

A Project Report on

“Customer Attrition Prediction System”

**Submitted in partial fulfillment of the requirement for
Degree in Bachelor of Engineering (Information Technology)**

By

**Pratik Wilson Bhosale (5019106)
Saurabh Ramchandra Phadke (5019146)
Sam Thomas Valumannil (5019164)**

Guided by:

Prof. Anandkumar Pardeshi



**Department of Information Technology
Fr. Conceicao Rodrigues Institute of Technology
Sector 9A, Vashi, Navi Mumbai – 400703**

**University of Mumbai
2021-2022**

CERTIFICATE

This is to certify that the project entitled

“Customer Attrition Prediction System”

Submitted By

Pratik Wilson Bhosale
Saurabh Ramchandra Phadke
Sam Thomas Valumannil

In partial fulfillment of degree of **B.E. in Information Technology** for term work of
Semester V - Mini Project 1B is approved.

External Examiner

Internal Examiner

External Guide

Internal Guide

Head of the Department

Principal

Date: -

College Seal

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

Pratik Wilson Bhosale (5019106)

(Name of student and Roll No.)

Saurabh Ramchandra Phadke (5019146)

(Name of student and Roll No.)

Sam Thomas Valumannil (5019164)

(Name of student and Roll No.)

Date:

Abstract

Customer churn decreases the value gained by companies from customers. If an enterprise is not able to tackle the increasing customer churn, it will gradually lose its competitive edge. And if the growth of new customers cannot meet the needs of development, the enterprise will fall into a survival dilemma. Focusing on the customer churn prediction model, this project takes the telecom industry as the research object and establishes a customer churn prediction model based on a dataset of high-value customer operation in the telecom industry, effectively identifying the potential churned customers along with plausible reasons for attrition. This project analyzes the trends and causes of customer churn through supervised machine learning algorithms and gives the answers to such questions as how the customer churn occurs and the influencing factors of customer churn.

1. The data set has 7043 rows and consists of 21 columns/attributes including Demographic information about customers including gender, age, marital status
2. Customer account information including the number of months staying with the company, paperless billing, payment method, monthly charges, and total charges
3. Customer consumption behaviour like streaming TV or streaming movies, etc
4. Services that the customer signed up for: phone service, multiples, internet service, online security, online backup, device protection, and tech support
5. Customer churn where the customer left within the last month.

Most of the features such as gender, phone service, all the way up to payment method were all categorical data whereas Monthly Charges and Total Charges are both numerical data. This dataset is then used with various Supervised Machine Learning models like Linear Regression, Random Forest, Naïve Bayes Classifier, Decision Tree, Support Vector Machine, etc and their accuracy is compared to select the best model. This project can help improve existing customer relationship management in the telecom industry and provide a reference for the telecom industry to identify high-risk churned customers in advance, enhance customer loyalty and viscosity, maintain “high-value” customers, and continue to provide customers with “value” and reduce the cost of maintaining customers.

The model has been deployed to a web app created using Streamlit for user feasibility and interactive interaction even by untrained personnel. The app provides an attractive and user-friendly interface with 2 modes of prediction.

1. **Online Mode:** In this mode, single customer prediction is provided. The user needs to input the features of the customer using the interactive options provided and the model will predict the attrition of that customer.
2. **Batch Mode:** Batch Mode predicts the prediction for multiple users. The data of multiple customers can be directly uploaded in the form of a csv file and the model will predict the attrition for each of the customers. This makes the app more easy to use even when more data needs to be predicted. This provides extensible application in real life scenarios.

INDEX

Sr. No.	Topic	Page No.
1	Introduction <ul style="list-style-type: none">- Back ground- Motivation- Problem Definition- Scope / Assumptions- Issues / Limitations	1
2	Literature Survey and Analysis <ul style="list-style-type: none">- Related work- Existing System- Requirement Analysis- Proposed System	3
3	System Design <ul style="list-style-type: none">- Architectural Diagram/ block diagram- Flowchart	10
4	Implementation Details <ul style="list-style-type: none">- System Requirements- Solution Approach- Model Used	14
5	Experimental Results <ul style="list-style-type: none">- GUI- Final Result	17
6	Conclusion and Future Scope <ul style="list-style-type: none">- Conclusion- Future Scope	22
7	References	24
8	Appendix A: Code Sample	26
9	Acknowledgements	51

LIST OF FIGURES

Sr. No.	Name of the Figure	Page No.
3.1	Architectural Design	11
3.2	System Flow	13
5.1	Homepage	18
5.2	Online mode (1)	18
5.3	Online mode (2)	19
5.4	Attrition prediction in online mode -Yes	19
5.5	Attrition prediction in online mode - No	20
5.6	Batch mode	20
5.7	Batch mode – import CSV file	21
5.8	Attrition prediction in batch mode	21

LIST OF TABLES

Sr. No.	Name of the Table	Page No.
2.1	Comparison of Related Works	5

Chapter 1

Introduction

The recent advancements and globalization of telecommunication sector has exponentially raised the number of operators in the market that has escalated the competition in this industry. In this competitive era, it has become mandatory to maximize the profits periodically, for this various strategies have been proposed, namely, acquiring new customers, up-selling the existing customers & increasing the retention period of existing customers.

Among all the strategies, the least expensive strategy is retention of existing customers. In order to adopt the third strategy, companies have to reduce the potential customer attrition i.e., customer movement from one service provider to another.

One of the main reasons for customers leaving a telecom company is their dissatisfaction with after sales support and customer service. The key to unlock solutions to this problem is by forecasting the customers which are at risk of attrition. One of the main aims of Customer Churn prediction is to help in establishing strategies for customer retention. Along with growing competition in markets for providing services, the risk of customer churn also increases exponentially. Therefore, establishing strategies to keep track of loyal customers has become a necessity. The customer attrition models aim to identify early attrition signals and try to predict the customers that leave voluntarily. Thus, many companies have realized that their existing database is one of their most valuable asset.

1.1 Problem Definition:

- To analyze attributes of dataset to help companies understand their customer base, preferences, strong and weak points, etc
- To predict Customer Attrition using Machine Learning techniques thus helping companies retain their customer base and survive in the market.
- To deploy the model so as to provide feasibility and user-friendliness so that the model can be used even by untrained personnel for attrition prediction.

1.2 Scope:

- To predict Customer Churn.
- Highlighting the main variables/factors influencing Customer Churn.
- Use various ML algorithms to build prediction models, evaluate the accuracy and performance of these models.
- Finding out the best model for our business case & providing executive summary.
- Deploying the model to make customer prediction feasible and user friendly.

1.3 Limitations:

- Based on the data provided, any false information in data can make model give inaccurate predictions.
- Predictive Models cannot comprehend features like human emotions.
- Accuracy can be improved.

Chapter 2

Literature Survey and Analysis

2.1 Related work:

1. Customer churn prediction system: a machine learning approach [1]:

The paper discusses Customer Attrition Prediction in Telecom Industry. The methodology of this project consists of six phases. In the first two phases, data pre-processing and feature analysis is performed. In the third phase, feature selection is taken into consideration using gravitational search algorithm. Next, the data has been split into two parts train and test set in the ratio of 80% and 20% respectively. In the prediction process, most popular predictive models have been applied, namely, logistic regression, naive bayes, support vector machine, random forest, decision trees, etc. on train set as well as boosting and ensemble techniques are applied to see the effect on accuracy of models. In addition, K-fold cross validation has been used over train set for hyperparameter tuning and to prevent overfitting of models. Finally, the obtained results on test set have been evaluated using confusion matrix and AUC curve. It was found that Adaboost and XGboost Classifier gives the highest accuracy of 81.71% and 80.8% respectively. The highest AUC score is achieved by both Adaboost and XGBoost Classifiers which outperforms others.

2. Telecom Churn Prediction System Based on Ensemble Learning Using Feature Grouping [2]:

This study proposes a customer-churn prediction system that uses an ensemble-learning technique consisting of stacking models and soft voting. This study proposes a new customer-churn prediction system and feature construction to improve accuracy, and the contributions of this study can be summarized as follows:

- (1) A new prediction system based on ensemble learning with relatively high accuracy is proposed.
- (2) New features derived from equidistant grouping of customer behavior features are used to improve the system performance.

Xgboost, Logistic regression, Decision tree, and Naïve Bayes machine-learning algorithms are selected to build a stacking model with two levels, and the three outputs of the second level are used for soft voting. Feature construction of the churn dataset includes equidistant grouping of customer behavior features to expand the space of features and discover latent information from the churn dataset. The original and new churn datasets are analyzed in the stacking ensemble model with four evaluation metrics. The experimental results show that the proposed customer churn predictions have accuracies of 96.12% and 98.09% for the original and new churn datasets, respectively. These results are better than state-of-the-art churn recognition systems.

3. Analysis of Customer Churn Prediction in Telecom Industry Using Decision Trees and Logistic Regression[3]:

In the project, R programming will be used to build the model for churn prediction. This project mainly focuses on tree-based and regression based machine learning methods and algorithms for prediction of churn in telecom industries. The project mainly focuses on Decision tree algorithm and logistic regression for efficient prediction model for customer churn. The system will have three main options namely “View performance analysis” which displays the results obtained by applying logistic regression and decision tree on the available dataset, “Testing” to construct a list of customers which have a high probability to churn from the input, given that the attributes of the input data are same as the available dataset used for training, “Training” and testing which builds a model along with generating a churn list if any other type of dataset is provided in performance analysis the results after using logistic regression and decision trees on the available dataset is illustrated using confusion matrix analysis.

4. Customer churn prediction in telecom using machine learning in big data platform[4]:

The main contribution of this project is to develop a churn prediction model which assists telecom operators to predict customers who are most likely subject to churn. The model developed in this work uses machine learning techniques on big data platform and builds a new way of features’ engineering and selection. In order to measure the performance of the model, the Area Under Curve (AUC) standard measure is adopted, and the AUC value obtained is 93.3%. Another main contribution is to use customer social network in the prediction model by extracting Social Network Analysis (SNA) features. The dataset contained all customers’ information over 9 months, and was used to train, test, and evaluate the system at SyriaTel. The model experimented four algorithms: Decision Tree, Random Forest, Gradient Boosted Machine Tree “GBM” and Extreme Gradient Boosting “XGBOOST”. However, the best results were obtained by applying XGBOOST algorithm.

Sr. No.	Title of Paper	Year of Publication	Authors	Features
1	Customer churn prediction system: a machine learning approach	January 2021	Praveen Lalwani, Manas Kumar Mishra, Jasroop Singh Chadha, Pratyush Sethi	<ol style="list-style-type: none">1. It consists of six phases.2. Most popular predictive models have been applied, namely, logistic regression, naive bayes, support vector machine, random forest, decision trees, etc. on train sets.3. Boosting and

				<p>ensemble techniques are applied to see the effect on accuracy of models.</p> <p>4. Adaboost and XGboost Classifier gives the highest accuracy of 81.71% and 80.8% respectively.</p>
2	Telecom Churn Prediction System Based on Ensemble Learning Using Feature Grouping	May 2021	Tianpei Xu, Ying Ma, Kangchul Kim	<p>1. The prediction system uses an ensemble-learning technique consisting of stacking models and soft voting.</p> <p>2. Xgboost, Logistic regression, Decision tree, and Naïve Bayes machine-learning algorithms are used.</p> <p>3. The experimental results show that the proposed customer churn predictions have accuracies of 96.12% and 98.09% for the original and new churn datasets, respectively.</p>
3	Analysis of Customer Churn Prediction in Telecom Industry Using Decision Trees and Logistic Regression	September 2016	Preeti K. Dalvi, Siddhi K. Khandge, Ashish Deomore, Aditya Bankar, V. A. Kanade	<p>1. R programming was used to build the model for churn prediction.</p> <p>2. Focus on Decision tree algorithm and logistic regression for efficient prediction model for customer churn.</p> <p>3. The system has three main options namely “View performance analysis”, “Testing” and “Training.”</p>

4	Customer churn prediction in telecom using machine learning in big data platform	March 2019	Abdelrahim Kasem Ahmad, Assef Jafar , Kadan Aljoumaa	<ol style="list-style-type: none"> 1. The model developed in this work uses machine learning techniques on big data platforms and builds a new way of features' engineering and selection. 2. The Area Under Curve (AUC) standard measure is adopted to measure the performance, and the AUC value obtained is 93.3%. 3. The model experimented with four algorithms: Decision Tree, Random Forest, Gradient Boosted Machine Tree "GBM" and Extreme Gradient Boosting "XGBOOST". 4. The best results were obtained by applying the XGBOOST algorithm.
---	--	------------	--	---

2.1 - Comparison of Related Works

2.2 Existing system:

1. A Customer Churn Prediction Model in Telecom Industry Using Boosting[5]:

With the rapid growth of digital systems and associated information technologies, there is an emerging trend in the global economy to build digital CRM systems. This trend is more obvious in the telecom industry, where companies become increasingly digitalized. Customer Churn Prediction is a main feature of modern telecom CRM systems. This research conducts a real-world study on customer churn prediction and proposes the use of boosting to enhance a customer churn prediction model. Unlike most research that uses boosting as a method to boost the accuracy of a given basis learner, this paper tries to separate customers into two clusters

based on the weight assigned by the boosting algorithm. As a result, a higher risk customer cluster has been identified. Logistic regression is used in this research as a basis learner, and a churn prediction model is built on each cluster respectively. The result is compared with a single logistic regression model. Experimental evaluation reveals that boosting also provides a good separation of churn data; thus, boosting is suggested for churn prediction analysis.

Dataset Used: Telecom data (2010)

Advantages: Accurate definition of high risk customer group

Disadvantages: Unable to finalize the reasons for customers churn

2. One-Class Support Vector Machine based undersampling: Application to Churn prediction and Insurance Fraud detection [6]:

In this paper, we propose One Class support vector machine (OCSVM) based undersampling. To demonstrate the effectiveness of the proposed methodology, we worked on Automobile Insurance fraud dataset and Credit card customer churn dataset taken from literature. We employed Decision Tree (DT), Support Vector Machine (SVM), Logistic Regression (LR), Probabilistic Neural Network (PNN) and Group Method of Data Handling (GMDH) for classification purpose. We observed significant improvement with respect to the Area Under Receiver Operating Characteristic Curve (AUC) over other techniques. For automobile insurance dataset, undersampling with the sigmoid kernel yielded AUC of 7605 when compared with Sundar kumar and Ravi, Vasu and Ravi, Farquad [9,12,47] with respect to Decision tree, while for Credit card customer churn dataset, undersampling with the radial basis kernel (proposed method) yielded significant performance with respect to DT (AUC 8506.5). We preferred DT over SVM (AUC 8728.5) as there is no statistically significant difference between them. Finally, we recommend DT over other classifiers as it also yields “if-then” rules, while achieving high AUC.

Dataset Used: Insurance dataset

Advantages: High accuracy.

Disadvantages: More applicable for fraud detection than churn prediction and increased system complexity.

3. Applying Bayesian Belief Network approach to customer churn analysis[7]:

In this study, a model is constructed by Bayesian Belief Network to identify the behaviors of customers with a propensity to churn. The data used are collected from one of the telecommunication providers in Turkey. First, as only discrete variables are used in Bayesian Belief Networks, CHAID (Chi-squared Automatic Interaction Detector) algorithm is applied to discretize continuous variables. Then, a causal map as a base of Bayesian Belief Network is brought out via the results of correlation analysis, multicollinearity test and experts' opinions. According to the results of Bayesian Belief Network, average minutes of calls, average billing

amount, the frequency of calls to people from different providers and tariff type are the most important variables that explain customer churn. At the end of the study, three different scenarios that examine the characteristics of the churners are analyzed and promotions are suggested to reduce the churn rate.

Dataset Used: Turkish Telecommunication data.

Advantages: Predicts the number of customers leaving and the factors very efficiently.

Disadvantages: Does not consider the relationship between the variables.

2.3 Requirement Analysis:

- Today's world revolves around data and companies have realized that to survive in the cut throat competition in the current industrial and commercial sectors, they need to be able to use this data to make progress and informed decisions.
- Predicting Customer attrition is the most viable method to retain customer base as it is more effective and at the same time cheaper as compared to other methods of surviving the competition like generating new customers, extensive but generalized marketing, etc.
- Making the prediction model easy to implement further reduces companies expenditure which would otherwise be required for imparting special training to the staff required for operating the model.

2.4 Proposed System:

This project proposes a simple and user-friendly Graphical User Interface (GUI) for predicting Customer Attrition in Telecom Industry to help in the overall Customer Relationship Management (CRM) process so as to help companies in the Telecom Sector to retain their Customer Base and work on their weaknesses by identifying them so as to survive in this highly competitive space.

The App is easy to operate and has two modes of operation:

- 1. Online Mode: Users need to input data for a single customer manually and prediction will be given as output.
- 2. Batch Mode: Users can input Comma Separated Values (CSV) file by either selecting the file in directory or through Drag and Drop which contains data of multiple customers and prediction for each customer will be given as output.

The Exploratory Data Analysis performed helps to understand customer behaviour and how churn is related to different factors/attributes. It helps to understand what are the weaknesses of the company and what services or factors lead to more customers leaving the company.

Using this analysis also helps companies formulate strategies for retaining customers.

Chapter 3

System Design

3.1 Architecture Diagram:

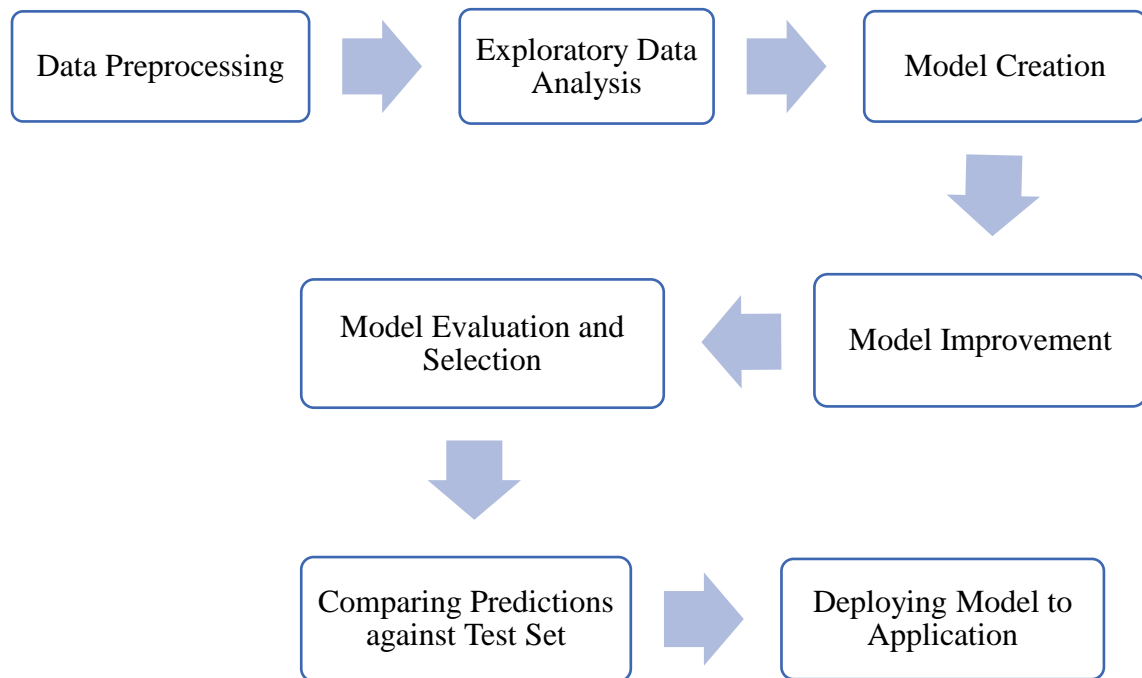


Fig 3.1 - Architectural Design

1.Data Preprocessing:

The telco company's data set is available on Kaggle, which stems from the IBM sample set collection. The company provides home and internet services to 7043 customers in California. Our challenge is to help the company predict behavior to retain customers and analyze all relevant customer data to develop focused customer retention programs.

The provided dataset consists of the information below:

1. Demographic information about customers including gender, age, marital status
2. Customer account information including the number of months staying with the company, paperless billing, payment method, monthly charges, and total charges
3. Customer usage behavior, such as streaming TV, streaming movie
4. Services that the customer signed up for: phone service, multiples, internet service, online security, online backup, device protection, and tech support
5. Customer churn where the customer left within the last month

Data Preprocessing involves the following steps:

1. Getting Familiar with the data and its structure.
2. Validate the Column Data Types.
3. Look for missing values.
4. Take care of missing data.
5. Identify unique values for the variables.
6. Check target variable distribution.
7. Label Encode Binary Data.

2. Exploratory Data Analysis:

EDA involves exploring and visualizing our data set by doing distribution of independent variables to better understand the patterns in the data and to potentially form some hypotheses.

It involves plotting graphs between different attributes so as to help find interesting patterns in the data that could help in discovering attributes that affect the target variable.

Performing EDA helps users better understand the data distribution. It enables users to analyze the factors due to which customers are leaving their company as well as help identify their weaknesses as well as strategize how these weaknesses can be overcome.

3. Model Creation:

This is the step where we apply various Machine learning algorithms on our data for creation of models.

It involves

1. **Splitting the dataset into dependent and independent variables:** Now we need to separate the dataset into X and y values. y would be the 'Churn' column whilst X would be the remaining list of independent variables in the dataset.
2. **Generate training and testing datasets:** Decouple the master dataset into training and testing set with an 80%-20% ratio.
3. **Applying various Machine Learning Algorithms:** like Logistic Regression, SVM, NB, K-Nearest Neighbours, Decision Tree, Random Forest, etc.
4. **Visualize Classification Algorithms Accuracy**

4. Model Improvement:

It involves methodologies adopted for trying to increase the accuracy of the classification algorithms.

For example:

- Finding Optimal number of K-Neighbours for KNN algorithm.
- Finding Optimal number of Trees for applying Random Forest Algorithm, etc.

5. Model Evaluation and Selection:

Calculate the Precision, Accuracy, Recall, etc for all the algorithms and compare the models based upon these values for selecting the model with the highest score for using in our model.

Then train the selected model and recheck accuracy using k-folds cross validation.

Further, create the confusion matrix and ROC graph for the selected model as well to validate accuracy, precision and fitness of the model.

6. Comparing Predictions against Test Set:

Apply the model on the Testing Set and plot Confusion Matrix in order to evaluate the accuracy of the model on Testing data. Testing data is independent of the training data so the accuracy obtained here is the accuracy the model gives on predicting unknown data i.e data on which the model is not trained.

7. Deploying Model to Application:

Save the model created and export it. Using this model create an application with a Graphical User Interface (GUI) to make using the model user-friendly as well as attractive and efficient.

This project deploys the model to a Web Application created using Streamlit and provides two modes for prediction:

1. Online Mode: For single prediction.
2. Batch Mode: For Batch predictions.

3.2 Flowchart:

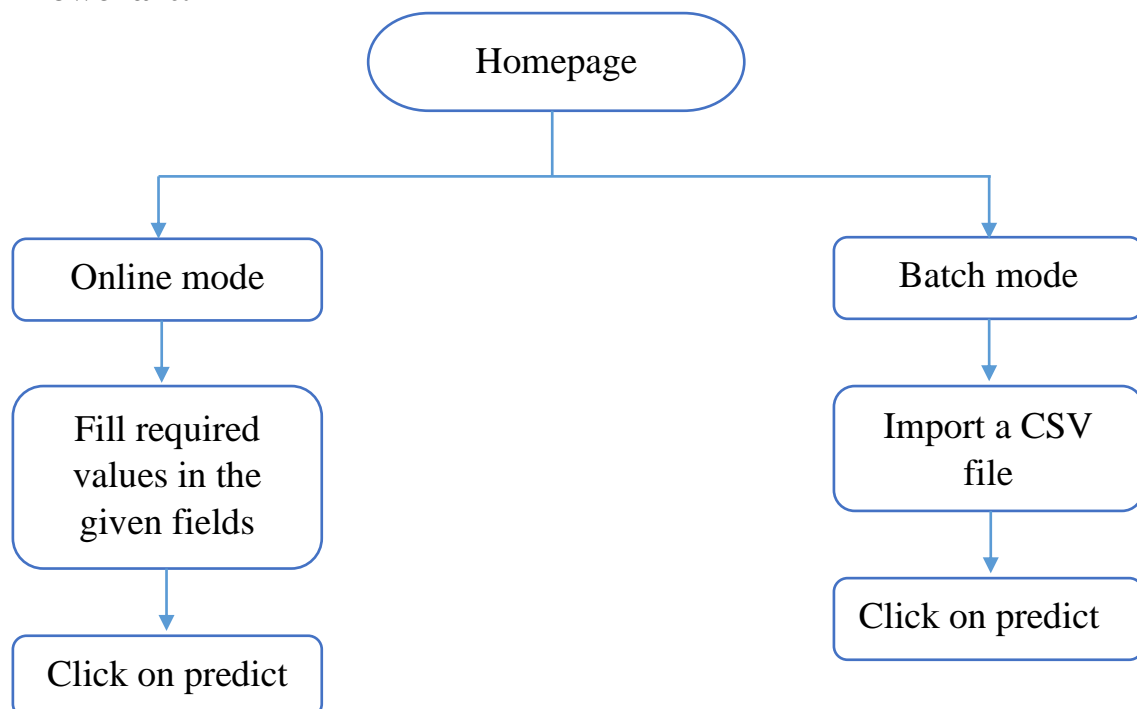


Fig 3.2: System flow

Chapter 4

Implementation Details

4.1 System Requirements:

Software Requirements:

1. Python
2. Streamlit
3. IDE (PyCharm and Jupyter Notebook)

Hardware Requirements:

1. RAM: 4 GB or more.
2. Processor: Intel i5 or equivalent.
3. Hard disk: Minimum 10 GB.

4.2 Solution Approach:

Our Prediction Model for Customer Attrition in Telecom Industry uses a dataset from Kaggle which consists of 7043 tuples and 21 attributes consisting of both Categorical and Numerical values. We make use of libraries like numpy and pandas to work with the dataset.

The dataset is checked for any missing values and the same are handled using methods like replacing with mean.

For categorical values containing 2 or less unique values we perform Label Encoding. Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important preprocessing step for the structured dataset in supervised learning.

Next we have performed exploratory data analysis to establish relations between the attributes to extract meaning from the data. For this process, libraries like matplotlib and scipy are used. We have plotted univariate distributions like Contract type to find what duration of plans are most popular with the customers. Similarly we have plotted various graphs to find information and develop knowledge about the relations of attributes and habits of customers. Bivariate graphs have also been plotted to check churn rates against specific attributes so as to try and conclude the reasons for churn. To better understand the reasons for attrition we have calculated and plotted correlations of attributes with churn. We have used different types of plots like Bar graphs, Stacked Graphs, Heat maps, Scatter plots with different various as required to better visualize the data to easily understand the relations.

After establishing relations, we move on to model creation. We have used multiple models like Logistic Regression, Support Vector Machine, Naïve Bayes, K-Nearest Neighbours, Decision Tree and Random Forest Classifier and then compared these models based on ROC AUC and Accuracy values.

AUC (Area Under The Curve) - ROC (Receiver Operating Characteristics) curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how

much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. We have also compared this visually using Boxplots. The Precision, Recall, F1 score and F2 score is also calculated for all the algorithms where,

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}$$

$$\text{F}\beta \text{ Score} = \frac{1 + \beta^2}{\frac{\beta^2}{\text{Recall}} + \frac{1}{\text{Precision}}}$$

The main advantage (and at the same time disadvantage) of the F1 score is that the recall and precision are of the same importance.

In many applications, this is not the case and some weight should be applied to break this balance assumption. For that we calculate F2, F3 or in general F β scores. A default beta value is 1.0, which is the same as the F-measure. A smaller beta value, such as 0.5, gives more weight to precision and less to recall, whereas a larger beta value, such as 2.0, gives less weight to precision and more weight to recall in the calculation of the score.

From Comparison on these values for all the algorithms we found that Logistic Regression performs the best. We have rechecked the accuracy of Logistic Regression using K-Folds Cross Validation and got results as 0.80 (+/- 0.04).

Next we created Confusion Matrix for the Model, Predicted the class labels on test Set And displayed the probability estimate of the predictions made.

And finally we deploy the model to a web app created using streamlit.

Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science.

We have also implemented batch processing wherein the user can directly enter a CSV file and the model will predict whether the customer will churn or not for each individual customer row.

4.3 Model Used:

Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. A Logistic Regression model is similar to a Linear Regression model, except that the Logistic Regression utilizes a more sophisticated cost function, which is known as the “Sigmoid function” or “logistic function” instead of a linear function. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on. Multinomial logistic regression can model scenarios where there are more than two possible discrete outcomes. Logistic regression is a useful analysis method for classification problems, where you are trying to determine a sample fits best into what category. The goal of Logistic Regression is to discover a link between characteristics and the likelihood of a specific outcome.

Chapter 5

Experimental Results

A Customer attrition prediction web app is successfully developed and has an attractive, elegant and easy to use User Interface. It provides all required functionalities to predict whether a customer will churn or not with an accuracy of 80.34%.

- It provides two options, an online mode and a batch mode
- The online mode asks user to fill in a set of fields in order to predict whether the customer will churn or not
- In the batch mode the user can directly import a csv file and the model will predict the churn for each individual customer.

The Exploratory Analysis helps companies analyse user preferences and behaviour and can be helping in formulating strategies for controlling Customer Attrition.

Web app GUI:

1. Homepage:

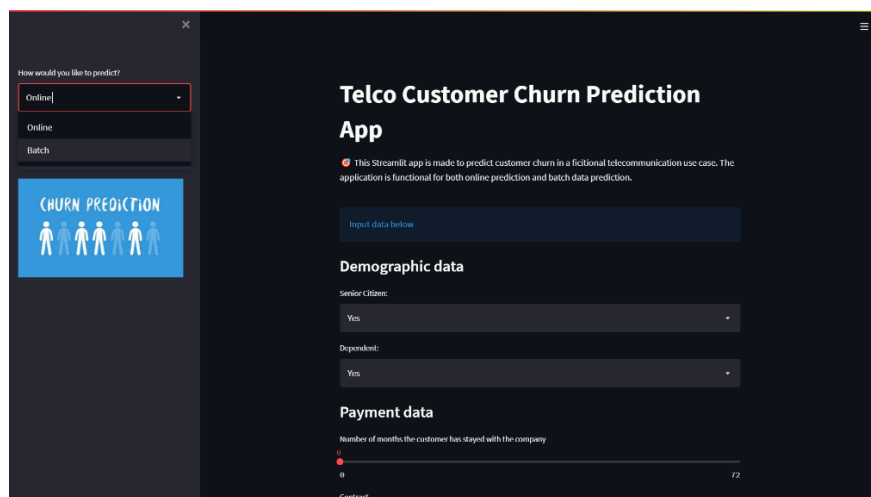


Fig 5.1 - Homepage

2. Online mode (1):

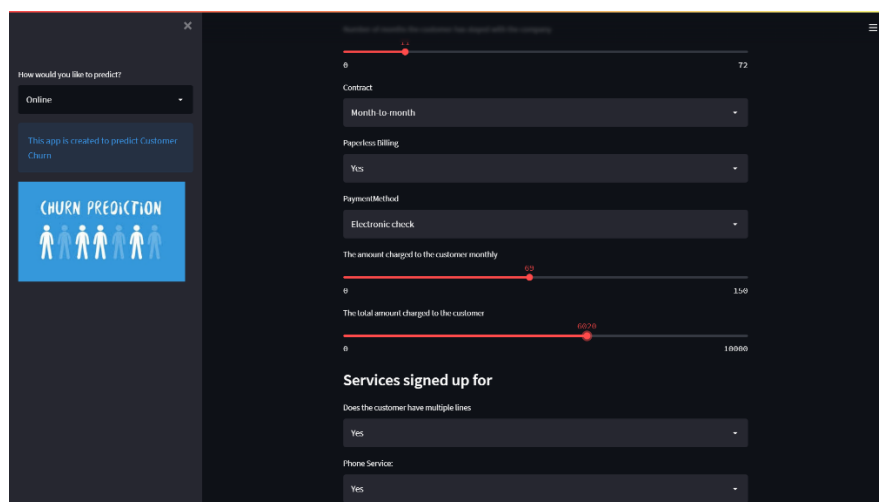


Fig 5.2 – Online mode (1)

3. Online mode (2):

How would you like to predict?

Online

This app is created to predict Customer Churn

CHURN PREDICTION

Does the customer have internet service

DSL

Does the customer have online security

Yes

Does the customer have online backup

Yes

Does the customer have technology support

Yes

Does the customer stream TV

Yes

Does the customer stream movies

Yes

Overview of input is shown below

	SeniorCitizen	Dependents	tenure	PhoneService	MultipleLines	InternetService
0	Yes	Yes	0	Yes	Yes	DSL

Predict

Fig 5.3 – Online mode (2)

4. Attrition prediction in online mode - Yes:

How would you like to predict?

Online

This app is created to predict Customer Churn

CHURN PREDICTION

Does the customer have internet service

DSL

Does the customer have online security

Yes

Does the customer have online backup

Yes

Does the customer have technology support

Yes

Does the customer stream TV

Yes

Does the customer stream movies

Yes

Overview of input is shown below

	SeniorCitizen	Dependents	tenure	PhoneService	MultipleLines	InternetService
0	Yes	Yes	0	Yes	Yes	DSL

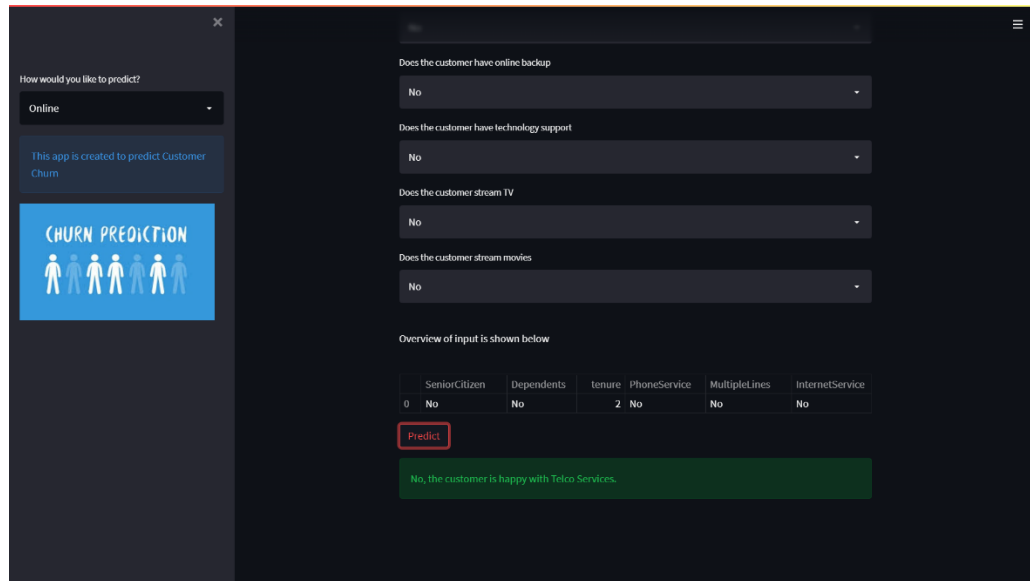
Predict

Yes, the customer will terminate the service.

Made with Streamlit

Fig 5.4 – Attrition prediction in online mode

5. Attrition prediction in online mode – No:



How would you like to predict?

Online

This app is created to predict Customer Churn

CHURN PREDICTION

Does the customer have online backup

No

Does the customer have technology support

No

Does the customer stream TV

No

Does the customer stream movies

No

Overview of input is shown below

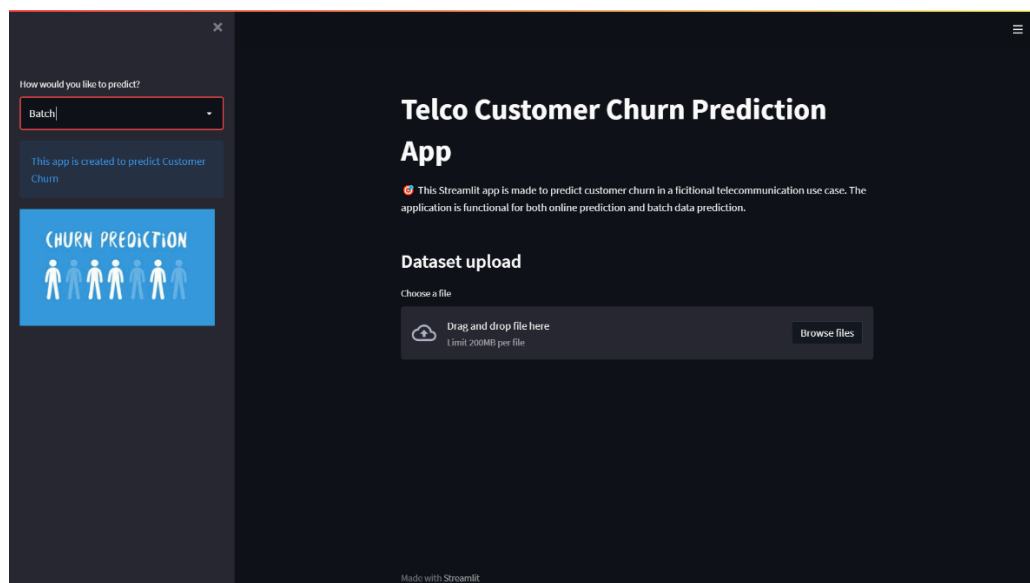
	SeniorCitizen	Dependents	tenure	PhoneService	MultipleLines	InternetService
0	No	No	2	No	No	No

Predict

No, the customer is happy with Telco Services.

Fig 5.5 - Attrition prediction in online mode – No

6. Batch mode:



How would you like to predict?

Batch

This app is created to predict Customer Churn

CHURN PREDICTION

Telco Customer Churn Prediction App

This Streamlit app is made to predict customer churn in a fictional telecommunication use case. The application is functional for both online prediction and batch data prediction.

Dataset upload

Choose a file

Drag and drop file here
Limit 200MB per file

Browse files

Made with Streamlit

Fig 5.6 – Batch mode

7. Batch mode – import CSV file:

How would you like to predict?

Batch

This app is created to predict Customer Churn

CHURN PREDICTION

Telco Customer Churn Prediction App

This Streamlit app is made to predict customer churn in a fictional telecommunication use case. The application is functional for both online prediction and batch data prediction.

Dataset upload

Choose a file

Drag and drop file here
Limit 200MB per file

Browse files

batch_churn.csv 0.9KB

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
0	8773-HHXXX	Female	Yes	No	Yes	6	Yes
1	5945-TMRGD	Female	No	No	Yes	1	Yes
2	7942-YXOOG	Male	No	No	No	5	Yes
3	4598-ABUDE	Female	Yes	Yes	Yes	25	Yes
4	3192-NQECA	Male	No	Yes	No	68	Yes

Predict

Fig 5.7 - Batch mode – import CSV file

8. Attrition prediction in Batch mode:

Dataset upload

Choose a file

Drag and drop file here
Limit 200MB per file

Browse files

batch_churn.csv 0.9KB

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
0	8773-HHXXX	Female	Yes	No	Yes	6	Yes
1	5945-TMRGD	Female	No	No	Yes	1	Yes
2	7942-YXOOG	Male	No	No	No	5	Yes
3	4598-ABUDE	Female	Yes	Yes	Yes	25	Yes
4	3192-NQECA	Male	No	Yes	No	68	Yes

Predict

Prediction

	Predictions
0	Yes, the customer will terminate the service.
1	No, the customer is happy with Telco Services.
2	No, the customer is happy with Telco Services.
3	Yes, the customer will terminate the service.
4	No, the customer is happy with Telco Services.

Fig 5.8 - Attrition prediction in Batch mode

Chapter 6

Conclusion and Future Scope

6.1 Conclusion:

Successfully performed Exploratory Data Analysis and created a model using Logistic Regression Algorithm to accurately predict the Customer Attrition based on different attributes of the Customer to help companies survive in this competitive industry by helping them retain customers and employing strategies to improve their services by understanding the customer behaviour and patterns.

Successfully deployed the model to a Web App created using Stramlit which is intuitive and user-friendly so as to enable users to make use of the model effortlessly and even without much knowledge about the background working of the model.

The App provides two modes:

1. **Online Mode:** In this mode, single customer prediction is provided. The user needs to input the features of the customer using the interactive options provided and the model will predict the attrition of that customer.
2. **Batch Mode:** Batch Mode predicts the prediction for multiple users. The data of multiple customers can be directly uploaded in the form of a csv file and the model will predict the attrition for each of the customers. This makes the app more easy to use even when more data needs to be predicted. This provides extensible application in real life scenarios.

6.2 Future Scope:

- The future scope of this paper will focus on ways to achieve better results such as using different hyper-parameter optimization techniques for the same algorithms in a smaller time frame.
- Different combinations of attributes can be used in the future to determine the customer retention policies.
- Also the performance factors can be improved using different deep learning approaches.
- Improving the User Interface Design to make it more intuitive and user-friendly.

Chapter 7

References

REFERENCES

- [1] Lalwani, Praveen & Mishra, Manas & Chadha, Jasroop & Sethi, Pratyush. (2022).
“Customer churn prediction system: a machine learning approach.”
<https://link.springer.com/article/10.1007/s00607-021-00908-y>
- [2] Xu, Tianpei, Ying Ma, and Kangchul Kim. (2021).
"Telecom Churn Prediction System Based on Ensemble Learning Using Feature Grouping" *Applied Sciences* 11, no. 11: 4742
<https://www.mdpi.com/2076-3417/11/11/4742>
- [3] P. K. Dalvi, S. K. Khandge, A. Deomore, A. Bankar and V. A. Kanade,(2016).
"Analysis of customer churn prediction in telecom industry using decision trees and logistic regression," 2016 Symposium on Colossal Data Analysis and Networking (CDAN), 2016
<https://ieeexplore.ieee.org/document/7570883>
- [4] Ahmad, A.K., Jafar, A. & Aljoumaa, K. (2019).
“Customer churn prediction in telecom using machine learning in big data platform. *J Big Data* 6, 28”
<https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0191-6>
- [5] N. Lu, H. Lin, J. Lu and G. Zhang. (2014)
"A Customer Churn Prediction Model in Telecom Industry Using Boosting," in *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1659-1665,
<https://ieeexplore.ieee.org/document/6329952>
- [6] G. G. Sundarkumar, V. Ravi and V. Siddeshwar. (2015)
"One-class support vector machine based undersampling: Application to churn prediction and insurance fraud detection," 2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC).
<https://ieeexplore.ieee.org/document/7435726>
- [7] Kisioglu, P., & Topcu, Y.I. (2011).
“Applying Bayesian Belief Network approach to customer churn analysis: A case study on the telecom industry of Turkey.” *Expert Syst. Appl.*, 38, 7151-7157.
https://www.researchgate.net/publication/220216955_Applying_Bayesian_Belief_Network_a_approach_to_customer_churn_analysis_A_case_study_on_the_telecom_industry_of_Turkey

Chapter 8

Appendix A: Code Sample

Churn.py:

```
#-----  
#-----Section A: Data Preprocessing-----  
#-----  
  
# Step 1: Import relevant libraries-----  
  
#Standard libraries for data analysis:-----  
import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd  
from scipy.stats import norm, skew  
from scipy import stats  
import statsmodels.api as sm  
  
# sklearn modules for data preprocessing-----  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import LabelEncoder, OneHotEncoder  
from sklearn.compose import ColumnTransformer  
from sklearn.preprocessing import OneHotEncoder  
from sklearn.model_selection import train_test_split  
from sklearn.preprocessing import StandardScaler  
  
#sklearn modules for Model Selection-----  
from sklearn import svm, tree, linear_model, neighbors  
from sklearn import naive_bayes, ensemble, discriminant_analysis, gaussian_process  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis  
from xgboost import XGBClassifier  
from sklearn.linear_model import LogisticRegression  
from sklearn.svm import SVC  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.naive_bayes import GaussianNB  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.ensemble import RandomForestClassifier  
  
#sklearn modules for Model Evaluation & Improvement-----  
from sklearn.metrics import confusion_matrix, accuracy_score  
from sklearn.metrics import f1_score, precision_score, recall_score, fbeta_score  
from statsmodels.stats.outliers_influence import variance_inflation_factor  
from sklearn.model_selection import cross_val_score  
from sklearn.model_selection import GridSearchCV  
from sklearn.model_selection import ShuffleSplit  
from sklearn.model_selection import KFold  
from sklearn import feature_selection  
from sklearn import model_selection  
from sklearn import metrics
```

```

from sklearn.metrics import classification_report, precision_recall_curve
from sklearn.metrics import auc, roc_auc_score, roc_curve
from sklearn.metrics import make_scorer, recall_score, log_loss
from sklearn.metrics import average_precision_score

```

```

#Standard libraries for data visualization-----

```

```

import seaborn as sn
from matplotlib import pyplot
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab
import matplotlib
%matplotlib inline
color = sn.color_palette()
import matplotlib.ticker as mtick
from IPython.display import display
pd.options.display.max_columns = None
from pandas.plotting import scatter_matrix
from sklearn.metrics import roc_curve

```

```

#Miscellaneous Utility Libraries-----

```

```

import random
import os
import re
import sys
import timeit
import string
import time
from datetime import datetime
from time import time
from dateutil.parser import parse
import joblib

```

```

# Step 2: Import the dataset-----

```

```

dataset = pd.read_csv('1.Input/customer_churn_data.csv')

```

```

# Step 3: Evaluate Datastructure -----

```

```

dataset.head()
dataset.columns
dataset.describe()
dataset.dtypes

```

```

#Recheck Column Datatypes and Missing Values:

```

```

dataset.columns.to_series().groupby(dataset.dtypes).groups
dataset.info()
dataset.isna().any()

```

```

#Unique values in each categorical variable:

```

```

dataset["PaymentMethod"].nunique()

```

```
dataset["PaymentMethod"].unique()
dataset["Contract"].nunique()
dataset["Contract"].unique()
```

```
#Step 4: Check Target Variable Distribution -----
dataset["Churn"].value_counts()
```

```
#Step 5: Clean the Dataset-----
dataset["TotalCharges"] = pd.to_numeric(dataset["TotalCharges"],errors='coerce')
dataset["TotalCharges"] = dataset["TotalCharges"].astype("float")
```

```
#Step 6: Take care of missing data-----
dataset.info()
dataset.isna().any()
```

```
##Find the average and fill missing values of each columns programmatically.
na_cols = dataset.isna().any()
na_cols = na_cols[na_cols == True].reset_index()
na_cols = na_cols["index"].tolist()
for col in dataset.columns[1:]:
    if col in na_cols:
        if dataset[col].dtype != 'object':
            dataset[col] = dataset[col].fillna(dataset[col].mean()).round(0)
```

```
#Revalidate:
dataset.isna().any()
```

```
#Step 7: label Encode Binary data-----
#Create a label encoder object
le = LabelEncoder()
# Label Encoding will be used for columns with 2 or less unique values
le_count = 0
for col in dataset.columns[1:]:
    if dataset[col].dtype == 'object':
        if len(list(dataset[col].unique())) <= 2:
            le.fit(dataset[col])
            dataset[col] = le.transform(dataset[col])
            le_count += 1
print('{} columns were label encoded.'.format(le_count))
```

```
#-----
#-----Section B: Data Evaluation-----
#-----
```

```
#Step 8: Exploratory Data Analysis-----
```

#Step 8.1. Plot Histogram of numeric Columns-----

```
dataset2 = dataset[['gender', 'SeniorCitizen', 'Partner', 'Dependents',  
    'tenure', 'PhoneService', 'PaperlessBilling',  
    'MonthlyCharges', 'TotalCharges']]
```

#Histogram:

```
fig = plt.figure(figsize=(15, 12))  
plt.suptitle('Histograms of Numerical Columns\n',horizontalalignment="center",fontstyle =  
"normal", fontsize = 24, fontfamily = "sans-serif")  
for i in range(dataset2.shape[1]):  
    plt.subplot(6, 3, i + 1)  
    f = plt.gca()  
    f.set_title(dataset2.columns.values[i])  
    vals = np.size(dataset2.iloc[:, i].unique())  
    if vals >= 100:  
        vals = 100  
    plt.hist(dataset2.iloc[:, i], bins=vals, color = '#ec838a')  
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```

#Step 8.2. Analyze distribution of Key Categorical Variables----- -----

#(1) Distribution of Contract Type----- -----

```
contract_split = dataset[['customerID', 'Contract']]  
sectors = contract_split.groupby("Contract")  
contract_split = pd.DataFrame(sectors["customerID"].count())  
contract_split.rename(columns={'customerID':'No. of customers'}, inplace=True)  
ax = contract_split[["No. of customers"]].plot.bar(title = 'Customers by Contract Type',  
legend = True, table = False, grid = False, subplots = False, figsize = (12, 7), color  
= '#ec838a', fontsize = 15, stacked=False)  
plt.ylabel('No. of Customers\n',horizontalalignment="center",fontstyle = "normal", fontsize =  
"large", fontfamily = "sans-serif")  
plt.xlabel('\n Contract Type',horizontalalignment="center",fontstyle = "normal", fontsize =  
"large", fontfamily = "sans-serif")  
plt.title('Customers by Contract Type \n',horizontalalignment="center", fontstyle = "normal",  
fontsize = "22", fontfamily = "sans-serif")  
plt.legend(loc='top right', fontsize = "medium")  
plt.xticks(rotation=0, horizontalalignment="center")  
plt.yticks(rotation=0, horizontalalignment="right")  
  
x_labels = np.array(contract_split[["No. of customers"]])  
def add_value_labels(ax, spacing=5):  
    for rect in ax.patches:  
        y_value = rect.get_height()  
        x_value = rect.get_x() + rect.get_width() / 2
```

```

space = spacing
va = 'bottom'
if y_value < 0:
    space *= -1
    va = 'top'
label = "{:.0f}".format(y_value)
ax.annotate(
    label,
    (x_value, y_value),
    xytext=(0, space),
    textcoords="offset points",
    ha='center',
    va=va)
add_value_labels(ax)

```

#(2) Distribution of Payment Method Type-----

```

payment_method_split = dataset[["customerID", "PaymentMethod"]]
sectors = payment_method_split.groupby("PaymentMethod")
payment_method_split = pd.DataFrame(sectors["customerID"].count())
payment_method_split.rename(columns={'customerID': 'No. of customers'}, inplace=True)
ax = payment_method_split[["No. of customers"]].plot.bar(title = 'Customers by Payment Method', legend = True, table = False, grid = False, subplots = False, figsize = (15, 10), color = '#ec838a', fontsize = 15, stacked=False)

```

```

plt.ylabel('No. of Customers\n',horizontalalignment="center",fontstyle = "normal", fontsize = "large", fontfamily = "sans-serif")
plt.xlabel('\n Contract Type',horizontalalignment="center",fontstyle = "normal", fontsize = "large", fontfamily = "sans-serif")
plt.title('Customers by Payment Method \n',horizontalalignment="center", fontstyle = "normal", fontsize = "22", fontfamily = "sans-serif")
plt.legend(loc='top right', fontsize = "medium")
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")

```

```

x_labels = np.array(payment_method_split[["No. of customers"]])

```

```

def add_value_labels(ax, spacing=5):
    for rect in ax.patches:
        y_value = rect.get_height()
        x_value = rect.get_x() + rect.get_width() / 2
        space = spacing
        va = 'bottom'
        if y_value < 0:
            space *= -1
            va = 'top'
        label = "{:.0f}".format(y_value)
        ax.annotate(

```

```

        label,
        (x_value, y_value),
        xytext=(0, space),
        textcoords="offset points",
        ha='center',
        va=va)
add_value_labels(ax)

```

#(3) Distribution of various Label Encoded Categorical Variables-----

```

services = ['PhoneService','MultipleLines','InternetService','OnlineSecurity',
            'OnlineBackup','DeviceProtection','TechSupport','StreamingTV','StreamingMovies']

fig, axes = plt.subplots(nrows = 3,ncols = 3,figsize = (15,12))
for i, item in enumerate(services):
    if i < 3:
        ax = dataset[item].value_counts().plot(kind = 'bar',ax=axes[i,0],rot = 0, color = '#f3babc'
        )
    elif i >=3 and i < 6:
        ax = dataset[item].value_counts().plot(kind = 'bar',ax=axes[i-3,1],rot = 0,color
        = '#9b9c9a')
    elif i < 9:
        ax = dataset[item].value_counts().plot(kind = 'bar',ax=axes[i-6,2],rot = 0,color =
        '#ec838a')
        ax.set_title(item)

```

#Step 8.3: Analyze Churn Rate by Categorical variables: -----

#(1) Overall Churn Rate-----

```

import matplotlib.ticker as mtick
churn_rate = dataset[["Churn", "customerID"]]
churn_rate ["churn_label"] = pd.Series(np.where((churn_rate["Churn"] == 0), "No", "Yes"))
sectors = churn_rate .groupby ("churn_label")
churn_rate = pd.DataFrame(sectors["customerID"].count())
churn_rate ["Churn Rate"] = (churn_rate ["customerID"] / sum(churn_rate ["customerID"])
)*100
ax = churn_rate[["Churn Rate"]].plot.bar(title = 'Overall Churn Rate', legend =True, table =
False, grid = False, subplots = False, figsize =(12, 7), color = '#ec838a', fontsize = 15,
stacked=False, ylim =(0,100))

plt.ylabel('Proportion of Customers',horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.xlabel('Churn',horizontalalignment="center",fontstyle = "normal", fontsize = "large",
fontfamily = "sans-serif")
plt.title('Overall Churn Rate \n',horizontalalignment="center", fontstyle = "normal", fontsize
= "22", fontfamily = "sans-serif")
plt.legend(loc='top right', fontsize = "medium")

```

```

plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
x_labels = np.array(churn_rate[["customerID"]])

```

```

def add_value_labels(ax, spacing=5):
    for rect in ax.patches:
        y_value = rect.get_height()
        x_value = rect.get_x() + rect.get_width() / 2
        space = spacing
        va = 'bottom'
        if y_value < 0:
            space *= -1
            va = 'top'
        label = "{:.1f}%".format(y_value)
        ax.annotate(
            label,
            (x_value, y_value),
            xytext=(0, space),
            textcoords="offset points",
            ha='center',
            va=va)
add_value_labels(ax)
ax.autoscale(enable=False, axis='both', tight=False)

```

#(2) Churn Rate by Contract Type -----

```

import matplotlib.ticker as mtick
contract_churn = dataset.groupby(['Contract', 'Churn']).size().unstack()
contract_churn.rename(columns={0:'No', 1:'Yes'}, inplace=True)
colors = ['#ec838a', '#9b9c9a']
ax = (contract_churn.T*100.0 / contract_churn.T.sum()).T.plot(kind='bar',
        width = 0.3,
        stacked = True,
        rot = 0,
        figsize = (12,7),
        color = colors)

```

```

plt.ylabel('Proportion of Customers\n',horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.xlabel('Contract Type\n',horizontalalignment="center",fontstyle = "normal", fontsize =
"large", fontfamily = "sans-serif")
plt.title('Churn Rate by Contract type \n',horizontalalignment="center", fontstyle = "normal",
fontsize = "22", fontfamily = "sans-serif")
plt.legend(loc='top right', fontsize = "medium")
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
ax.yaxis.set_major_formatter(mtick.PercentFormatter())

```



```

for p in ax.patches:
    width, height = p.get_width(), p.get_height()
    x, y = p.get_xy()
    ax.text(x+width/2,
            y+height/2,
            '{:.1f}%'.format(height),
            horizontalalignment='center',
            verticalalignment='center')
ax.autoscale(enable=False, axis='both', tight=False)

```

#(3) Churn Rate by Payment Method Type-----

```

import matplotlib.ticker as mtick
contract_churn = dataset.groupby(['Contract','PaymentMethod']).size().unstack()
contract_churn.rename(columns={0:'No', 1:'Yes'}, inplace=True)
colors = ['#ec838a','#9b9c9a', '#f3babc' , '#4d4f4c']

ax = (contract_churn.T*100.0 / contract_churn.T.sum()).T.plot(kind='bar',
                                width = 0.3,
                                stacked = True,
                                rot = 0,
                                figsize = (12,7),
                                color = colors)

plt.ylabel('Proportion of Customers\n',horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.xlabel('Contract Type\n',horizontalalignment="center",fontstyle = "normal", fontsize =
"large", fontfamily = "sans-serif")
plt.title('Churn Rate by Payment Method \n',horizontalalignment="center", fontstyle =
"normal", fontsize = "22", fontfamily = "sans-serif")
plt.legend(loc='best', fontsize = "medium")
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
for p in ax.patches:
    width, height = p.get_width(), p.get_height()
    x, y = p.get_xy()
    ax.text(x+width/2,
            y+height/2,
            '{:.1f}%'.format(height),
            horizontalalignment='center',
            verticalalignment='center')
ax.autoscale(enable=False, axis='both', tight=False)

```

Step 8.4. Find positive and negative correlations with the Response Variable-----

```

--
dataset2 = dataset[['SeniorCitizen', 'Partner', 'Dependents',
                    'tenure', 'PhoneService', 'PaperlessBilling',

```

```

'MonthlyCharges', 'TotalCharges']]

correlations = dataset2.corrwith(dataset.Churn)
correlations = correlations[correlations!=1]
positive_correlations = correlations[correlations > 0].sort_values(ascending = False)
negative_correlations = correlations[correlations < 0].sort_values(ascending = False)
print('Most Positive Correlations: \n', positive_correlations)
print("\nMost Negative Correlations: \n", negative_correlations)

#Step 8.5. Plot positive & negative correlation with Response Variable-----
--
correlations = dataset2.corrwith(dataset.Churn)
correlations = correlations[correlations!=1]

correlations.plot.bar(
    figsize = (18, 10), fontsize = 15, color = '#ec838a',
    rot = 45, grid = True)

plt.title('Correlation with Churn Rate \n', horizontalalignment="center", fontstyle = "normal",
    fontsize = "22", fontfamily = "sans-serif")

#Step 8.6. Plot Correlation Matrix of all independent variables-----

## Set and compute the Correlation Matrix
sn.set(style="white")
corr = dataset2.corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure and a diverging colormap
f, ax = plt.subplots(figsize=(18, 15))
cmap = sn.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sn.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
    square=True, linewidths=.5, cbar_kws={"shrink": .5})

#Step 8.7: Check Multicollinearity using VIF-----

def calc_vif(X):
    # Calculating VIF
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

```

```

    return(vif)
dataset2 = dataset[['gender', 'SeniorCitizen', 'Partner', 'Dependents',
    'tenure', 'PhoneService', 'PaperlessBilling', 'MonthlyCharges', 'TotalCharges']]
calc_vif(dataset2)

#Total Charges seem to be colinear with Monthly Charges.
#check colinearity:

dataset2[['MonthlyCharges', 'TotalCharges']].plot.scatter(figsize = (15, 10), x =
'MonthlyCharges',
                                y='TotalCharges', color = '#ec838a')
plt.title('Co-linearity of Monthly Charges and Total Charges
\n',horizontalalignment="center", fontstyle = "normal", fontsize = "22", fontfamily = "sans-
serif")

#dropping TotalCharges:
dataset2 = dataset2.drop(columns = "TotalCharges")

#Revalidate Colinearity:
dataset2 = dataset[['gender', 'SeniorCitizen', 'Partner', 'Dependents',
    'tenure', 'PhoneService', 'PaperlessBilling', 'MonthlyCharges']]

calc_vif(dataset2)
#Applying changes in the main dataset:

dataset = dataset.drop(columns = "TotalCharges")

#Step 9: Encode Categorical data-----
#Incase if user_id is an object:

identity = dataset["customerID"]
dataset = dataset.drop(columns="customerID")
# convert rest of categorical variable into dummy

dataset= pd.get_dummies(dataset)

#Rejoin userid to dataset (column concatenation)
dataset = pd.concat([dataset, identity], axis = 1)

#Step 10: Split dataset into dependent and independent variables-----

#identify response variable:
response = dataset["Churn"]
dataset = dataset.drop(columns="Churn")

#Step 11: Generate training and test datasets of dependent and independent variables-----
-----

```

```

X_train, X_test, y_train, y_test = train_test_split(dataset, response,
                                                    stratify=response,
                                                    test_size = 0.2, #use 0.9 if data is huge.
                                                    random_state = 0)

#to resolve any class imbalance - use stratify parameter.
print("Number transactions X_train dataset: ", X_train.shape)
print("Number transactions y_train dataset: ", y_train.shape)
print("Number transactions X_test dataset: ", X_test.shape)
print("Number transactions y_test dataset: ", y_test.shape)

# Step 12: Removing Identifiers-----
train_identity = X_train['customerID']
X_train = X_train.drop(columns = ['customerID'])

test_identity = X_test['customerID']
X_test = X_test.drop(columns = ['customerID'])

# Step 13: Feature Scaling-----
sc_X = StandardScaler()
X_train2 = pd.DataFrame(sc_X.fit_transform(X_train))
X_train2.columns = X_train.columns.values
X_train2.index = X_train.index.values
X_train = X_train2

X_test2 = pd.DataFrame(sc_X.transform(X_test))
X_test2.columns = X_test.columns.values
X_test2.index = X_test.index.values
X_test = X_test2
#-----
#-----Section C: Model Selection-----
#-----

#Step 14.1: Compare Baseline Classification Algorithms - First Iteration
#Using Accuracy and ROC AUC Mean Metrics

models = []
models.append(('Logistic Regression', LogisticRegression(solver='liblinear', random_state =
0,
                                                         class_weight='balanced'))
models.append(('SVC', SVC(kernel = 'linear', random_state = 0)))
models.append(('Kernel SVM', SVC(kernel = 'rbf', random_state = 0)))
models.append(('KNN', KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p =
2)))
models.append(('Gaussian NB', GaussianNB()))
models.append(('Decision Tree Classifier',
               DecisionTreeClassifier(criterion = 'entropy', random_state = 0)))
models.append(('Random Forest', RandomForestClassifier(

```

```

n_estimators=100, criterion = 'entropy', random_state = 0))

#Evaluating Model Results:
acc_results = []
auc_results = []
names = []
# set table to table to populate with performance results
col = ['Algorithm', 'ROC AUC Mean', 'ROC AUC STD',
       'Accuracy Mean', 'Accuracy STD']
model_results = pd.DataFrame(columns=col)
i = 0
# evaluate each model using k-fold cross-validation
for name, model in models:
    kfold = model_selection.KFold(
        n_splits=10, random_state=0) # 10-fold cross-validation
    cv_acc_results = model_selection.cross_val_score( # accuracy scoring
        model, X_train, y_train, cv=kfold, scoring='accuracy')
    cv_auc_results = model_selection.cross_val_score( # roc_auc scoring
        model, X_train, y_train, cv=kfold, scoring='roc_auc')
    acc_results.append(cv_acc_results)
    auc_results.append(cv_auc_results)
    names.append(name)
    model_results.loc[i] = [name,
                           round(cv_auc_results.mean()*100, 2),
                           round(cv_auc_results.std()*100, 2),
                           round(cv_acc_results.mean()*100, 2),
                           round(cv_acc_results.std()*100, 2)
                           ]
    i += 1

model_results.sort_values(by=['ROC AUC Mean'], ascending=False)

#Step 14.2. Visualize Classification Algorithms Accuracy Comparisons:-----
-----

#Using Accuracy Mean:
fig = plt.figure(figsize=(15, 7))
ax = fig.add_subplot(111)
plt.boxplot(acc_results)
ax.set_xticklabels(names)

#plt.ylabel('ROC AUC Score\n',horizontalalignment="center",fontstyle = "normal", fontsize
= "large", fontfamily = "sans-serif")
#plt.xlabel("\n Baseline Classification Algorithms\n",horizontalalignment="center",fontstyle =
"normal", fontsize = "large", fontfamily = "sans-serif")
plt.title('Accuracy Score Comparison \n',horizontalalignment="center", fontstyle = "normal",
fontsize = "22", fontfamily = "sans-serif")
#plt.legend(loc='top right', fontsize = "medium")

```

```

plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
plt.show()

#using Area under ROC Curve:
fig = plt.figure(figsize=(15, 7))
ax = fig.add_subplot(111)
plt.boxplot(auc_results)
ax.set_xticklabels(names)

#plt.ylabel('ROC AUC Score\n',horizontalalignment="center",fontstyle = "normal", fontsize
= "large", fontfamily = "sans-serif")
#plt.xlabel('\n Baseline Classification Algorithms\n',horizontalalignment="center",fontstyle =
"normal", fontsize = "large", fontfamily = "sans-serif")
plt.title('ROC AUC Comparison \n',horizontalalignment="center", fontstyle = "normal",
fontsize = "22", fontfamily = "sans-serif")
#plt.legend(loc='top right', fontsize = "medium")
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
plt.show()

```

#Step 14.3. Get the right parameters for the baseline models:-----

#Identify optimal number of K neighbors for KNN Model:

```

score_array = []
for each in range(1,25):
    knn_loop = KNeighborsClassifier(n_neighbors = each) #set K neighbor as 3
    knn_loop.fit(X_train,y_train)
    score_array.append(knn_loop.score(X_test,y_test))

```

```

fig = plt.figure(figsize=(15, 7))
plt.plot(range(1,25),score_array, color = '#ec838a')

```

```

plt.ylabel('Range\n',horizontalalignment="center",fontstyle = "normal", fontsize = "large",
fontfamily = "sans-serif")
plt.xlabel('Score\n',horizontalalignment="center",fontstyle = "normal", fontsize = "large",
fontfamily = "sans-serif")
plt.title('Optimal Number of K Neighbors \n',horizontalalignment="center", fontstyle =
"normal", fontsize = "22", fontfamily = "sans-serif")
#plt.legend(loc='top right', fontsize = "medium")
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
plt.show()

```

#Identify optimal number of trees for Random Forest Model:

```

score_array = []
for each in range(1,100):
    rf_loop = RandomForestClassifier(n_estimators = each, random_state = 1)

```

```
rf_loop.fit(X_train,y_train)
score_array.append(rf_loop.score(X_test,y_test))
```

```
fig = plt.figure(figsize=(15, 7))
plt.plot(range(1,100),score_array, color = '#ec838a')
plt.ylabel('Range\n',horizontalalignment="center",fontstyle = "normal", fontsize = "large",
fontfamily = "sans-serif")
plt.xlabel('Score\n',horizontalalignment="center",fontstyle = "normal", fontsize = "large",
fontfamily = "sans-serif")
plt.title('Optimal Number of Trees for Random Forest Model
\n',horizontalalignment="center", fontstyle = "normal", fontsize = "22", fontfamily = "sans-
serif")
#plt.legend(loc='top right', fontsize = "medium")
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
plt.show()
```

#Step 14.4. Compare Baseline Classification Algorithms - Second Iteration-----

#--Step 14.4.1. Logistic Regression-----

```
# Fitting Logistic Regression to the Training set
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

```
#Evaluate results
acc = accuracy_score(y_test, y_pred )
prec = precision_score(y_test, y_pred )
rec = recall_score(y_test, y_pred )
f1 = f1_score(y_test, y_pred )
f2 = fbeta_score(y_test, y_pred, beta=2.0)
```

```
results = pd.DataFrame(['Logistic Regression', acc, prec, rec, f1, f2]),
                        columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score', 'F2 Score'])
```

#Step 14.4.2. . Support Vector Machine (linear classifier)-----

Fitting SVM (SVC class) to the Training set:

```
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)
```

```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

```

#Evaluate results
acc = accuracy_score(y_test, y_pred )
prec = precision_score(y_test, y_pred )
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred )
f2 = fbeta_score(y_test, y_pred, beta=2.0)

model_results = pd.DataFrame([[ 'SVM (Linear)', acc, prec, rec, f1, f2]],
                             columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score', 'F2 Score'])

results = results.append(model_results, ignore_index = True)

```

#Step 14.4.3. K-Nearest Neighbours-----

Fitting KNN to the Training set:

```

classifier = KNeighborsClassifier(n_neighbors = 22, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)

```

Predicting the Test set results

```
y_pred = classifier.predict(X_test)
```

#Evaluate results

```

acc = accuracy_score(y_test, y_pred )
prec = precision_score(y_test, y_pred )
rec = recall_score(y_test, y_pred )
f1 = f1_score(y_test, y_pred )
f2 = fbeta_score(y_test, y_pred, beta=2.0)

```

```

model_results = pd.DataFrame([[ 'K-Nearest Neighbours', acc, prec, rec, f1, f2]],
                             columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score', 'F2 Score'])

```

```
results = results.append(model_results, ignore_index = True)
```

#Step 14.4.4. Kernel SVM-----

Fitting Kernel SVM to the Training set:

```

classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)

```

Predicting the Test set results

```
y_pred = classifier.predict(X_test)
```

#Evaluate results

```

acc = accuracy_score(y_test, y_pred )
prec = precision_score(y_test, y_pred )
rec = recall_score(y_test, y_pred )
f1 = f1_score(y_test, y_pred )
f2 = fbeta_score(y_test, y_pred, beta=2.0)

```



```
model_results = pd.DataFrame([[ 'Kernel SVM', acc, prec, rec, f1, f2]],
                             columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score', 'F2 Score'])
results = results.append(model_results, ignore_index = True)
```

#Step 14.4.5. Naive Byes-----

Fitting Naive Byes to the Training set:

```
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

Predicting the Test set results

```
y_pred = classifier.predict(X_test)
```

#Evaluate results

```
acc = accuracy_score(y_test, y_pred )
prec = precision_score(y_test, y_pred )
rec = recall_score(y_test, y_pred )
f1 = f1_score(y_test, y_pred )
f2 = fbeta_score(y_test, y_pred, beta=2.0)
```

```
model_results = pd.DataFrame([[ 'Naive Byes', acc, prec, rec, f1, f2]],
                             columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score', 'F2 Score'])
```

```
results = results.append(model_results, ignore_index = True)
```

#Step 14.4.6. Decision Tree-----

Fitting Decision Tree to the Training set:

```
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

Predicting the Test set results

```
y_pred = classifier.predict(X_test)
```

#Evaluate results

```
acc = accuracy_score(y_test, y_pred )
prec = precision_score(y_test, y_pred )
rec = recall_score(y_test, y_pred )
f1 = f1_score(y_test, y_pred )
f2 = fbeta_score(y_test, y_pred, beta=2.0)
```

```
model_results = pd.DataFrame([[ 'Decision Tree', acc, prec, rec, f1, f2]],
                             columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score', 'F2 Score'])
```

```

results = results.append(model_results, ignore_index = True)

#Step 14.4.7. Random Forest-----
# Fitting Random Forest to the Training set:

classifier = RandomForestClassifier(n_estimators = 72, criterion = 'entropy', random_state =
0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

#Evaluate results
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score, precision_score,
recall_score
acc = accuracy_score(y_test, y_pred )
prec = precision_score(y_test, y_pred )
rec = recall_score(y_test, y_pred )
f1 = f1_score(y_test, y_pred )
f2 = fbeta_score(y_test, y_pred, beta=2.0)

model_results = pd.DataFrame(['Random Forest', acc, prec, rec, f1, f2]),
                             columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score', 'F2 Score'])

results = results.append(model_results, ignore_index = True)

#Step 14.5. Visualize the results and compare the baseline algorithms-----
-----
results = results.sort_values(["Precision", "Recall", "F2 Score"], ascending = False)
print (results)

#-----
#-----Section D: Model Evaluation (Logistic Regression)-----
#-----
#Step 15: Train & evaluate Chosen Model-----

# Fit Logistic Regression on the Training dataset:
classifier = LogisticRegression(random_state = 0, penalty = 'l2')
classifier.fit(X_train, y_train)

# Predict the Test set results
y_pred = classifier.predict(X_test)

#Evaluate Model Results on Test Set:

```

```

acc = accuracy_score(y_test, y_pred )
prec = precision_score(y_test, y_pred )
rec = recall_score(y_test, y_pred )
f1 = f1_score(y_test, y_pred )
f2 = fbeta_score(y_test, y_pred, beta=2.0)

results = pd.DataFrame(['Logistic Regression', acc, prec, rec, f1, f2],
                        columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score', 'F2 Score'])

print (results)

# Re-check k-Fold Cross Validation:

accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv = 10)
print("Logistic Regression Classifier Accuracy: %0.2f (+/- %0.2f)" % (accuracies.mean(),
accuracies.std() * 2))

#Visualize results on a Confusion Matrix:
cm = confusion_matrix(y_test, y_pred)
df_cm = pd.DataFrame(cm, index = (0, 1), columns = (0, 1))
plt.figure(figsize = (28,20))

fig, ax = plt.subplots()
sn.set(font_scale=1.4)
sn.heatmap(df_cm, annot=True, fmt='g',cmap="YlGnBu"
)
class_names=[0,1]
tick_marks = np.arange(len(class_names))
plt.tight_layout()
plt.title('Confusion matrix\n', y=1.1)
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
ax.xaxis.set_label_position("top")
plt.ylabel('Actual label\n')
plt.xlabel('Predicted label\n')

# Evaluate the model using ROC Graph
classifier.fit(X_train, y_train)
probs = classifier.predict_proba(X_test)
probs = probs[:, 1]
classifier_roc_auc = accuracy_score(y_test, y_pred )
rf_fpr, rf_tpr, rf_thresholds = roc_curve(y_test, classifier.predict_proba(X_test)[:,:1])
plt.figure(figsize=(14, 6))

# Plot Logistic Regression ROC
plt.plot(rf_fpr, rf_tpr, label='Logistic Regression (area = %0.2f)' % classifier_roc_auc)

```

```

# Plot Base Rate ROC
plt.plot([0,1], [0,1],label='Base Rate' 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])

plt.ylabel('True Positive Rate \n',horizontalalignment="center",fontstyle = "normal", fontsize
= "medium", fontfamily = "sans-serif")
plt.xlabel("\nFalse Positive Rate \n',horizontalalignment="center",fontstyle = "normal",
fontsize = "medium", fontfamily = "sans-serif")
plt.title('ROC Graph \n',horizontalalignment="center", fontstyle = "normal", fontsize = "22",
fontfamily = "sans-serif")
plt.legend(loc="lower right", fontsize = "medium")
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
plt.show()

#Step 16:Predict Feature Importance-----
# Analyzing Coefficients
feature_importances = pd.concat([pd.DataFrame(dataset.drop(columns =
'customerID').columns, columns = ["features"]),
    pd.DataFrame(np.transpose(classifier.coef_), columns = ["coef"])
    ],axis = 1)
feature_importances.sort_values("coef", ascending = False)

#-----
#-----Section E: Comparing Model Predictions against test set-----
#-----

# Step 17: Compare predictions against test set -----
#Revalidate final results with Confusion Matrix:
cm = confusion_matrix(y_test, y_pred)
print (cm)

#Confusion Matrix as a quick Crosstab:
pd.crosstab(y_test,pd.Series(y_pred),rownames=['ACTUAL'],colnames=['PRED'])

#visualize Confusion Matrix:
cm = confusion_matrix(y_test, y_pred)
df_cm = pd.DataFrame(cm, index = (0, 1), columns = (0, 1))
plt.figure(figsize = (28,20))

fig, ax = plt.subplots()
sn.set(font_scale=1.4)
sn.heatmap(df_cm, annot=True, fmt='g'#,cmap="YlGnBu"
    )
class_names=[0,1]
tick_marks = np.arange(len(class_names))
plt.tight_layout()
plt.title('Confusion matrix\n', y=1.1)

```

```

plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
ax.xaxis.set_label_position("top")
plt.ylabel('Actual label\n')
plt.xlabel('Predicted label\n')
print("Test Data Accuracy: %0.4f" % accuracy_score(y_test, y_pred))

# Step 18: Format Final Results:-----

final_results = pd.concat([test_identity, y_test], axis = 1).dropna()
final_results['predictions'] = y_pred
final_results["propensity_to_convert(%)"] = y_pred_probs
final_results["propensity_to_convert(%)"] = final_results["propensity_to_convert(%)"]*100
final_results["propensity_to_convert(%)"] = final_results["propensity_to_convert(%)"].round(
2)
final_results = final_results[['customerID', 'Churn', 'predictions', 'propensity_to_convert(%)']]
final_results ['Ranking'] = pd.qcut(final_results['propensity_to_convert(%)'].rank(method =
'first'),10,labels=range(10,0,-1))
print (final_results)

#-----
#-----Section G: Deploy the Model and Predict Future Datapoints-----
#-----

#Step 19: Save the model-----
filename = 'final_model.model'
i = [lr_classifier]
joblib.dump(i,filename)

```

app.py:

```

#Import libraries
import streamlit as st
import pandas as pd
import numpy as np
from PIL import Image

#load the model from disk
import joblib
model = joblib.load(r"D:\Sem6\deploy\churn_prediction_model.sav")

#Import python scripts
from process import preprocess

def main():
    #Setting Application title
    st.title('Telco Customer Churn Prediction App')

    #Setting Application description

```

```

st.markdown("""
:dart: This Streamlit app is made to predict customer churn in a fictional
telecommunication use case.
The application is functional for both online prediction and batch data prediction. \n
""")
st.markdown("<h3></h3>", unsafe_allow_html=True)

#Setting Application sidebar default
image = Image.open('App.jpg')
add_selectbox = st.sidebar.selectbox(
    "How would you like to predict?", ("Online", "Batch"))
st.sidebar.info("This app is created to predict Customer Churn")
st.sidebar.image(image)

if add_selectbox == "Online":
    st.info("Input data below")
    #Based on our optimal features selection
    st.subheader("Demographic data")
    seniorcitizen = st.selectbox('Senior Citizen:', ('Yes', 'No'))
    dependents = st.selectbox('Dependent:', ('Yes', 'No'))
    st.subheader("Payment data")
    tenure = st.slider('Number of months the customer has stayed with the company',
min_value=0, max_value=72, value=0)
    contract = st.selectbox('Contract', ('Month-to-month', 'One year', 'Two year'))
    paperlessbilling = st.selectbox('Paperless Billing', ('Yes', 'No'))
    PaymentMethod = st.selectbox('PaymentMethod',('Electronic check', 'Mailed check',
'Bank transfer (automatic)', 'Credit card (automatic)'))
    monthlycharges = st.slider('The amount charged to the customer monthly', min_value=0,
max_value=150, value=0)
    totalcharges = st.slider('The total amount charged to the customer',min_value=0,
max_value=10000, value=0)
    st.subheader("Services signed up for")
    mutliiplelines = st.selectbox("Does the customer have multiple lines",('Yes','No','No
phone service'))
    phoneservice = st.selectbox('Phone Service:', ('Yes', 'No'))
    internetservice = st.selectbox("Does the customer have internet service", ('DSL', 'Fiber
optic', 'No'))
    onlinesecurity = st.selectbox("Does the customer have online security",('Yes','No','No
internet service'))
    onlinebackup = st.selectbox("Does the customer have online backup",('Yes','No','No
internet service'))
    techsupport = st.selectbox("Does the customer have technology support", ('Yes','No','No
internet service'))
    streamingtv = st.selectbox("Does the customer stream TV", ('Yes','No','No internet
service'))
    streamingmovies = st.selectbox("Does the customer stream movies", ('Yes','No','No
internet service'))

    data = {

```

```

'SeniorCitizen': seniorcitizen,
'Dependents': dependents,
'tenure':tenure,
'PhoneService': phoneservice,
'MultipleLines': mutiplelines,
'InternetService': internetservice,
'OnlineSecurity': onlinesecurity,
'OnlineBackup': onlinebackup,
'TechSupport': techsupport,
'StreamingTV': streamingtv,
'StreamingMovies': streamingmovies,
'Contract': contract,
'PaperlessBilling': paperlessbilling,
'PaymentMethod':PaymentMethod,
'MonthlyCharges': monthlycharges,
'TotalCharges': totalcharges
}
features_df = pd.DataFrame.from_dict([data])
st.markdown("<h3></h3>", unsafe_allow_html=True)
st.write('Overview of input is shown below')
st.markdown("<h3></h3>", unsafe_allow_html=True)
st.dataframe(features_df)
#Preprocess inputs
preprocess_df = preprocess(features_df, 'Online')
prediction = model.predict(preprocess_df)
if st.button('Predict'):
    if prediction == 1:
        st.warning('Yes, the customer will terminate the service.')
    else:
        st.success('No, the customer is happy with Telco Services.')
else:
    st.subheader("Dataset upload")
    uploaded_file = st.file_uploader("Choose a file")
    if uploaded_file is not None:
        data = pd.read_csv(uploaded_file)
        #Get overview of data
        st.write(data.head())
        st.markdown("<h3></h3>", unsafe_allow_html=True)
        #Preprocess inputs
        preprocess_df = preprocess(data, "Batch")
        if st.button('Predict'):
            #Get batch prediction
            prediction = model.predict(preprocess_df)
            prediction_df = pd.DataFrame(prediction, columns=["Predictions"])
            prediction_df = prediction_df.replace({1:'Yes, the customer will terminate the
service.',
0:'No, the customer is happy with Telco Services.'})

            st.markdown("<h3></h3>", unsafe_allow_html=True)

```

```

        st.subheader('Prediction')
        st.write(prediction_df)

if __name__ == '__main__':
    main()

```

process.py:

```

import pandas as pd
from sklearn.preprocessing import MinMaxScaler

```

```

def preprocess(df, option):
    """

```

This function is to cover all the preprocessing steps on the churn dataframe. It involves selecting important features, encoding categorical data, handling missing values, feature scaling and splitting the data

```

    """
    #Defining the map function
    def binary_map(feature):
        return feature.map({'Yes':1, 'No':0})
    # Encode binary categorical features
    binary_list = ['SeniorCitizen','Dependents', 'PhoneService', 'PaperlessBilling']
    df[binary_list] = df[binary_list].apply(binary_map)

    #Drop values based on operational options
    if (option == "Online"):
        columns = ['SeniorCitizen', 'Dependents', 'tenure', 'PhoneService', 'PaperlessBilling',
'MonthlyCharges', 'TotalCharges', 'MultipleLines_No_phone_service', 'MultipleLines_Yes',
'InternetService_Fiber_optic', 'InternetService_No', 'OnlineSecurity_No_internet_service',
'OnlineSecurity_Yes', 'OnlineBackup_No_internet_service',
'TechSupport_No_internet_service', 'TechSupport_Yes', 'StreamingTV_No_internet_service',
'StreamingTV_Yes', 'StreamingMovies_No_internet_service', 'StreamingMovies_Yes',
'Contract_One_year', 'Contract_Two_year', 'PaymentMethod_Electronic_check']
        #Encoding the other categorical features with more than two categories
        df = pd.get_dummies(df).reindex(columns=columns, fill_value=0)
    elif (option == "Batch"):
        pass
    df =
df[['SeniorCitizen','Dependents','tenure','PhoneService','MultipleLines','InternetService','OnlineSecurity',

'OnlineBackup','TechSupport','StreamingTV','StreamingMovies','Contract','PaperlessBilling','
PaymentMethod',
'MonthlyCharges','TotalCharges']]
    columns = ['SeniorCitizen', 'Dependents', 'tenure', 'PhoneService', 'PaperlessBilling',
'MonthlyCharges', 'TotalCharges', 'MultipleLines_No_phone_service', 'MultipleLines_Yes',
'InternetService_Fiber_optic', 'InternetService_No', 'OnlineSecurity_No_internet_service',
'OnlineSecurity_Yes', 'OnlineBackup_No_internet_service',

```



```

'TechSupport_No_internet_service', 'TechSupport_Yes', 'StreamingTV_No_internet_service',
'StreamingTV_Yes', 'StreamingMovies_No_internet_service', 'StreamingMovies_Yes',
'Contract_One_year', 'Contract_Two_year', 'PaymentMethod_Electronic_check']
    #Encoding the other categorical features with more than two categories
    df = pd.get_dummies(df).reindex(columns=columns, fill_value=0)
else:
    print("Incorrect operational options")

#feature scaling
sc = MinMaxScaler()
df['tenure'] = sc.fit_transform(df[['tenure']])
df['MonthlyCharges'] = sc.fit_transform(df[['MonthlyCharges']])
df['TotalCharges'] = sc.fit_transform(df[['TotalCharges']])
return df

```

Chapter 9

Acknowledgements

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to our project guide Prof. AnandKumar Pardeshi for his guidance and support throughout the course of the Mini Project. His constructive advice and constant motivation have been the driving force behind the successful completion of the project. We would like to thank ourselves for believing in us, learning new things, contributing our time and effort to complete this project.

Lastly, we would like to thank all those who helped us directly or indirectly during the project.