systems to support the process. Hence there is the need of computational tools able to treat these large amounts of data and extract valuable information.

In this context, Data Mining provides automated systems capable of processing large amounts of data that are already present in databases. Data Mining is used to automatically extract important patterns and trends from databases seeking regular patterns that can reveal the structure of the data and answer business problems. Data Mining includes learning techniques that fall into the field of Machine learning. The growth of databases in recent years brings data mining at the forefront of new business technologies.

A key challenge for the telco industry is to see that each customer gets an appropriate internet connection. Retention of the customer varies widely from customer to customer and a deep understanding of different services provided by the company helps predict the retention of the customer. The goal of this program is to see how well various statistical methods perform in predicting churn of telco customer based on the characteristics of the internet service, Online Backup,Total Charges details and others.

1.1. What are the different types of Machine Learning?

The term ‘machine learning’ is often, incorrectly, interchanged with Artificial Intelligence, but machine learning is actually a sub field/type of AI. Machine learning is also often referred to as predictive analytics, or predictive modelling.

The different types of Machine Learning models are mainly classified as follows:

1. Supervised Learning

Data sets include their desired outputs or labels so that a function can calculate an error for any given prediction. The supervision part comes into play when a prediction is created, and an error is produced to change the function and learn the mapping. Supervised learning’s goal is to create a function that effectively generalizes over data it has never seen.

2. Unsupervised Learning

There are cases where a data set doesn’t have the desired output, so there’s no means of supervising the function. Instead, the process tries to segment the data set into “classes” so that each class has a segment of the data set with common features. Unsupervised learning aims to build a mapping function that classifies data based on features found within the data.

3. Reinforcement Learning

With reinforcement learning, the algorithm tries to learn actions for a given set of states that lead to a goal state. Thus, errors aren’t flagged after each example but rather on receiving a reinforcement signal, like reaching the goal state. This process closely resembles human learning, where feedback isn’t provided for every action, only when the situation calls for a reward.

1.2. Benefits of Using Machine Learning in Telco Customer

With the enormous increase in the number of customers using telephone services, the marketing division for a telco company wants to attract more new customers and avoid contract termination from existing customers (churn rate). For the telco company to expand its clientele, its growth rate (number of new customers) must exceed its churn rate (number of customers existing). Some of the factors that caused existing customers to leave their telco companies are better price offers, faster internet services, and a more secure online experience from other companies.

A high churn rate will adversely affect a company's profits and impede growth. Our churn prediction would be able to provide clarity to the telco company on how well it is retaining its existing customers and understand what are the underlying reasons that are causing existing customers to terminate their contract (high churn rate).

The telco company can use our analysis to measure if it is providing a useful product compared with the product provided by its competitors. Since the cost of acquiring new customers is much higher than retaining its existing customers, the company can use the churn rate analysis to provide discounts, special offers, and superior products to keep current customers.

1.3. About Industry (Telco industry)

The telecommunication sector is made up of companies that make communication possible on a global scale, whether it is through the phone or the Internet, through airwaves or cables, through wires or wire lessly. These companies created the infrastructure that allows data in words, voice, audio, or video to be sent anywhere in the world. The largest companies in the sector are telephone (both wired and wireless) operators, satellite companies, cable companies, and internet service providers.

1.3.1 AI / ML Role in Telco Customer Industry

With the rapid development of telecommunication industry, the service providers are inclined more towards expansion of the subscriber base. To meet the need of surviving in the competitive environment, the retention of existing customers has become a huge challenge. In the survey done in the Telecom industry, it is stated that the cost of acquiring a new customer is far more that retaining the existing one. Therefore, by collecting knowledge from the telecom industries can help in predicting the association of the customers as whether or not they will leave the company. The required action needs to be undertaken by the telecom industries in order to initiate the acquisition of their associated customers for making their market value stagnant. Our paper proposes a new framework for the churn prediction model and implements it using the weka

Data Mining software. The efficiency and the performance of Decision tree and Logistic regression techniques have been compared.

2.0 Telco Customer Churn (Internship project)

Customer churn is often referred to as customer attrition, or customer defection which is the rate at which the customers are lost. Telecom companies often use customer churn as a key business metrics to predict the number of customers that will leave a telecom service provider. Churn is significant in the telecommunication industry because it directly affects the competitiveness of the service provider. In this work, Logistic Regression and Random Forest performed better than Decision Tree for customer churn analysis.

This data contains information about a fictitious telecom company that provided home phone and Internet services to 7,043 California residents in the third quarter. This information shows whether a customer has left, remained or opted to sign up for the service.

Customer ID: A unique ID that identifies each customer.

Gender: The customer’s gender: Male, Female

Age: The customer’s current age, in years, at the time the fiscal quarter ended.

Senior Citizen: Indicates if the customer is 65 or older: Yes, No

Married: Indicates if the customer is married: Yes, No

Dependents: Indicates if the customer lives with any dependents: Yes, No. Dependents could be children, parents, grandparents, etc.

Number of Dependents: Indicates the number of dependents that live with the customer.

Phone Service: Indicates if the customer subscribes to home phone service with the company: Yes, No

Multiple Lines: Indicates if the customer subscribes to multiple telephone lines with the company: Yes, No

Internet Service: Indicates if the customer subscribes to Internet service with the company: No, DSL, Fiber Optic, Cable.

Online Security: Indicates if the customer subscribes to an additional online security service provided by the company: Yes, No

Online Backup: Indicates if the customer subscribes to an additional online backup service provided by the company: Yes, No

Device Protection Plan: Indicates if the customer subscribes to an additional device protection plan for their Internet equipment provided by the company: Yes, No

Premium Tech Support: Indicates if the customer subscribes to an additional technical support plan from the company with reduced wait times: Yes, No

Streaming TV: Indicates if the customer uses their Internet service to stream television programming from a third party provider: Yes, No. The company does not charge an additional fee for this service.

Streaming Movies: Indicates if the customer uses their Internet service to stream movies from a third party provider: Yes, No. The company does not charge an additional fee for this service.

Contract: Indicates the customer's current contract type: Month-to-Month, One Year, Two Year.

Paperless Billing: Indicates if the customer has chosen paperless billing: Yes, No

Payment Method: Indicates how the customer pays their bill: Bank Withdrawal, Credit Card, Mailed Check

Monthly Charge: Indicates the customer's current total monthly charge for all their services from the company.

Charges: Indicates the customer's total charges, calculated to the end of the quarter specified above.

Tenure: Indicates the total amount of months that the customer has been with the company.

Churn: Yes = the customer left the company this quarter. No = the customer remained with the company. Directly related to Churn Value.

2.1 Main Drivers for AI Teleco Customer Analysis

Predictive modelling allows for simultaneous consideration of many variables and quantification of their overall effect. When a large number of data is analyzed, patterns regarding the characteristics of the telco customer data that drive loss development begin to emerge.

The following are the main drivers which influencing the Telco Customer churn :

- **Phone Service**
  - ✓ Present
  - ✓ Not Present
- **Multiple Lines**
  - ✓ Present
  - ✓ Not Present
  - ✓ Phone Service not present
- **Internet Service**
  - ✓ Through Digital Subscriber Line (DSL)
  - ✓ Through Fiber Optic
  - ✓ No Internet Service
- **Online Security**
  - ✓ Present
  - ✓ Not Present
  - ✓ No Internet Service
- **Online Backup**
  - ✓ Yes
  - ✓ No
  - ✓ No Internet Service
- **Device Protection**
  - ✓ Present
  - ✓ Not Present
  - ✓ No Internet Service
- **Streaming TV**
  - ✓ Yes
  - ✓ No
  - ✓ No Internet Service
- **Streaming Movies**
  - ✓ Yes
  - ✓ No
  - ✓ No Internet Service
- **Contract**
  - ✓ One Year
  - ✓ Two Year
  - ✓ Month-To-Month
- **Paperless Bill**
  - ✓ Yes
  - ✓ No
- **Payment Method**
  - ✓ Electronic Mail
  - ✓ Mailed Check
  - ✓ Bank Transfer
  - ✓ Credit Card
- **Monthly Charges**
  - ✓ Amount Paid Monthly
- **Total Charges**

• **Tech Support** ✓ Present ✓ Not Present ✓ No Internet Service	✓ Amount Paid Annually • **Churn** ✓ Will the customer continue or quit the services

2.2 Internship Project - Data Link

For Telco companies it is key to attract new customers and at the same time avoid contract terminations (churn) to grow their revenue generating base. Looking at churn, different reasons trigger customers to terminate their contracts, for example better price offers, more interesting packages, bad service experiences or change of customers' personal situations.

The internship project data has taken from Kaggle and the link is given below -

https://www.kaggle.com/datasets/blastchar/telco-customer-churn

3.0 AI / ML Modelling and Results

3.1 Problem of Statement

Predictive models are most effective when they are constructed using a company's own historical telco customer data since this allows the model to recognize the specific nature of a company's exposure as well as its claims practices.The construction of the model also involves input from the company throughout the process, as well as consideration of industry leading telco customer data and benchmarks.

1. Predictive modelling can be used to quantify the impact to the claims department resulting from the failure to meet or exceed claim service leading practices. It can also be used to identify the root cause of claim leakage.Proper use of predictive modelling will allow for potential savings across two dimensions:

2. Early identification of claims with the potential for high leakage, thereby allowing for the proactive management of the claim

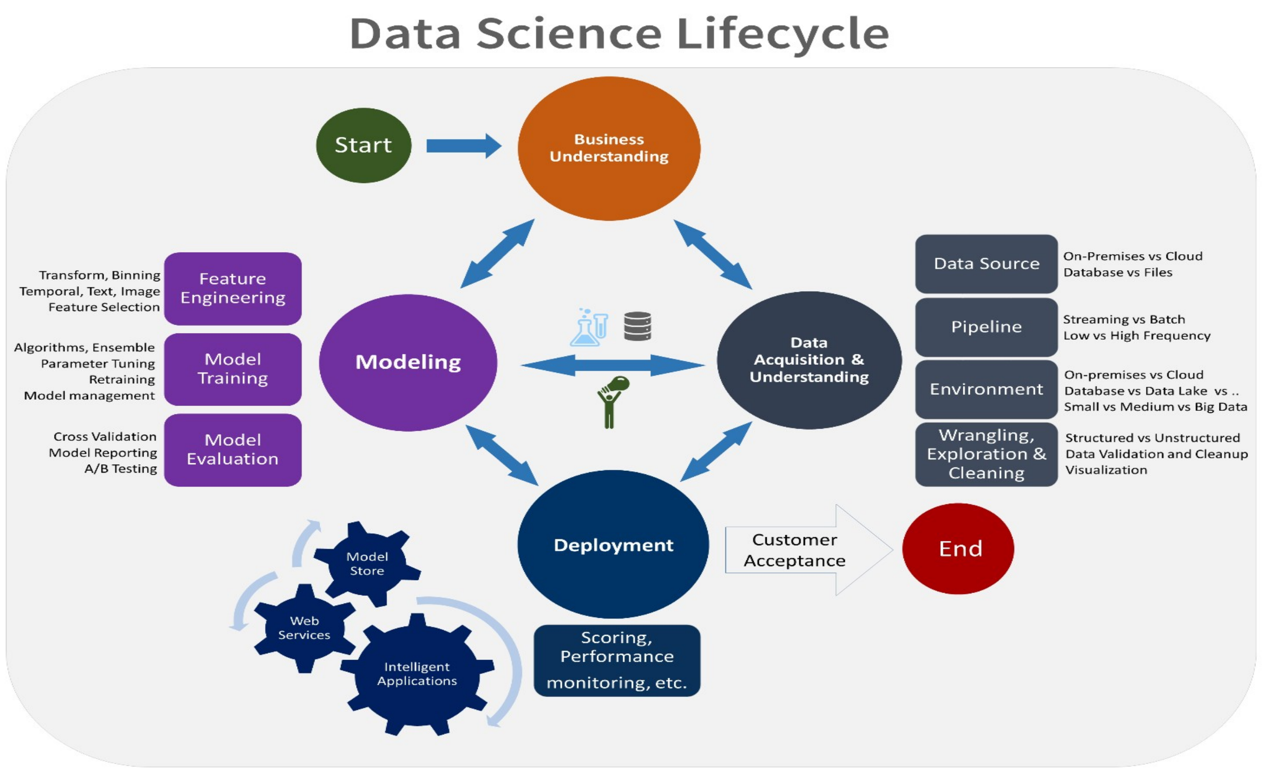
3. Recognition of practices that are unnecessarily increasing claims settlement payments.

3.2 Data Science Project Life Cycle

Data Science is a multidisciplinary field of study that combines programming skills, domain expertise and knowledge of statistics and mathematics to extract useful insights and knowledge from data.

3.2.1 Data Exploratory Analysis

Exploratory data analysis has been done on the data to look for relationship and correlation between different variables and to understand how they impact or target variable.



3.2.2 Data Pre-processing

We removed variables which does not affect our target variable(Churn_Target)as they may add noise and also increase our computation time,we checked the data for anomalous data points and outliers. We did principal component analysis on the data set to filter out unnecessary variables and to select only the important variables which have greater correlation with our target variable.

3.2.2.1 Check the Duplicate and low variation data

To find duplicates on a specific column, we can simply call duplicated() method on the column. The result is a Boolean Series with the value True denoting duplicate. In other words, the value True means the entry is identical to a previous one.

3.2.2.2 Identify and address the missing variables

There are no missing variables present in the data.

3.2.2.3 Handling of Outliers

An outlier is an observation that lies an abnormal distance from other values in a random sample from a population. In a sense, this definition leaves it up to the analyst (or a consensus process) to decide what will be considered abnormal.

The outliers present in the data have been removed using the Gaussian distribution (also known as normal distribution) method in which a histogram is drawn and the outliers the removed.There is not a much difference in the accuracy results after removing the outliers.

3.2.2.4 Categorical data and Encoding Techniques

The categorical data is encoded with label encoder fit_transform function which is present in the sklearn module.

3.2.2.5 Feature Scaling

The two most popular techniques for scaling numerical data prior to modeling are normalization and standardization. Normalization scales each input variable separately to the range 0-1, which is the range for floating-point values where we have the most precision. Standardization scales each input variable separately by subtracting the mean (called centering) and dividing by the standard deviation to shift the distribution to have a mean of zero and a standard deviation of one.

MinMaxScaler function from the sklearn preprocessing model is used for the feature scaling in the project.

3.2.3 Selection of Dependent and Independent variables

The dependent or target variable here is Churn, which tells us whether a particular customer will continue the internet services or the target variable is selected based on our business problem and what we are trying to predict.

The independent variables are selected after doing exploratory data analysis and we used Boruta to select which variables are most affecting our target variable.

3.2.4 Data Sampling Methods

The data we have is highly unbalanced data so we used some sampling methods which are used to balance the target variable so we our model will be developed with good accuracy and precision. We used three Sampling methods.

3.2.4.1 Stratified sampling

Stratified sampling randomly selects data points from majority class so they will be equal to the data points in the minority class. So, after the sampling both the class will have same no of observations.It can be performed using strata function from the library sampling.

3.2.4.2 Simple random sampling

Simple random sampling is a sampling technique where a set percentage of the data is selected randomly. It is generally done to reduce bias in the data set which can occur if data is selected manually without randomizing the data set.

We used this method to split the data set into train data set which contains 70% of the total data and test data set with the remaining 30% of the data.

3.2.5 Models Used for Development

We built our predictive models by using the following ten algorithms:

3.2.5.1 Logistic Regression

Logistic regression, despite its name, is a classification model rather than regression model. Logistic regression is a simple and more efficient method for binary and linear classification problems. It is a classification model, which is very easy to realize and achieves very good performance with linearly separable classes. It is an extensively employed algorithm for classification in industry. The logistic regression model, like the Adaline and perceptron, is a statistical method for binary classification that can be generalized to multiclass classification.

Scikit-learn has a highly optimized version of logistic regression implementation, which supports multiclass classification task.

3.2.5.2 Decision Tree Classifier

Decision tree is one of the predictive modelling approaches used in statistics, data mining and machine learning. Decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. Tree models where the target variable can take a discrete set of values are called classification trees. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. Classification And Regression Tree (CART) is general term for this.

3.2.5.3 Random Forest Classifier

Random forest is an algorithm that consists of many decision trees. It was first developedby Leo Breiman and Adele Cutler. The idea behind it is to build several trees,to have the instance classified by each tree, and to give a "vote" at each class.The model uses a "bagging" approach and the random selection of features to build a collection of decision trees with controlled variance. The instance's class is to the class with the highest number of votes, the class that occurs the most within the leaf in which the instance is placed.

The error of the forest depends on:

- Trees correlation: the higher the correlation, the higher the forest error rate.
- The strength of each tree in the forest. A strong tree is a tree with low error. By using trees that classify the instances with low error the error rate of the forest decreases.

3.2.5.4 Extra Trees Classifier

Extra Trees Classifier is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a “forest” to output it’s classification result.

In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest.

Each Decision Tree in the Extra Trees Forest is constructed from the original training sample. Then, at each test node, Each tree is provided with a random sample of k features from the feature-set from which each decision tree must select the best feature to split the data based on some mathematical criteria (typically the Gini Index). This random sample of features leads to the creation of multiple de-correlated decision trees.

3.2.5.5 KNeighborsClassifier

The k-nearest neighbors algorithm, also known as KNN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

3.2.5.6 Gaussian NB

Model description Artificial neural networks can theoretically solve any problem. ANNs can identify hidden patterns between the variables and can find how different combinations of variables can affect the target variable. The error correction is done by gradient descent algorithm which can reduce the error rate as much as possible for the given data.

3.2.5.6 Support Vector Machine

Support vector machines (SVMs) are powerful yet flexible supervised machine learning methods used for classification, regression, and, outliers detection. SVM’s are very efficient in high dimensional spaces and generally are used in classification problems. SVM’s are popular and memory efficient because they use a subset of training points in the decision function. The main goal of SVMs is to divide the data sets into number of classes in order to find a maximum marginal hyper plane (MMH) which can be done by Support Vector Machines will first generate hyper planes iteratively that separates the classes in the best way. After that it will choose the hyper plane that segregate the classes correctly.

3.2.5.8 Bagging Classifier

Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it. Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples(or data) from the original training dataset – where N is the size of the original training set. Training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out.

3.2.5.9 Gradient Boosting Classifier

Gradient boosting algorithm is one of the most powerful algorithms in the field of machine learning. As we know that the errors in machine learning algorithms are broadly classified into two categories i.e. Bias Error and Variance Error. As gradient boosting is one of the boosting algorithms it is used to minimize bias error of the model. Unlike, Adaboosting algorithm, the base estimator in the gradient boosting algorithm cannot be mentioned by us. The base estimator for the Gradient Boost algorithm is fixed and i.e. *Decision Stump*. Like, AaBoost, we can tune the n_estimator of the gradient boosting algorithm. However, if we do not mention the value of n_estimator, the default value of n_estimator for this algorithm is Gradient boosting algorithm can be used for predicting not only continuous target variable (as a Regressor) but also categorical target variable (as a Classifier). When it is used as a regressor, the cost function is Mean Square Error (MSE) and when it is used as a classifier then the cost function is Log loss.

3.2.5.10 Light GBM Classifier

LightGBM is a gradient boosting framework based on decision trees to increases the efficiency of the model and reduces memory usage. It uses two novel techniques: Gradient-based One Side Sampling and Exclusive Feature Bundling (EFB) which fulfills the limitations of histogram-based algorithm that is primarily used in all GBDT (Gradient Boosting Decision

Tree) frameworks. The two techniques of GOSS and EFB described below form the characteristics of LightGBM Algorithm. They comprise together to make the model work efficiently and provide it a cutting edge over other GBDT frameworks.

1.4. AI / ML Models Analysis and Final Results

We used our train data set to build the above models and used our test data to check the accuracy and performance of our models.

We used confusion matrix to check accuracy, Precision, Recall and F1 score of our models and compare and select the best model for given auto data set of size ~ 7044 policies.

Different Model codes

1. Logistic Regression Python Code

```
from sklearn.linear_model import LogisticRegression
models = LogisticRegression()
#Fit the model
models.fit(x_train, y_train)
 y_pred = models.predict(x_test)
 y_pred_prob = models.predict_proba(x_test)
 print('Model Name: ', models)
 # confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
 # confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
 print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
```

```
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
 balanced_accuracy = round((sensitivity+specificity)/2, 3);
 precision = round(tp/(tp+fp), 3);
 f1Score = round((2*tp/(2*tp + fp + fn)), 3);
 # Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
 # A model with a score of +1 is a perfect model and -1 is a poor model
 from math import sqrt
 mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
 MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
 print('Accuracy :', round(accuracy*100, 2),'%')
 print('Precision :', round(precision*100, 2),'%')
 print('Recall :', round(sensitivity*100,2), '%')
 print('F1 Score :', f1Score)
 print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
 print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
 print('MCC :', MCC)
 # Area under ROC curve
 from sklearn.metrics import roc_curve, roc_auc_score
 print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
  # ROC Curve
  from sklearn.metrics import roc_auc_score
  from sklearn.metrics import roc_curve
  logit_roc_auc = roc_auc_score(actual, predicted)
  fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
  plt.figure()
  # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
  plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
  plt.plot([0, 1], [0, 1],'r--')
  plt.xlim([0.0, 1.0])
  plt.ylim([0.0, 1.05])
  plt.xlabel('False Positive Rate')
  plt.ylabel('True Positive Rate')
  plt.title('Receiver operating characteristic')
  plt.legend(loc="lower right")
  plt.savefig('Log_ROC')
  plt.show()
  new_row = {'Model Name' : models,
            'True Positive' : tp,
            'False Negative' : fn,
            'False Positive' : fp,
            'True Negative' : tn,
            'Accuracy' : accuracy,
            'Precision' : precision,
            'Recall' : sensitivity,
            'F1 Score' : f1Score,
            'Specificity' : specificity,
            'MCC':MCC,
            'ROC_AUC_Score':roc_auc_score(y_test, y_pred),
            'Balanced Accuracy':balanced_accuracy}
  CSResults = CSResults.append(new_row, ignore_index=True)
```

2. Decision Tree Classifier Python Code

```
from sklearn.tree import DecisionTreeClassifier
models= DecisionTreeClassifier()
#Fit the model
models.fit(x_train, y_train)
 y_pred = models.predict(x_test)
 y_pred_prob = models.predict_proba(x_test)
 print('Model Name: ', models)
 # confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
 # confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
 print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
  balanced_accuracy = round((sensitivity+specificity)/2, 3);
  precision = round(tp/(tp+fp), 3);
  f1Score = round((2*tp/(2*tp + fp + fn)), 3);
  # Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
  # A model with a score of +1 is a perfect model and -1 is a poor model
  from math import sqrt
  mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
  MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
  print('Accuracy :', round(accuracy*100, 2),'%')
  print('Precision :', round(precision*100, 2),'%')
  print('Recall :', round(sensitivity*100,2), '%')
  print('F1 Score :', f1Score)
  print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
  print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
  print('MCC :', MCC)
  # Area under ROC curve
  from sklearn.metrics import roc_curve, roc_auc_score
```

```
print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
 # ROC Curve
 from sklearn.metrics import roc_auc_score
 from sklearn.metrics import roc_curve
 logit_roc_auc = roc_auc_score(actual, predicted)
 fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
 plt.figure()
 # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
 plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
 plt.plot([0, 1], [0, 1],'r--')
 plt.xlim([0.0, 1.0])
 plt.ylim([0.0, 1.05])
 plt.xlabel('False Positive Rate')
 plt.ylabel('True Positive Rate')
 plt.title('Receiver operating characteristic')
 plt.legend(loc="lower right")
 plt.savefig('Log_ROC')
 plt.show()
 new_row = {'Model Name' : models,
            'True Positive' : tp,
            'False Negative' : fn,
            'False Positive' : fp,
            'True Negative' : tn,
            'Accuracy' : accuracy,
            'Precision' : precision,
            'Recall' : sensitivity,
            'F1 Score' : f1Score,
            'Specificity' : specificity,
            'MCC':MCC,
            'ROC_AUC_Score':roc_auc_score(y_test, y_pred),
            'Balanced Accuracy':balanced_accuracy}
 CSResults = CSResults.append(new_row, ignore_index=True)
```

3. Random Forest Classifier Python Code

```
from sklearn.ensemble import RandomForestClassifier
models = RandomForestClassifier()
#Fit the model
models.fit(x_train, y_train)
 y_pred = models.predict(x_test)
 y_pred_prob = models.predict_proba(x_test)
 print('Model Name: ', models)
 # confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
 # confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
 print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
  balanced_accuracy = round((sensitivity+specificity)/2, 3);
  precision = round(tp/(tp+fp), 3);
  f1Score = round((2*tp/(2*tp + fp + fn)), 3);
  # Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
  # A model with a score of +1 is a perfect model and -1 is a poor model
  from math import sqrt
  mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
```

```
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
 # ROC Curve
 from sklearn.metrics import roc_auc_score
 from sklearn.metrics import roc_curve
 logit_roc_auc = roc_auc_score(actual, predicted)
 fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
 plt.figure()
 # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
 plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
 plt.plot([0, 1], [0, 1],'r--')
 plt.xlim([0.0, 1.0])
 plt.ylim([0.0, 1.05])
 plt.xlabel('False Positive Rate')
 plt.ylabel('True Positive Rate')
 plt.title('Receiver operating characteristic')
 plt.legend(loc="lower right")
 plt.savefig('Log_ROC')
 plt.show()
 new_row = {'Model Name' : models,
            'True Positive' : tp,
            'False Negative' : fn,
            'False Positive' : fp,
            'True Negative' : tn,
            'Accuracy' : accuracy,
            'Precision' : precision,
            'Recall' : sensitivity,
            'F1 Score' : f1Score,
            'Specificity' : specificity,
            'MCC':MCC,
            'ROC_AUC_Score':roc_auc_score(y_test, y_pred),
            'Balanced Accuracy':balanced_accuracy}
  CSResults = CSResults.append(new_row, ignore_index=True)
```

4. Extra Tress Classifier Python Code

```
from sklearn.ensemble import ExtraTreesClassifier
models = ExtraTreesClassifier()
#Fit the model
models.fit(x_train, y_train)
 y_pred = models.predict(x_test)
 y_pred_prob = models.predict_proba(x_test)
 print('Model Name: ', models)
 # confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
 # confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
 print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
```

```
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
 balanced_accuracy = round((sensitivity+specificity)/2, 3);
 precision = round(tp/(tp+fp), 3);
 f1Score = round((2*tp/(2*tp + fp + fn)), 3);
 # Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
 # A model with a score of +1 is a perfect model and -1 is a poor model
 from math import sqrt
 mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
 MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
 print('Accuracy :', round(accuracy*100, 2),'%')
 print('Precision :', round(precision*100, 2),'%')
 print('Recall :', round(sensitivity*100,2), '%')
 print('F1 Score :', f1Score)
 print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
 print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
 print('MCC :', MCC)
 # Area under ROC curve
 from sklearn.metrics import roc_curve, roc_auc_score
 print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
  # ROC Curve
  from sklearn.metrics import roc_auc_score
  from sklearn.metrics import roc_curve
  logit_roc_auc = roc_auc_score(actual, predicted)
  fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
  plt.figure()
  # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
  plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
  plt.plot([0, 1], [0, 1],'r--')
  plt.xlim([0.0, 1.0])
  plt.ylim([0.0, 1.05])
  plt.xlabel('False Positive Rate')
  plt.ylabel('True Positive Rate')
  plt.title('Receiver operating characteristic')
  plt.legend(loc="lower right")
  plt.savefig('Log_ROC')
  plt.show()
  new_row = {'Model Name' : models,
           'True Positive' : tp,
           'False Negative' : fn,
           'False Positive' : fp,
           'True Negative' : tn,
           'Accuracy' : accuracy,
           'Precision' : precision,
           'Recall' : sensitivity,
           'F1 Score' : f1Score,
           'Specificity' : specificity,
           'MCC':MCC,
           'ROC_AUC_Score':roc_auc_score(y_test, y_pred),
           'Balanced Accuracy':balanced_accuracy}
  CSResults = CSResults.append(new_row, ignore_index=True)
```

5. KNeighbours Classifier Python Code

```
from sklearn.neighbors import KNeighborsClassifier
ModelKNN = KNeighborsClassifier(n_neighbors=5)
#Fit the model
models.fit(x_train, y_train)
 y_pred = models.predict(x_test)
 y_pred_prob = models.predict_proba(x_test)
 print('Model Name: ', models)
 # confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
 # confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
 print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
```

```
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
 balanced_accuracy = round((sensitivity+specificity)/2, 3);
 precision = round(tp/(tp+fp), 3);
 f1Score = round((2*tp/(2*tp + fp + fn)), 3);
 # Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
 # A model with a score of +1 is a perfect model and -1 is a poor model
 from math import sqrt
 mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
 MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
 print('Accuracy :', round(accuracy*100, 2),'%')
 print('Precision :', round(precision*100, 2),'%')
 print('Recall :', round(sensitivity*100,2), '%')
 print('F1 Score :', f1Score)
 print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
 print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
 print('MCC :', MCC)
 # Area under ROC curve
 from sklearn.metrics import roc_curve, roc_auc_score
 print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
  # ROC Curve
  from sklearn.metrics import roc_auc_score
  from sklearn.metrics import roc_curve
  logit_roc_auc = roc_auc_score(actual, predicted)
  fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
  plt.figure()
  # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
  plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
  plt.plot([0, 1], [0, 1],'r--')
  plt.xlim([0.0, 1.0])
  plt.ylim([0.0, 1.05])
  plt.xlabel('False Positive Rate')
  plt.ylabel('True Positive Rate')
  plt.title('Receiver operating characteristic')
  plt.legend(loc="lower right")
  plt.savefig('Log_ROC')
  plt.show()
  new_row = {'Model Name' : models,
            'True Positive' : tp,
            'False Negative' : fn,
            'False Positive' : fp,
            'True Negative' : tn,
            'Accuracy' : accuracy,
            'Precision' : precision,
            'Recall' : sensitivity,
            'F1 Score' : f1Score,
            'Specificity' : specificity,
            'MCC':MCC,
            'ROC_AUC_Score':roc_auc_score(y_test, y_pred),
            'Balanced Accuracy':balanced_accuracy}
  CSResults = CSResults.append(new_row, ignore_index=True)
```

6. Support Vector Machine Python Code

```
from sklearn.svm import SVC
models = SVC(probability=True)
#Fit the model
models.fit(x_train, y_train)
 y_pred = models.predict(x_test)
 y_pred_prob = models.predict_proba(x_test)
 print('Model Name: ', models)
 # confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

```
# actual values
actual = y_test
# predicted values
predicted = y_pred
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
 print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
  balanced_accuracy = round((sensitivity+specificity)/2, 3);
  precision = round(tp/(tp+fp), 3);
  f1Score = round((2*tp/(2*tp + fp + fn)), 3);
  # Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
  # A model with a score of +1 is a perfect model and -1 is a poor model
  from math import sqrt
  mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
  MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
  print('Accuracy :', round(accuracy*100, 2),'%')
  print('Precision :', round(precision*100, 2),'%')
  print('Recall :', round(sensitivity*100,2), '%')
  print('F1 Score :', f1Score)
  print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
  print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
  print('MCC :', MCC)
  # Area under ROC curve
  from sklearn.metrics import roc_curve, roc_auc_score
  print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
   # ROC Curve
   from sklearn.metrics import roc_auc_score
   from sklearn.metrics import roc_curve
   logit_roc_auc = roc_auc_score(actual, predicted)
   fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
   plt.figure()
   # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
   plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
   plt.plot([0, 1], [0, 1],'r--')
   plt.xlim([0.0, 1.0])
   plt.ylim([0.0, 1.05])
   plt.xlabel('False Positive Rate')
   plt.ylabel('True Positive Rate')
   plt.title('Receiver operating characteristic')
   plt.legend(loc="lower right")
   plt.savefig('Log_ROC')
   plt.show()
   new_row = {'Model Name' : models,
           'True Positive' : tp,
           'False Negative' : fn,
           'False Positive' : fp,
           'True Negative' : tn,
           'Accuracy' : accuracy,
           'Precision' : precision,
           'Recall' : sensitivity,
           'F1 Score' : f1Score,
           'Specificity' : specificity,
           'MCC':MCC,
           'ROC_AUC_Score':roc_auc_score(y_test, y_pred),
           'Balanced Accuracy':balanced_accuracy}
   CSResults = CSResults.append(new_row, ignore_index=True)
```

7. Gaussian NB

```
from sklearn.naive_bayes import GaussianNB
models = GaussianNB()
#Fit the model
models.fit(x_train, y_train)
```

```
 y_pred = models.predict(x_test)
 y_pred_prob = models.predict_proba(x_test)
 print('Model Name: ', models)
 # confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
 print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
  balanced_accuracy = round((sensitivity+specificity)/2, 3);
  precision = round(tp/(tp+fp), 3);
  f1Score = round((2*tp/(2*tp + fp + fn)), 3);
  # Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
  # A model with a score of +1 is a perfect model and -1 is a poor model
  from math import sqrt
  mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
  MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
  print('Accuracy :', round(accuracy*100, 2),'%')
  print('Precision :', round(precision*100, 2),'%')
  print('Recall :', round(sensitivity*100,2), '%')
  print('F1 Score :', f1Score)
  print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
  print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
  print('MCC :', MCC)
  # Area under ROC curve
  from sklearn.metrics import roc_curve, roc_auc_score
  print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
   # ROC Curve
   from sklearn.metrics import roc_auc_score
   from sklearn.metrics import roc_curve
   logit_roc_auc = roc_auc_score(actual, predicted)
   fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
   plt.figure()
   # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
   plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
   plt.plot([0, 1], [0, 1],'r--')
   plt.xlim([0.0, 1.0])
   plt.ylim([0.0, 1.05])
   plt.xlabel('False Positive Rate')
   plt.ylabel('True Positive Rate')
   plt.title('Receiver operating characteristic')
   plt.legend(loc="lower right")
   plt.savefig('Log_ROC')
   plt.show()
   new_row = {'Model Name' : models,
             'True Positive' : tp,
             'False Negative' : fn,
             'False Positive' : fp,
             'True Negative' : tn,
             'Accuracy' : accuracy,
             'Precision' : precision,
             'Recall' : sensitivity,
             'F1 Score' : f1Score,
             'Specificity' : specificity,
             'MCC':MCC,
             'ROC_AUC_Score':roc_auc_score(y_test, y_pred),
             'Balanced Accuracy':balanced_accuracy}
```

```
  CSResults = CSResults.append(new_row, ignore_index=True)
```

8. Bagging Classifier

```
from sklearn.ensemble import BaggingClassifier
models=BaggingClassifier(base_estimator=None,n_estimators=100,max_samples=1.0,
max_features=1.0,bootstrap=True,bootstrap_features=False,oob_score=False,warm_start=False,n_jobs=Non
e, random_state=None, verbose=0)
#Fit the model
models.fit(x_train, y_train)
 y_pred = models.predict(x_test)
 y_pred_prob = models.predict_proba(x_test)
 print('Model Name: ', models)
 # confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
 print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
  balanced_accuracy = round((sensitivity+specificity)/2, 3);
  precision = round(tp/(tp+fp), 3);
  f1Score = round((2*tp/(2*tp + fp + fn)), 3);
  # Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
  # A model with a score of +1 is a perfect model and -1 is a poor model
```

```
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
 # ROC Curve
 from sklearn.metrics import roc_auc_score
 from sklearn.metrics import roc_curve
 logit_roc_auc = roc_auc_score(actual, predicted)
 fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
 plt.figure()
 # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
 plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
 plt.plot([0, 1], [0, 1],'r--')
 plt.xlim([0.0, 1.0])
 plt.ylim([0.0, 1.05])
 plt.xlabel('False Positive Rate')
 plt.ylabel('True Positive Rate')
 plt.title('Receiver operating characteristic')
 plt.legend(loc="lower right")
 plt.savefig('Log_ROC')
 plt.show()
 new_row = {'Model Name' : models,
           'True Positive' : tp,
           'False Negative' : fn,
           'False Positive' : fp,
           'True Negative' : tn,
           'Accuracy' : accuracy,
           'Precision' : precision,
           'Recall' : sensitivity,
           'F1 Score' : f1Score,
           'Specificity' : specificity,
           'MCC':MCC,
           'ROC_AUC_Score':roc_auc_score(y_test, y_pred),
           'Balanced Accuracy':balanced_accuracy}
 CSResults = CSResults.append(new_row, ignore_index=True)
```

9. GradientBoostingClassifier

```
from sklearn.ensemble import GradientBoostingClassifier
models=GradientBoostingClassifier(loss='deviance',learning_rate=0.1,n_estimators=100,
subsample=1.0,criterion='friedman_mse',min_samples_split=2,min_samples_leaf=1,
min_weight_fraction_leaf=0.0,max_depth=3,min_impurity_decrease=0.0,min_impurity_split=None,init=No
ne,random_state=None,max_features=None,verbose=0,max_leaf_nodes=None,warm_start=False,validation_
fraction=0.1,n_iter_no_change=None,tol=0.0001,ccp_alpha=0.0)
#Fit the model
models.fit(x_train, y_train)
 y_pred = models.predict(x_test)
 y_pred_prob = models.predict_proba(x_test)
 print('Model Name: ', models)
 # confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
 print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
```

```
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
 balanced_accuracy = round((sensitivity+specificity)/2, 3);
 precision = round(tp/(tp+fp), 3);
 f1Score = round((2*tp/(2*tp + fp + fn)), 3);
 # Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
 # A model with a score of +1 is a perfect model and -1 is a poor model
 from math import sqrt
 mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
 MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
 print('Accuracy :', round(accuracy*100, 2),'%')
 print('Precision :', round(precision*100, 2),'%')
 print('Recall :', round(sensitivity*100,2), '%')
 print('F1 Score :', f1Score)
 print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
 print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
 print('MCC :', MCC)
 # Area under ROC curve
 from sklearn.metrics import roc_curve, roc_auc_score
 print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
  # ROC Curve
  from sklearn.metrics import roc_auc_score
  from sklearn.metrics import roc_curve
  logit_roc_auc = roc_auc_score(actual, predicted)
  fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
  plt.figure()
  # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
  plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
  plt.plot([0, 1], [0, 1],'r--')
  plt.xlim([0.0, 1.0])
  plt.ylim([0.0, 1.05])
  plt.xlabel('False Positive Rate')
  plt.ylabel('True Positive Rate')
  plt.title('Receiver operating characteristic')
  plt.legend(loc="lower right")
  plt.savefig('Log_ROC')
  plt.show()
  new_row = {'Model Name' : models,
            'True Positive' : tp,
            'False Negative' : fn,
            'False Positive' : fp,
            'True Negative' : tn,
            'Accuracy' : accuracy,
            'Precision' : precision,
            'Recall' : sensitivity,
            'F1 Score' : f1Score,
            'Specificity' : specificity,
            'MCC':MCC,
            'ROC_AUC_Score':roc_auc_score(y_test, y_pred),
            'Balanced Accuracy':balanced_accuracy}
  CSResults = CSResults.append(new_row, ignore_index=True)
```

10. LightGBM

```
import lightgbm as lgb
ModelLGB = lgb.LGBMClassifier()
#Fit the model
models.fit(x_train, y_train)
 y_pred = models.predict(x_test)
 y_pred_prob = models.predict_proba(x_test)
 print('Model Name: ', models)
 # confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
 # confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
```

```
 print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
  balanced_accuracy = round((sensitivity+specificity)/2, 3);
  precision = round(tp/(tp+fp), 3);
  f1Score = round((2*tp/(2*tp + fp + fn)), 3);
  # Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
  # A model with a score of +1 is a perfect model and -1 is a poor model
  from math import sqrt
  mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
  MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
  print('Accuracy :', round(accuracy*100, 2),'%')
  print('Precision :', round(precision*100, 2),'%')
  print('Recall :', round(sensitivity*100,2), '%')
  print('F1 Score :', f1Score)
  print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
  print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
  print('MCC :', MCC)
  # Area under ROC curve
  from sklearn.metrics import roc_curve, roc_auc_score
  print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
   # ROC Curve
   from sklearn.metrics import roc_auc_score
   from sklearn.metrics import roc_curve
   logit_roc_auc = roc_auc_score(actual, predicted)
   fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
   plt.figure()
```

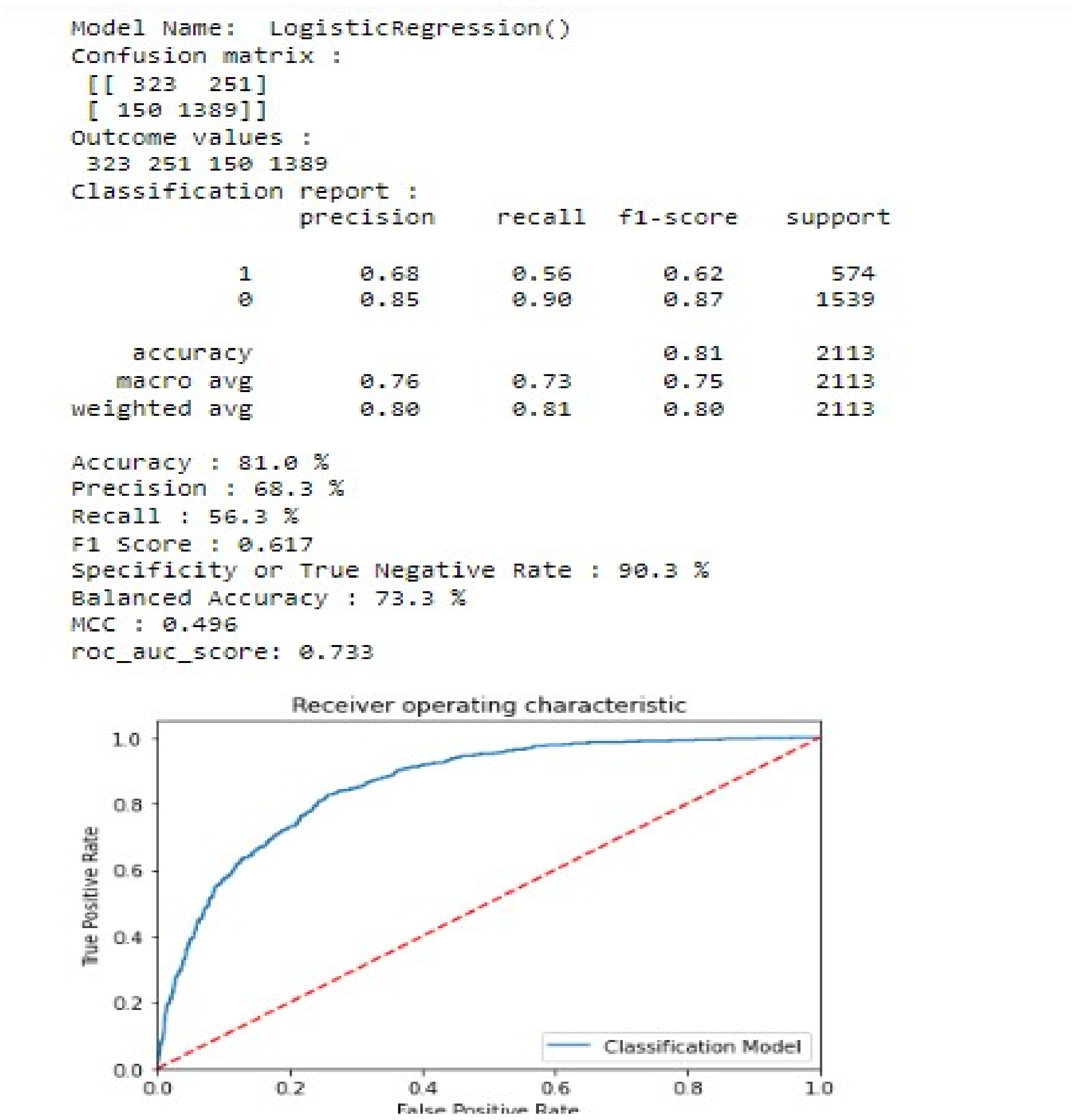
```
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
new_row = {'Model Name' : models,
        'True Positive' : tp,
        'False Negative' : fn,
        'False Positive' : fp,
        'True Negative' : tn,
        'Accuracy' : accuracy,
        'Precision' : precision,
        'Recall' : sensitivity,
        'F1 Score' : f1Score,
        'Specificity' : specificity,
        'MCC':MCC,
        'ROC_AUC_Score':roc_auc_score(y_test, y_pred),
        'Balanced Accuracy':balanced_accuracy}
CSResults = CSResults.append(new_row, ignore_index=True)
```

4.0 Conclusions and Future Work

The model results in the following order by considering the model accuracy, F1 score and ROC AUC score.

1. Logistic Regression with simple random sampling
2. Light GBM Classifier with simple random sampling
3. Support Vector Machines with simple random sampling

We recommend model - Logistic Regression with Random Sampling technique as a best fit for the given Telco Customer Churn data set. We considered Logistic Regression because it uses bootstrap aggregation which can reduce bias and variance in the data and can leads to good predictions with claims data set.



```
Model Name:  LogisticRegression()
Confusion matrix :
 [[ 323  251]
 [ 150 1389]]
Outcome values :
 323 251 150 1389
Classification report :
               precision    recall  f1-score   support

           1       0.68      0.56      0.62       574
           0       0.85      0.90      0.87      1539

    accuracy                           0.81      2113
   macro avg       0.76      0.73      0.75      2113
weighted avg       0.80      0.81      0.80      2113

Accuracy : 81.0 %
Precision : 68.3 %
Recall : 56.3 %
F1 Score : 0.617
Specificity or True Negative Rate : 90.3 %
Balanced Accuracy : 73.3 %
MCC : 0.496
roc_auc_score: 0.733
```

5.0 References

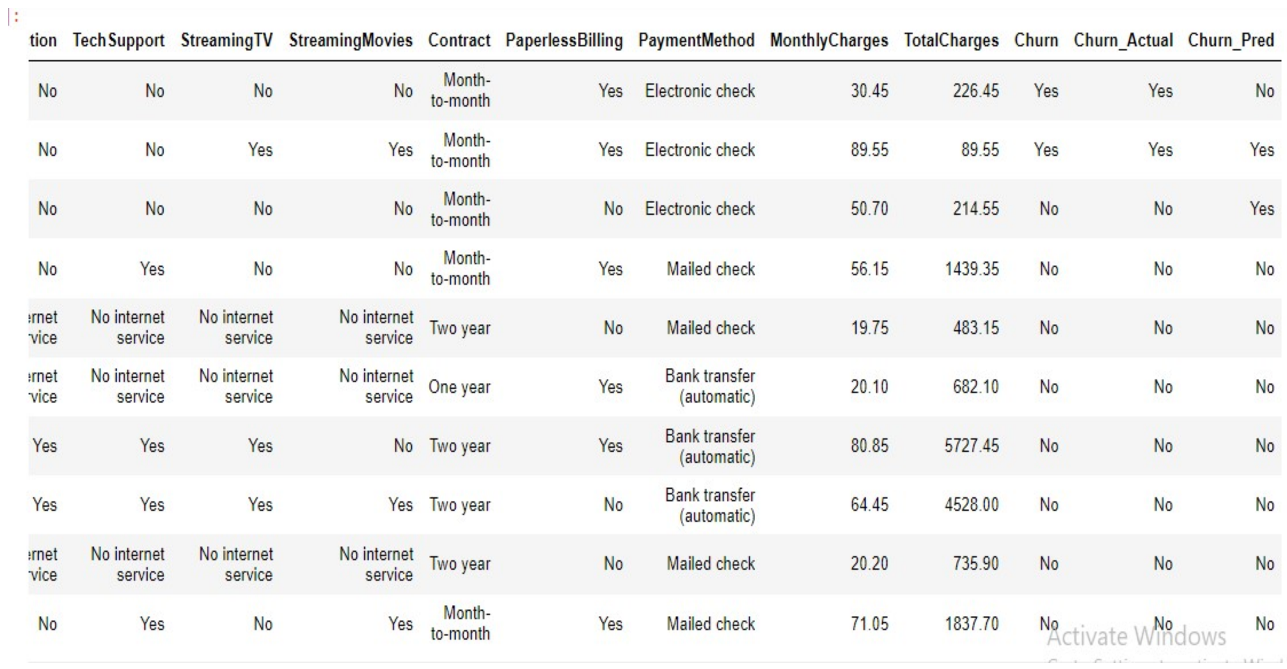
- https://www.kaggle.com/
- https://scikit-learn.org/stable/
- https://towardsdatascience.com/
- https://www.learndatasci.com/
- https://en.wikipedia.org/

6.0 Appendixes

- Network algorithms
- Computational Social Science
- Natural Language Processing Recent Trends in deep learning and representation learning
- Kernel Methods
- Graphical Methods
- Reinforcement Learning
- Convex Analysis
- Facial Recognition
- Virtual Personal Assistants
- Optimization
- Decision Theory
- Topics in Information Technology

6.1 Python code Results

	Model Name	True Positive	False Negative	False Positive	True Negative	Accuracy	Precision	Recall	F1 Score	Specificity	MCC	ROC_AUC_Score
0	LogisticRegression()	323	251	150	1389	0.810	0.683	0.563	0.617	0.903	0.496	0.732626
1	DecisionTreeClassifier()	279	295	281	1258	0.727	0.498	0.486	0.492	0.817	0.306	0.651738
2	(DecisionTreeClassifier(max_features='auto', r...	275	299	128	1411	0.798	0.682	0.479	0.563	0.917	0.448	0.697962
3	(ExtraTreeClassifier(random_state=1636317368),...	262	312	162	1377	0.776	0.618	0.456	0.525	0.895	0.390	0.675591
4	KNeighborsClassifier()	281	293	227	1312	0.754	0.553	0.490	0.519	0.853	0.356	0.671024
5	SVC(probability=True)	276	298	125	1414	0.800	0.688	0.481	0.566	0.919	0.453	0.699807
6	(DecisionTreeClassifier(random_state=174380082...	273	301	153	1386	0.785	0.641	0.476	0.546	0.901	0.417	0.688097
7	([DecisionTreeRegressor(criterion='friedman_ms...	295	279	147	1392	0.798	0.667	0.514	0.581	0.904	0.458	0.709210
8	LGBMClassifier()	302	272	149	1390	0.801	0.670	0.526	0.589	0.903	0.466	0.714658
9	GaussianNB()	439	135	379	1160	0.757	0.537	0.765	0.631	0.754	0.474	0.759272

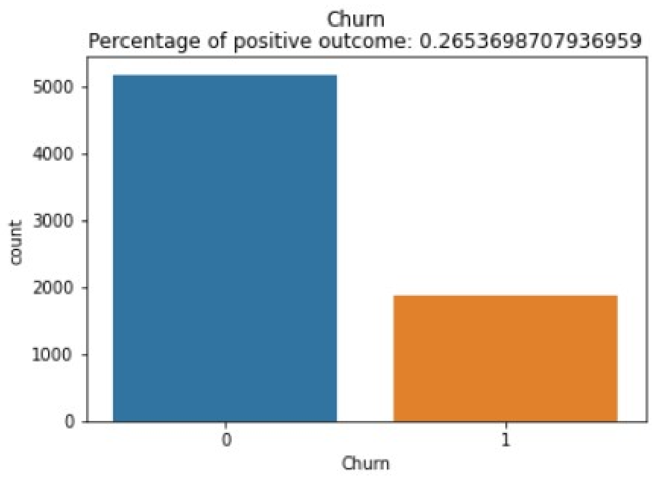


tion	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn	Churn_Actual	Churn_Pred
No	No	No	No	Month-to-month	Yes	Electronic check	30.45	226.45	Yes	Yes	No
No	No	Yes	Yes	Month-to-month	Yes	Electronic check	89.55	89.55	Yes	Yes	Yes
No	No	No	No	Month-to-month	No	Electronic check	50.70	214.55	No	No	Yes
No	Yes	No	No	Month-to-month	Yes	Mailed check	56.15	1439.35	No	No	No
rnet vice	No internet service	No internet service	No internet service	Two year	No	Mailed check	19.75	483.15	No	No	No
rnet vice	No internet service	No internet service	No internet service	One year	Yes	Bank transfer (automatic)	20.10	682.10	No	No	No
Yes	Yes	Yes	No	Two year	Yes	Bank transfer (automatic)	80.85	5727.45	No	No	No
Yes	Yes	Yes	Yes	Two year	No	Bank transfer (automatic)	64.45	4528.00	No	No	No
rnet vice	No internet service	No internet service	No internet service	Two year	No	Mailed check	20.20	735.90	No	No	No
No	Yes	No	Yes	Month-to-month	Yes	Mailed check	71.05	1837.70	No	No	No

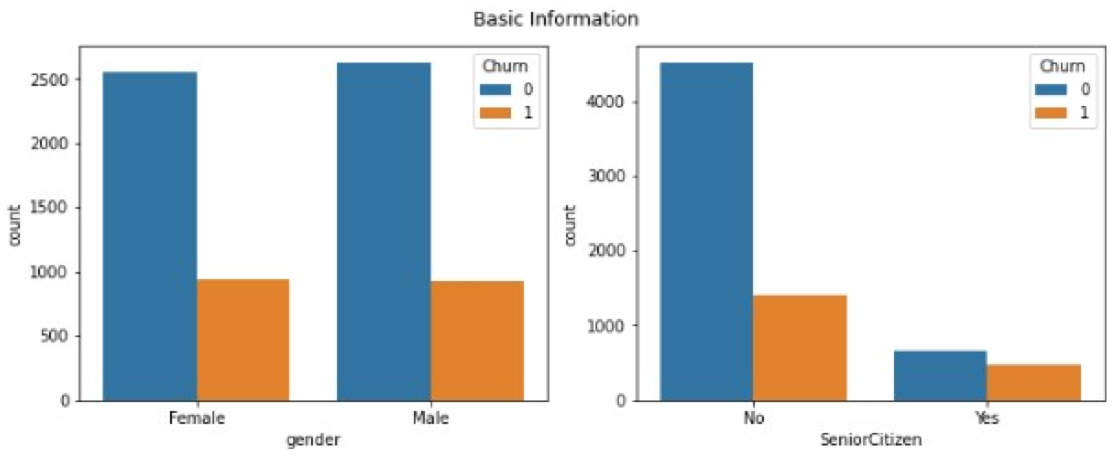
Activate Windows

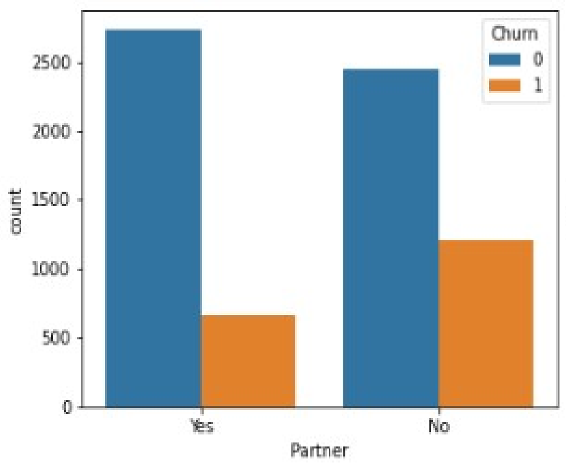
6.2 List of Charts

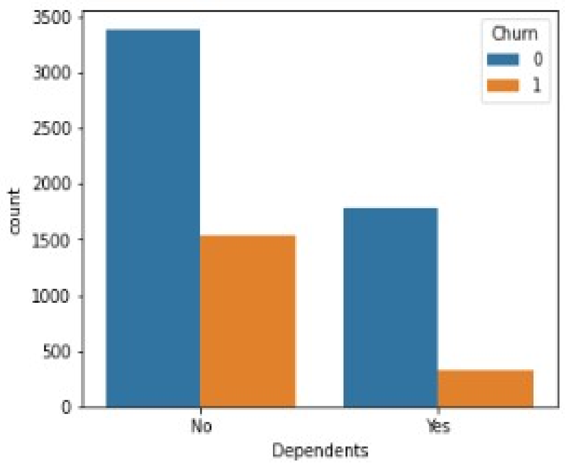
6.2.1 Chart 01: Count of Churn data



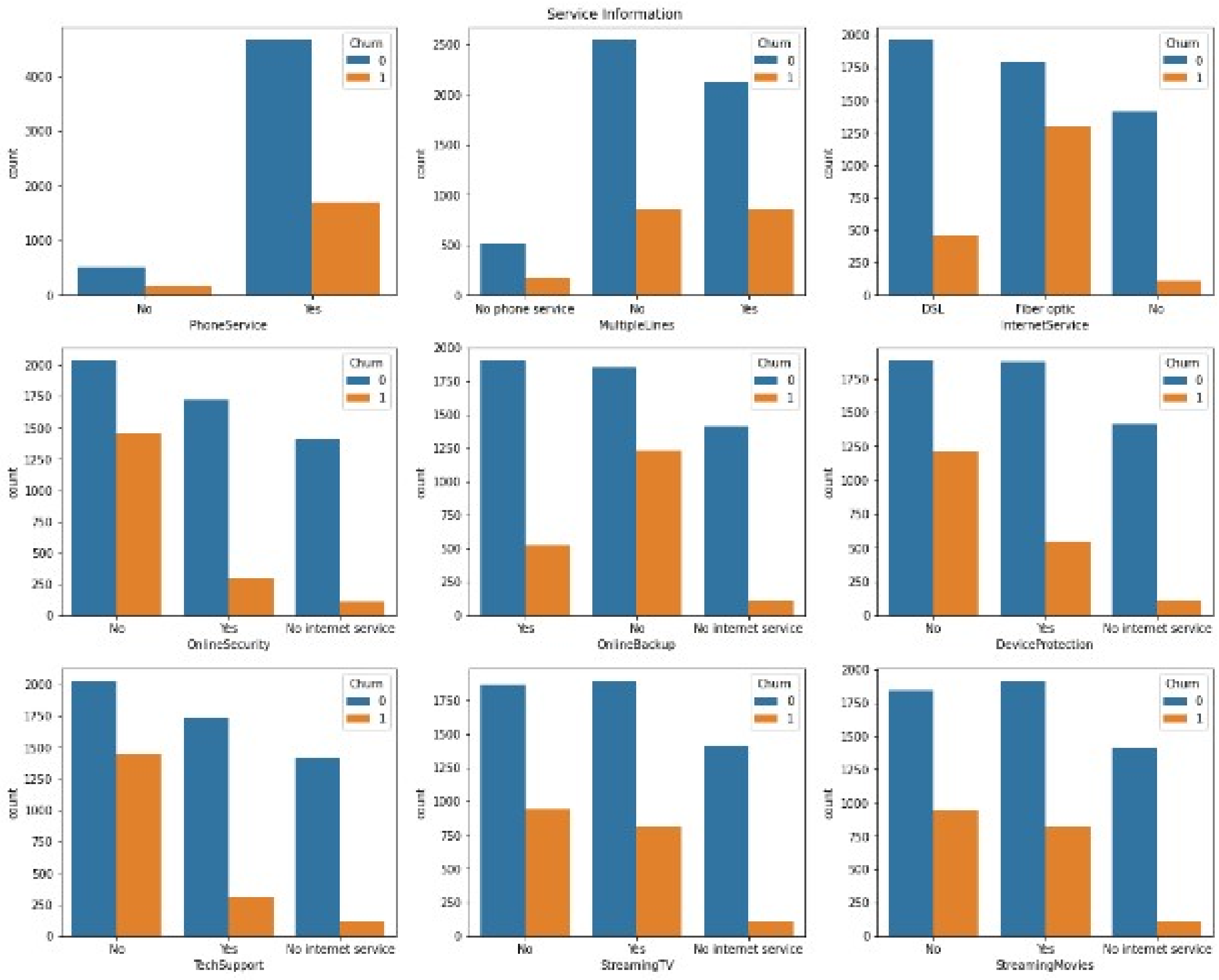
6.2.2 Chart 02: Count of churn data column wise in basic information



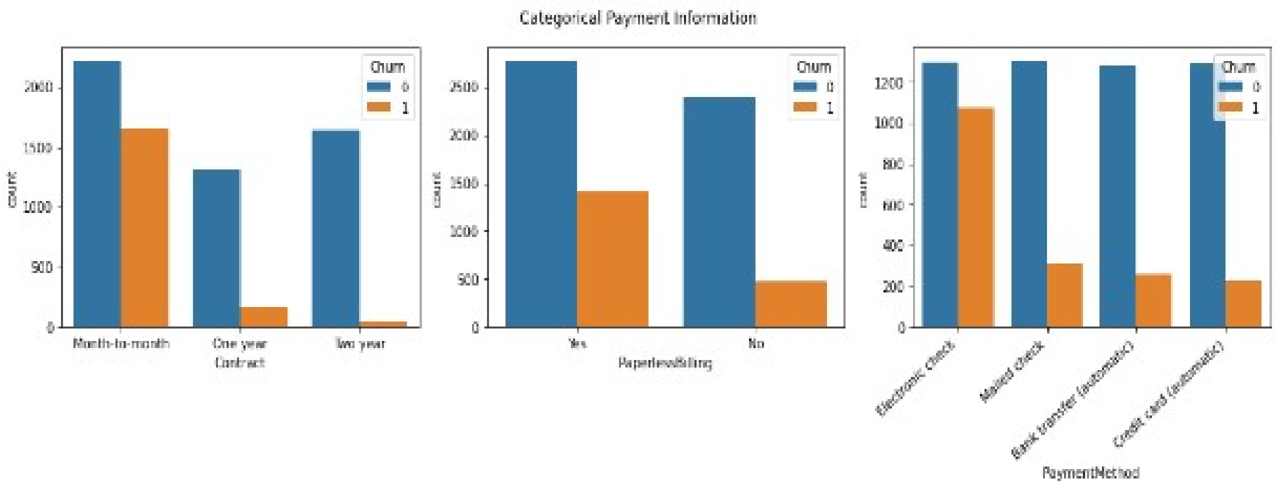




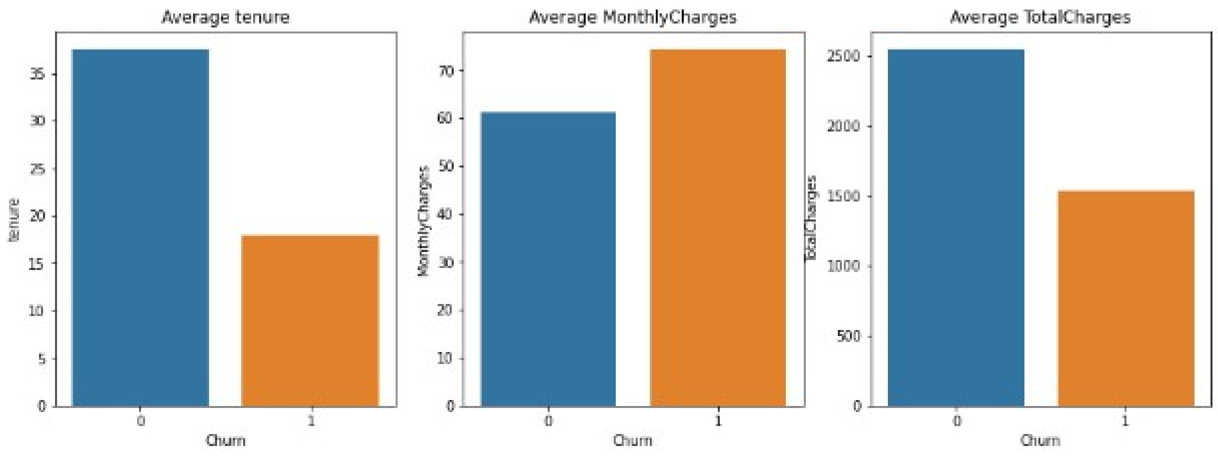
6.2.3 Chart 03: Count of Churn in different service providing attributes



6.2.4 Chart 04:Count of Churn in different Categorical Payment Information



6.2.5 Chart 05:Count of Churn in Different charges


Average tenure
tenure
Average MonthlyCharges
MonthlyCharges
Average TotalCharges
TotalCharges
Churn