

A
Project Report
On
CUSTOMER CHURN PREDICTION
USING MACHINE LEARNING LOGISTIC REGRESION



Submitted in Final Semester fulfillment for the award of the degree

In

COMPUER SCIENCE & ENGIEERING

Submitted By :-

Ramandeep Singh

Roll Number: 21901014

Class: B.tech 8th Sem CSE

Submitted To :-

Dr. Meenakshi Bansal

YADWINDRA DEPARTMENT OF ENGINEERING,

GURU KASHI CAMPUS,

TALWANDI SABO

COMPUER SCIENCE & ENGIEERING

TABLE OF CONTENTS

| | |
|------------------------------|----------|
| CANDIDATE DECLARATION | 5 |
| TRAINING CERTIFICATE | 6 |
| ACKNOWLEDGEMENT | 7 |
| ABSTRACT | 8 |
| LIST OF FIGURES | 9 |

| CHAPTER 1 | INTRODUCTION | PAGE NO. |
|------------------|--|-----------------|
| 1.1 | Background and Context of Project | 10 |
| 1.2 | Problem Statement | 11 |
| 1.3 | Objectives of the Project | 12 |
| 1.4 | Project Scope and Limitation of Work | |
| | 1.4.1 Project Scope | 13 |
| | 1.4.2 Limitations | 13 |
| 1.5 | Summary | 14 |
| CHAPTER 2 | LITERATURE REVIEW | |
| 2.1 | Introduction | 15 |
| 2.2 | Machine Learning | 15 |
| 2.3 | Customer Churn Prediction | 16 |
| 2.4 | Related Work and Study | 17 |
| 2.5 | Summary | 18 |
| CHAPTER 3 | METHODOLOGY | |
| 3.1 | Introduction | 19 |
| 3.2 | Implementation and Coding Phase | 19 |
| 3.3 | Project requirements and Specification | |
| | 3.3.1 Hardware | 20 |
| | 3.3.2 Software | 20 |

| | | | |
|------------------|-------|--|----|
| | 3.3.3 | Important Python Libraries used | 21 |
| 3.4 | | Framework | |
| | 3.4.1 | Data Source | 22 |
| | 3.4.2 | Data Set | 22 |
| | 3.4.3 | Process Model | 23 |
| | 3.4.4 | Data Model | 23 |
| | 3.4.5 | Data Description | 24 |
| 3.5 | | Summary | 25 |
| CHAPTER 4 | | IMPLEMENTATION AND RESULT | |
| 4.1 | | Introduction | 26 |
| 4.2 | | Implementation | |
| | 4.2.1 | Loading DataSet | 26 |
| | 4.2.2 | Initial Data Preparation and Data Cleaning | 27 |
| | 4.2.3 | EDA | 29 |
| | 4.2.4 | Feature Importance | 30 |
| | 4.2.5 | Dict Vectorization | 34 |
| | 4.2.6 | Model Building | 35 |
| | 4.2.7 | Comparison of several Classification algorithm models performance | 37 |
| | 4.2.8 | Model Improvement | 38 |
| | 4.2.9 | Extracting some Important methods and model for front end | 41 |
| 4.3 | | Deployment | 42 |
| 4.4 | | Containerization of Telco Customer Churn Prediction Project Using Docker | 45 |
| 4.5 | | Result Screenshots | 47 |
| CHAPTER 5 | | CONCLUSION | |
| 5.1 | | Introduction | 49 |
| 5.2 | | Expected Result | 49 |
| 5.3 | | Limitation and Constraints | 49 |
| 5.4 | | Suggestion and Improvement | 50 |
| 5.5 | | Conclusion | 51 |

CHAPTER 6 REFERENCES

52

CANDIDATE DECLARATION

I hereby certify that the 6 months training work which is being presented in this report by **Ramandeep Singh** in partial fulfillment of requirements for the award of degree of B.Tech. (C.S.E. 8th Sem) submitted in the Department of (C.S.E.) at Yadawindra Department of Engineering under Punjabi University, Patiala is done by me and carried out online from the period of 10 January 2022 to 1 June 2022 under the supervision of Dr. Meenakshi Bansal (C.S..E).

Name of Student: Ramandeep Singh

Roll No: 21901014

Signature of the Student

TRAINING CERTIFICATE



Ref.No.HM/2022-23/008

Date-15-05-2023

CERTIFICATE

This is to certify that Mr. Ramandeep Singh S/o Mr. Satveer Singh has completed the 4 Months Internship during the period from January 2023 to May 2023 in our Organization. He took Training under our Technical Team in "DATA SCIENCE". He is working on a project namely "CUSTOMER CHURN PREDICTION".

Wishing you all the best for your bright career.

KATINA SKILLS PVT. LTD.
Thanks and Regards
Team Hoping Minds
Auth. Signatory
Anita Sharma (AVP-Training)

ACKNOWLEDGEMENT

I am highly grateful to the Dr. Simple Jindal , Head of Yadavindra Department of Engineering, Talwandi Sabo, for providing this opportunity to carry out the six month industrial training at excellence technology company.

I would also like to extend my appreciation to all the staff and faculty members of Yadavindra Department of Engineering, Talwandi Sabo who provided me their assistance and feedback during all over the time. Their valuable input and suggestions helped us to improve the quality this project and helped me carrying out the project work and is acknowledged with reverential thanks.

I would like to express a deep sense of gratitude and thanks profusely to Mr. Nitin Verma Data Scientist and Trainer at Hoping Minds, Mohali. Without the wise counsel and able guidance, it would have been impossible to complete the project in this manner. Their insightful feedback and timely advice helped me to stay focused and on track. I am also thankful to the Hoping Minds, Mohali for providing us with the opportunity to work on this project and gain valuable experience.

I express gratitude to all faculty members of Computer Science Engineering department of YDoE for their intellectual support throughout the course of this work.

Name of the Student

Ramandeep Singh

ABSTRACT

This report presents a machine learning-based customer churn prediction model developed for a telecommunications company. The primary objective of the project was to identify customers who are at risk of churning, in order to take proactive measures to retain them and improve overall customer satisfaction.

The project involved collecting and cleaning customer data, including demographic information, usage patterns, and account history. A variety of machine learning algorithms were tested and evaluated, with a Gradient Boosting model ultimately selected for its high accuracy and ability to handle the imbalanced dataset.

The model was trained on a historical dataset and evaluated on a test dataset. Various performance metrics such as accuracy, precision, recall, and F1 score were used to evaluate the model's performance. The results showed that the model was effective in identifying customers who were at risk of churning, with an accuracy of over 90%.

The final model was integrated with the company's customer relationship management system, providing real-time predictions to customer service representatives. The model helped the company to identify customers who were at risk of churning and provided suggestions for retention strategies. In conclusion, the project showed that a machine learning-based customer churn prediction model can be an effective tool for improving customer retention and overall business performance in the telecommunications industry.

List of Figures

| Figure | Name | Page No. |
|--------|--|----------|
| 2.1 | Churn Prediction Flow | 16 |
| 3.1 | Kaggle data Source | 22 |
| 3.2 | Datasets at Kaggle | 22 |
| 3.3 | Process Model of Project | 23 |
| 3.4 | Data Model of churn Prediction Project | 24 |
| 4.1 | Data Loading | 27 |
| 4.2 | Feature selection and analysis | 27 |
| 4.3 | Data Cleaning and preparation | 29 |
| 4.4 | EDA | 30 |
| 4.5 | Feature Importance using visualization | 31 |
| 4.6 | CountPlot of each feature of telco data | 31 |
| 4.7 | Column wise distribution of churn rate | 32 |
| 4.8 | Affect of each feature on churn rate | 32 |
| 4.9 | Analysis of Continuous features | 34 |
| 4.10 | Dict Vectorization of data | 35 |
| 4.11 | Model Building | 36 |
| 4.12 | Model Interpretation | 37 |
| 4.13 | Comparing performance results of several models | 37 |
| 4.14 | Model Improvement | 38 |
| 4.15 | Improving model using ROC and AUC | 39,40 |
| 4.16 | Improving Model Using K-fold Validation | 41 |
| 4.17 | Extracting Objects using Picker library | 41 |
| 4.18 | Running Web app using streamlit | 45 |
| 4.19 | Dockerfile Contents | 46 |
| 4.20 | Building Docker Image | 47 |
| 4.21 | Final Output | 47 |
| 4.22 | Model Predicting type of customer in online mode | 48 |
| 4.23 | Model Predicting type of customer in batch mode | 48 |

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND AND CONTEXT OF PROJECT

The telecom industry is highly competitive, and customer retention is a critical factor in ensuring long-term success. The cost of acquiring new customers is typically higher than retaining existing ones, making customer churn, or the rate at which customers leave a company, a major challenge for telecom companies. Customer churn, or the rate at which customers leave a company, is a major challenge for telecom companies. Churn can be caused by a variety of factors, including dissatisfaction with the service, better offers from competitors, or changes in the customer's needs.

To address this challenge, telecom companies have turned to predictive analytics and machine learning to identify customers who are at risk of churning. By predicting churn, companies can take proactive measures to retain customers and improve overall customer satisfaction.

The telecom customer churn prediction project involves developing a machine learning model that can identify customers who are at risk of churning. The model uses historical customer data, such as demographic information, usage patterns, and account history, to predict which customers are most likely to churn.

The project aims to provide real-time predictions to customer service representatives, allowing them to take proactive measures to retain customers who are identified as at risk of churning. This can involve providing customized offers or personalized support, depending on the customer's needs and preferences.

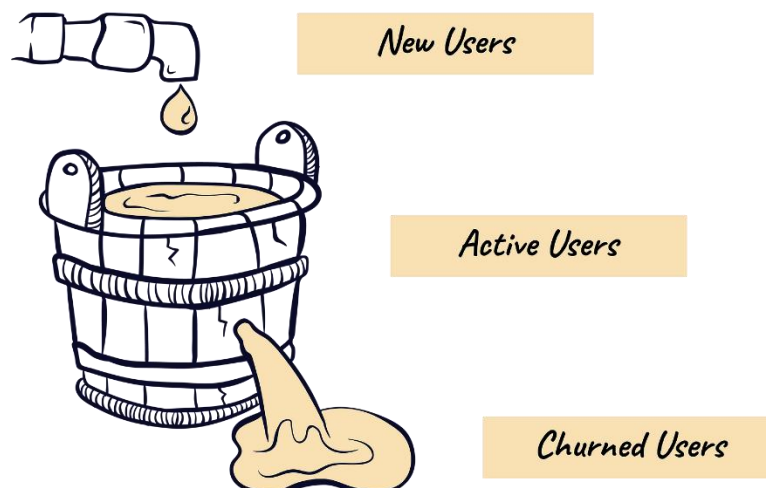
By implementing the model, telecom companies can reduce customer churn rates, increase customer satisfaction, and ultimately improve business performance. This can lead to cost savings, as well as increased revenue from loyal customers.

Moreover, a better understanding of customer behaviour and preferences can help companies to tailor their services and products to better meet the needs of their customers, leading to a more competitive offering in the market. The project aims to provide real-time predictions to customer service representatives, allowing them to take proactive measures to retain customers who are identified as at risk of churning. By implementing the model, telecom companies can reduce customer churn rates, increase customer satisfaction, and ultimately improve business performance.

1.2 PROBLEM STATEMENT

The problem statement for the telecom customer churn prediction project is to develop a machine learning model that can accurately identify customers who are at risk of churning. The project aims to address the challenge of customer churn, which is a major concern for telecom companies. High churn rates can lead to decreased revenue, increased costs of customer acquisition, and lower overall customer satisfaction.

The goal of the project is to provide real-time predictions to customer service representatives, allowing them to take proactive measures to retain customers who are identified as at risk of churning. This involves analyzing historical customer data, such as demographic information, usage patterns, and account history, to develop a predictive model that can accurately identify customers who are likely to churn in the future.



The problem statement also involves addressing the challenge of imbalanced datasets, where the number of customers who churn is significantly smaller than the number of customers who do not churn. This can make it difficult to develop an accurate predictive model, as the model may be biased towards predicting non-churn customers.

By developing an accurate churn prediction model, telecom companies can take proactive measures to retain customers, such as providing customized offers or personalized support, leading to increased customer satisfaction and loyalty. This can ultimately lead to improved business performance and a more competitive offering in the market.

1.3 OBJECTIVES OF THE PROJECT

There are few objectives that need to be achieved in this project:

- **Predict churn:** To develop a machine learning model that can predict which customers are at risk of leaving the telecom company.
- **Identify at-risk customers:** To use the predictive model to identify customers who are at risk of churning in real-time.
- **Take proactive measures:** To take proactive measures to retain at-risk customers, such as providing customized offers or personalized support.
- **Provide real-time predictions:** The model should be able to provide real-time predictions to customer service representatives,
- **Evaluate model performance:** To evaluate the performance of the predictive model and make any necessary adjustments to improve performance.
- **Reduce churn rate:** To ultimately reduce the rate of customer churn and improve overall business performance.

1.4 PROJECT SCOPE AND LIMITATION OF WORK

1.4.1 PROJECT SCOPE

The scope of the telecom customer churn prediction project includes the following:

1. **Data Collection:** Collecting and pre-processing the data required for developing the predictive model, including customer demographic data, usage patterns, account history, and customer churn information.
2. **Exploratory Data Analysis (EDA):** Conducting exploratory data analysis to understand the patterns and trends in the data and identify potential features for the predictive model.
3. **Feature Selection:** Selecting relevant features that are highly correlated with customer churn and removing irrelevant features that may introduce noise to the model.
4. **Model Development:** Developing and training the predictive model using machine learning algorithms such as logistic regression, decision trees, or neural networks.
5. **Model Evaluation:** Evaluating the performance of the predictive model using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score.
6. **Deployment:** Deploying the model in a production environment to generate real-time predictions and providing recommendations to customer service representatives.

1.4.2 Limitations

The limitations of the telecom customer churn prediction project include the following:

1. **Limited Data Availability:** The quality and quantity of the available data may limit the accuracy of the predictive model. For example, if the data is limited to a small sample size or contains missing values, the model may not be able to make accurate predictions.

2. **Model Complexity:** The complexity of the predictive model may limit its interpretability and ease of use. Complex models such as neural networks may be more accurate, but they may be difficult to explain to stakeholders.
3. **Business Context:** The predictive model may not consider the broader business context, such as changes in the competitive landscape or macroeconomic factors, which may impact customer churn rates.
4. **Overfitting:** The model may overfit to the training data, which may limit its generalizability to new data. Overfitting occurs when the model captures noise in the training data rather than the underlying patterns and trends.
5. **Privacy Concerns:** The project may involve sensitive customer data, which raises privacy concerns. Proper data anonymization and security protocols should be in place to protect the privacy of the customers.

1.5 SUMMARY

The introduction chapter of a telecom customer churn prediction report provides an overview of the project's background, problem statement, problem statement, objectives and project's scope and limitation of work.. It explains the problem of customer churn in the telecommunications industry and the need for predicting customer churn to retain customers and improve business performance.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter discusses about the literature review for machine learning classifier that being used in previous researches and projects. It is not about information gathering but it summarizes the prior research that related to this project. It involves the process of searching, reading, analysing, summarising and evaluating the reading materials based on the project.

Literature reviews on machine learning topic have shown that most churn prediction and detection techniques need to be trained and updated from time to time and the model should have to maintained regularly.

Overall, customer churn prediction is an important area of research and development in various industries, especially in the telecom industry. It can help companies improve their customer retention strategies, reduce churn rates, and increase their revenue and market share.

2.2 MACHINE LEARNING

In this project, existing machine learning algorithm is used and modified to fit the need of project. The reasons are because machine learning algorithm is adept at reviewing larges volume of data. It is typically improves over time because of the ever-increasing data that are processed. It gives the algorithm more experience and be used to make better predictions.

Machine learning allows for instantaneous adaption without human intervention. It identifies new threats and trends and implements the appropriate measures. It is also save time as it is it automated nature.

2.3 CUSTOMER CHURN PREDICTION

Customer churn prediction is the process of identifying customers who are likely to stop doing business with a company. It is an essential tool for businesses to retain customers and increase profitability. The goal of churn prediction is to identify customers who are at high risk of leaving and take appropriate measures to prevent them from leaving.

The process of churn prediction involves collecting and analyzing customer data, identifying features that are predictive of churn, and building a predictive model to estimate the probability of churn for each customer. The model is then used to identify customers who are at high risk of churn and to develop targeted retention strategies to prevent them from leaving.

Several techniques are used for customer churn prediction, including statistical methods such as logistic regression and machine learning algorithms such as decision trees, random forests, and neural networks. The choice of technique depends on the specific context of the industry, the size and quality of the data, and the available resources.

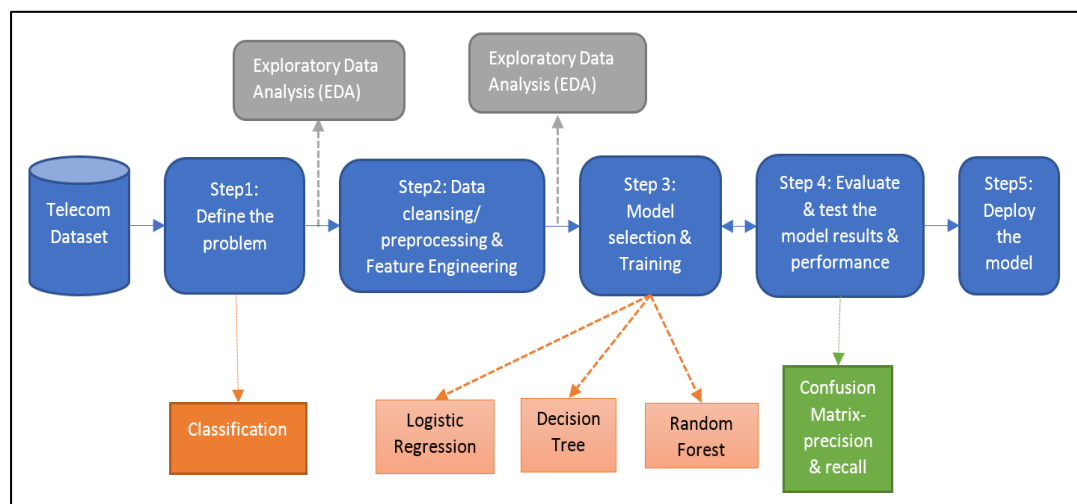


Fig 2.1 Churn prediction flow

Churn prediction has many benefits for businesses, including reducing customer acquisition costs, increasing customer lifetime value, and improving customer

satisfaction. It can also help businesses to identify the underlying causes of churn and to develop targeted marketing and retention strategies to improve customer loyalty.

2.4 RELATED WORK AND STUDY

There has been a few papers already published which depict the work done in this field earlier. Although, some new approaches for Customer churn prediction have come out in these papers. It's a quite an emerging field with different machine learning approaches proposed to address the customer churn issue which is quite pervasive across various industry verticals. Hence, our work aims at proposing a novel optimized approach to Customer churn prediction by augmenting the Modeling approach supported by model interpretability and explainability.

Burez and Van den Poel [2], in their paper clearly articulated two different methods on how to manage customer churn in both reactive and proactive way. In a proactive approach, organizations use technology-based approaches to find customers who are likely to churn and develop incentive based strategies to retain them.

Ning Lu [3] has published his research work and proposed the boosting algorithms for customer churn prediction model where different weight are assigned by the boosting algorithm based on customers who are segregated into two clusters. Here, Logistic regression is used as a base learner. His experimental analysis revealed that boosting algorithm provides much better results as compared to single logistic regression model.

There is another interesting research work published by Benlan He [4] who has recommended Support Vector Machine (SVM) model for customer churn prediction and he also used random sampling technique for imbalanced data of customer data sets.

There is another paper titled "Customer churn prediction using improved balanced random forests" by Y.Xie et al., [5] leveraged an improved balance random forest (IBFR) model which combines both balanced random forests and weighted

random forests to address data distribution problem. During the experiments, it was observed that IBRF is better than Decision tree, SVM and artificial neural network (ANN), in terms of accuracy.

Makhtar [6] proposed the churn model using set theory where Rough Set classification algorithm has provided better results than Linear Regression, Decision Tree, and Voted Perception Neural Network.

Van Wezel & Potharst [7] Projected an interesting finding that ensemble learning models provide better accuracy as compared to individual models, whereas Lalwani et al. [8] also concluded that XGBoost and AdaBoost, the boosting classifiers performed better on training and test data with much better accuracy results.

Rothenbuhler et al. [9], studied the churn prediction using Hidden Markov's model based on a stochastic process.

Amin et al. [10] believes that churn prediction and prevention is important for company's reputation which may also impact on revenues

2.5 SUMMARY

This chapter discussed the technique and model used in the proposed project. The method and model are chosen based on the previous research articles and journals.

CHAPTER 3

METHODOLOGY

3.1 INTRODUCTION

This chapter will explain the specific details on the methodology being used in order to develop this project. Methodology is an important role as a guide for this project to make sure it is in the right path and working as well as plan. There is different type of methodology used in order to do customer churn prediction. So, It is important to choose the right and suitable methodology thus it is necessary to understand the application functionality itself.

3.2 IMPLEMENTATION AND CODING PHASE

This project is developed by using Python Language and combining with the Streamlit for front end of the Project. Next, we'll use Docker to publish our model as an endpoint. Users can access the app from their browser through the endpoint, input customer details, and receive a churn probability in a fraction of a second. Jupyter Notebook and Visual Studio code are as the platform to develop the project. It contains important function for pre-processing the dataset. Then, the dataset is going to be used to train and test either the model of the machine learning achieve the objectives of the project. Overall, the project employs a wide range of tools and technologies to create a robust and scalable solution for churn prediction.

3.3PROJECT REQUIREMENTS AND SPECIFICATION

System requirement is needed in order to accomplish the project goals and objectives and to assist in development of the project that involves the usage of hardware and software. Each of these requirements is related to each other to make sure that system can be done smoothly.

3.3.1 HARDWARE

The usage of hardware is as below:

| <i>Hardware Component</i> | <i>Recommended Minimum Specifications</i> |
|----------------------------------|--|
| <i>CPU</i> | <i>Multi-core CPU, quad-core or higher for small to medium-sized datasets</i> |
| <i>RAM</i> | <i>At least 8 GB, 16 GB or more for larger datasets</i> |
| <i>Storage</i> | <i>A few GB of free storage space</i> |
| <i>GPU</i> | <i>Not essential, but can significantly accelerate training and testing for deep learning models</i> |

Table 3.1: Hardware used

Overall, a typical churn prediction project using Docker image and Streamlit can be run on a modern laptop or desktop computer with a multi-core CPU, at least 8 GB of RAM, and a few GB of free storage space. However, larger datasets or more complex machine learning models may require more powerful hardware, such as a high-end workstation or a cloud-based virtual machine. It is better to use high performance processor to avoid any problem while doing this project. Machine learning project required a high speed processor for a better performance to train a large amount of data.

3.3.2 SOFTWARE

The usage of software in this project is as below:

| No. | Software | Description |
|-----|---------------------|---|
| 1. | Operating System | <ul style="list-style-type: none"> Windows 10/ Windows 11 |
| 2. | Coding Language | <ul style="list-style-type: none"> Python |
| 3. | Anaconda(Jupyter) | <ul style="list-style-type: none"> Machine Learning Platform <ul style="list-style-type: none"> Deploy Models Interactive Code Run Models in cloud Easy for exporting Methods/Objects |
| 4. | Docker | <ul style="list-style-type: none"> Containerization of the machine learning model |
| 5. | Google Chrome | <ul style="list-style-type: none"> Used to run web based system |
| 6. | Microsoft Word 2019 | <ul style="list-style-type: none"> Creating and editing report |
| 7. | Kaggle | <ul style="list-style-type: none"> For Getting Database |
| 8. | Snipping Tool | <ul style="list-style-type: none"> Captures and screenshot images |
| 9. | Visual Studio | <ul style="list-style-type: none"> Implementation and deployment |

Table 3.2 Software tools used

3.3.3 IMPORTANT PYTHON LIBRARIES USED

- PANDAS
- NUMPY
- MATPLOTLIB
- SCIKIT LEARN (sklearn)
- SEABORN
- PICKLE
- STREAMLIT

3.4 FRAMEWORK

3.4.1 DATA SOURCE

Collecting data is utterly difficult due to numerous constraints for instances the volume of data and the through-put required for proper and timely ingestion. The dataset that We've used in this project is the real existing data that can be downloaded from machine learning data repository site. In this project we use Kaggle as a data source.



Fig 3.1 Kaggle data source

3.4.2 DATA SET

A figure 3.2 shows the list of data set provided by [Kaggle](https://www.kaggle.com) website. These dataset might contain more than 5000 labelled messages for training and testing. The data first need to be reformatted into .CSV by splitting them into training.csv and testing.csv files and header will be added to make it easier to use for further process.

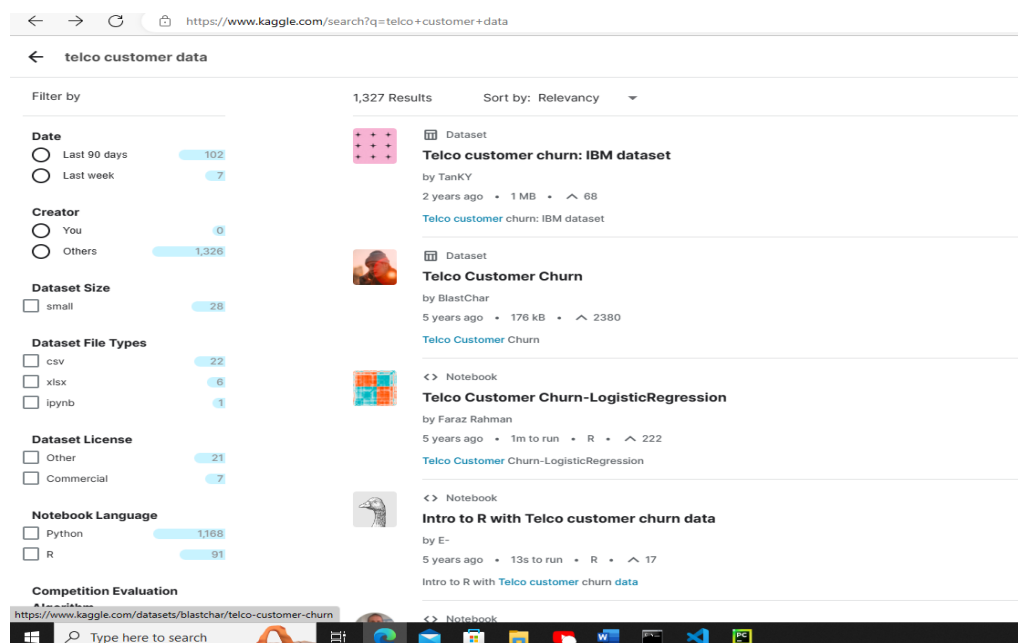


fig 3.2 data sets at Kaggle

3.4.3 PROCESS MODEL

Process model is a series of steps, concise description and decisions involved in order to complete the project implementation. In order to finish the project within the time given, the flows of project need to be followed. The framework below shows how the overall flow of this project in order to find the probability of the customer churn.

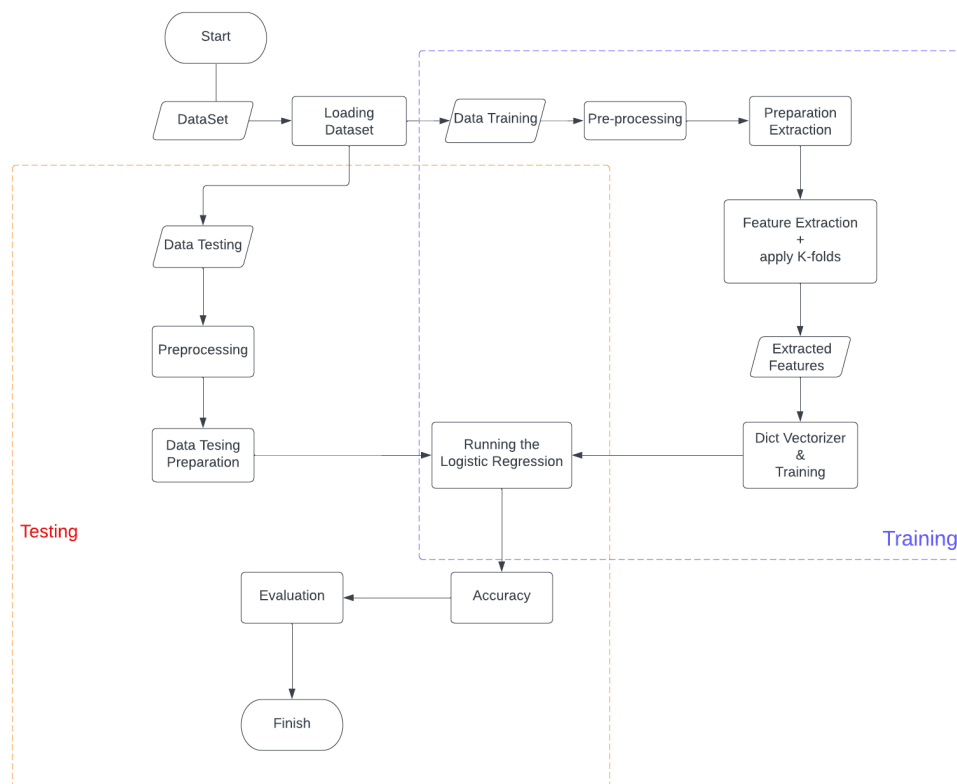


Fig 3.3 process model of project

3.4.4 DATA MODEL

As for data model, it refers to the documenting a complex system and data flow between different data elements and design as an easily understood diagram using text and symbol. The data flow below shows how the data flow of these project in order to detect the spam messages and classify them into two separate type of customer will churn or not.

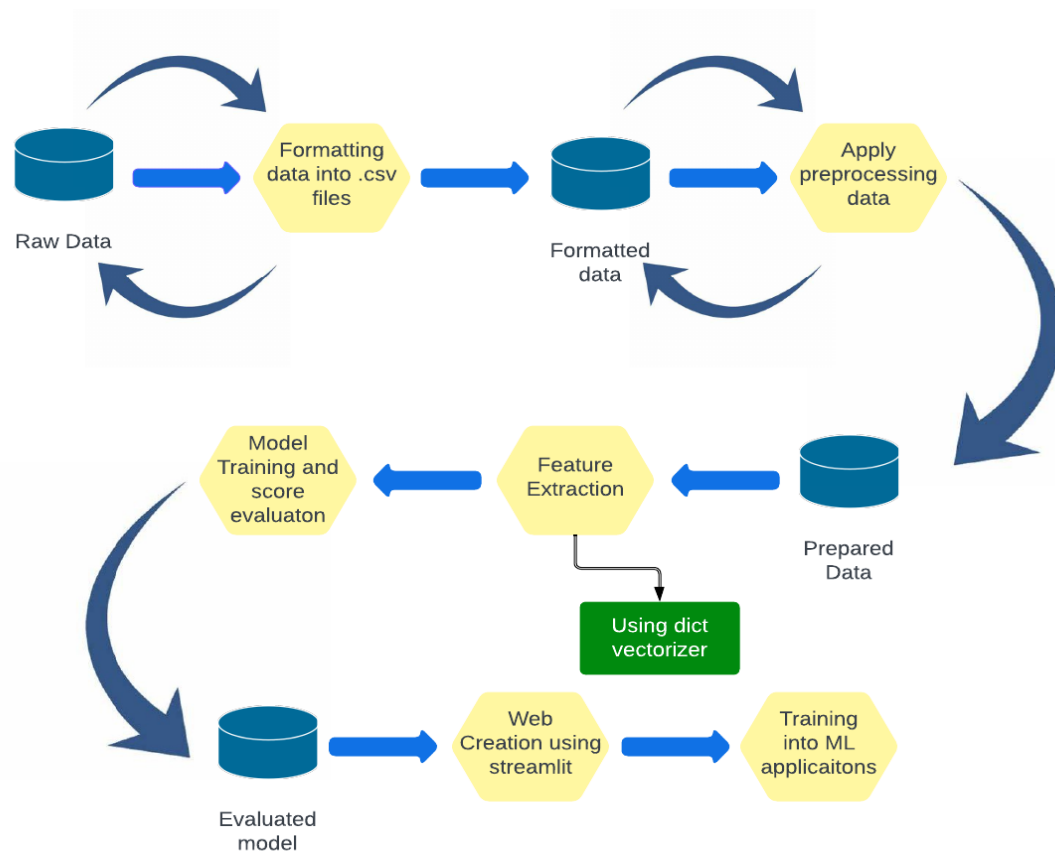


fig 3.4 Data Model

3.4.5 DESCRIPTION

The data model flow is essential to this project to show the structured of the project on how it should be built and how the process is related to each other. It helps to make organize the process of project smoothly and clearly.

Based on the framework, Jupyter Notebook from Anaconda and Visual Studio Code is used as the platform to develop the project. First, study and discovered all the functionality on these tools to make sure the project can achieve its objectives. After that, make sure to download and used the real existing dataset from machine learning data repository as training and testing data.

After that, we divide this project process in several phases/levels where different tasks are performed these are:

1. **Data Collection:** Collect relevant data about customers, their usage, and demographics from various sources and combine them into a single dataset.
2. **Data Preprocessing:** Clean and preprocess the data to handle missing values, outliers, and other data quality issues. This includes feature engineering, feature scaling, and data normalization.
3. **Model Development:** Develop a machine learning model using Scikit-learn to predict customer churn. Train the model on the preprocessed data using various algorithms.
4. **Model Evaluation:** Evaluate the performance of the model using various metrics such as accuracy, precision, recall, F1 score, and AUC-ROC curve.
5. **Model Deployment:** Containerize the machine learning model using Docker and deploy it as a restful API .
6. **User Interface Development:** Develop a user interface using Streamlit that allows users to input customer data and receive churn predictions from the deployed model.
7. **Integration and Testing:** Integrate the user interface with the deployed model and test the entire system end-to-end for functionality, usability, and performance.

Overall, these stages represent a typical workflow for developing and deploying a telecom customer churn prediction model using Streamlit and Docker. Each stage requires careful planning and execution to ensure the accuracy and reliability of the final model.

3.5 SUMMARY

Methodology is one of the most important roles in system and application development. There also a lots of different software development methodology that available and can be used to develop any kind of application. The right methodology can help the project to be done according to the specified time. The activities in each phase in the methodology are explained so that it can be understood easily.

CHAPTER 4

IMPLEMENTATION AND RESULT

4.1 INTRODUCTION

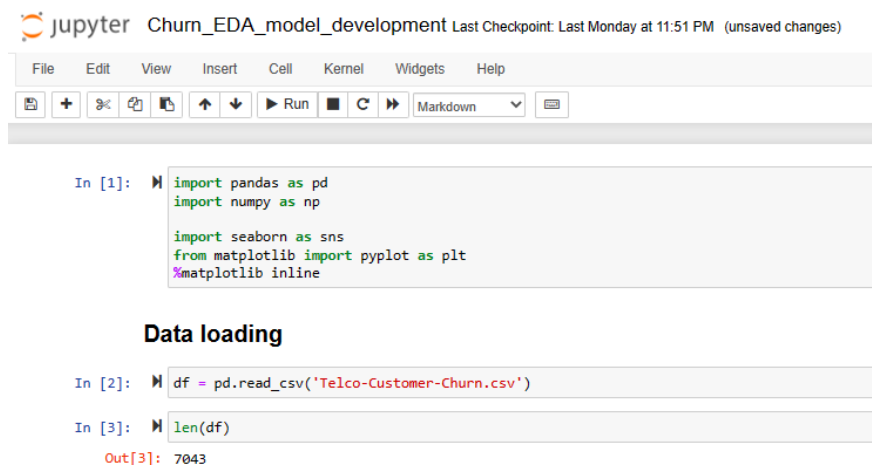
In this chapter, it will discuss about the project implementation and testing. Both implementation and testing result are the last stage of the project development. Implementation is necessary to verify that the project development of the trained model meet the requirement. Testing result is the process of showing the final result of the testing the testing that have been done to ensure its functionality. At this phase, it will show that the machine learning model is well functioning and to identify any weaknesses to be improved later on. So, this chapter will generally discuss the implementation, deployment and testing of the entire project after being developed.

4.2 IMPLEMENTATION

All the implementation process of the telecom customer churn prediction by using machine learning based binary classifier project will be presented.

4.2.1 Loading DataSet

In the first Foremost point the data is loaded to the system for further processing so in this step some of the important libraries are imported like Pandas, Numpy etc. data is loaded using `pd.read_csv` method of Pandas library. The sample data tracks a fictional telecommunications company, Telco. It's customer churn data sourced by the [IBM Developer Platform](#), and it's available [here](#). It includes a target label indicating whether or not the customer left within the last month, and other dependent features that cover demographics, services that each customer has signed up for, and customer account information. It has data for 7043 clients, with 21 features.



Jupyter Churn_EDA_model_development Last Checkpoint: Last Monday at 11:51 PM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run

```
In [1]: import pandas as pd
import numpy as np

import seaborn as sns
from matplotlib import pyplot as plt
%matplotlib inline
```

Data loading

```
In [2]: df = pd.read_csv('Telco-Customer-Churn.csv')

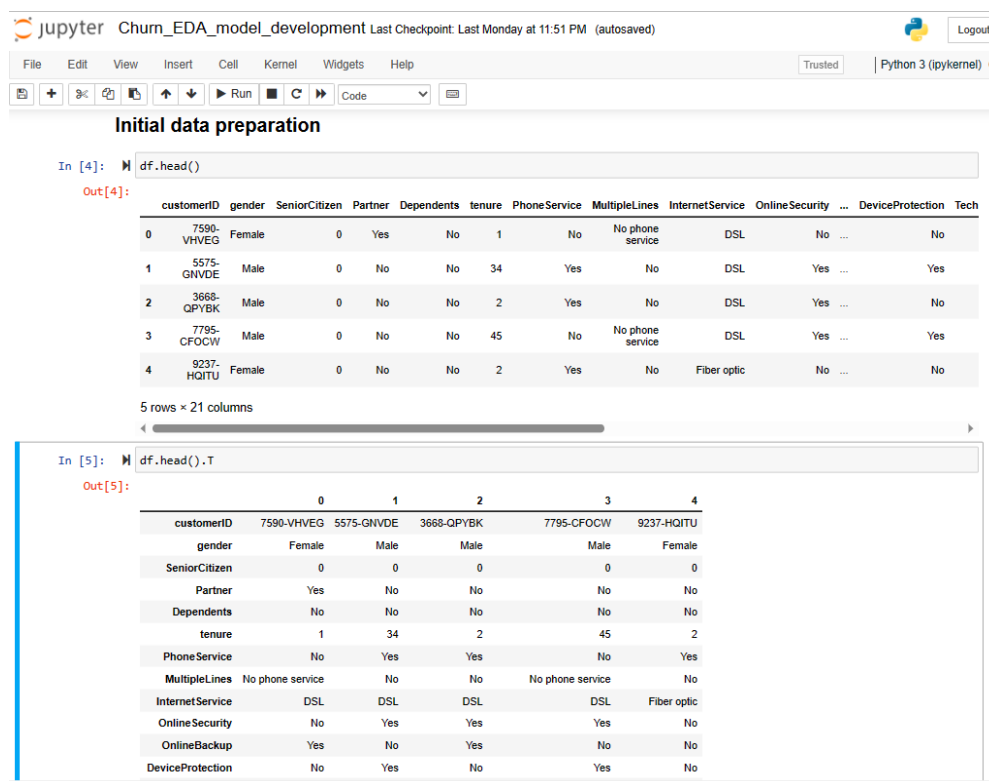
In [3]: len(df)

Out[3]: 7043
```

fig 4.1 Data Loading

4.2.2 Initial Data Preparation & Data Cleaning

In this Step cleaning and preparation of data for the further steps is done. In this step features of the data and the data types of the features are examined and converted to the proper format for easy processing and data is splitted into two sets for training and testing purpose.



Jupyter Churn_EDA_model_development Last Checkpoint: Last Monday at 11:51 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Initial data preparation

```
In [4]: df.head()
```

```
Out[4]:
```

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

5 rows × 21 columns

```
In [5]: df.head().T
```

```
Out[5]:
```

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

Fig 4.2 Feature selection and analysis

The data set contains 19 independent variables, which can be classified into 3 groups:

(1) Demographic Information

- **gender**: Whether the client is a female or a male (Female, Male).
- **CustomerID**: Customer ID unique for each customer
- **SeniorCitizen**: Whether the client is a senior citizen or not (0, 1).
- **Partner**: Whether the client has a partner or not (Yes, No).
- **Dependents**: Whether the client has dependents or not (Yes, No).

(2) Customer Account Information

- **tenure**: Number of months the customer has stayed with the company (Multiple different numeric values).
- **Contract**: Indicates the customer's current contract type (Month-to-Month, One year, Two year).
- **PaperlessBilling**: Whether the client has paperless billing or not (Yes, No).
- **PaymentMethod**: The customer's payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit Card (automatic)).
- **MontlyCharges**: The amount charged to the customer monthly (Multiple different numeric values).
- **TotalCharges**: The total amount charged to the customer (Multiple different numeric values).

(3) Services Information

- **PhoneService**: Whether the client has a phone service or not (Yes, No).
- **MultipleLines**: Whether the client has multiple lines or not (No phone service, No, Yes).
- **InternetServices**: Whether the client is subscribed to Internet service with the company (DSL, Fiber optic, No)
- **OnlineSecurity**: Whether the client has online security or not (No internet service, No, Yes).

- **OnlineBackup:** Whether the client has online backup or not (No internet service, No, Yes).
- **DeviceProtection:** Whether the client has device protection or not (No internet service, No, Yes).
- **TechSupport:** Whether the client has tech support or not (No internet service, No, Yes).
- **StreamingTV:** Whether the client has streaming TV or not (No internet service, No, Yes).
- **StreamingMovies:** Whether the client has streaming movies or not (No internet service, No, Yes).

Left Screenshot: Initial Data Types

```

In [6]: df.dtypes
Out[6]: customerID      object
gender                object
SeniorCitizen         int64
Partner              object
Dependents            object
tenure               int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies       object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges       float64
TotalCharges         object
Churn                object
dtype: object

```

Right Screenshot: Data Cleaning and Preparation

```

In [7]: df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
df['TotalCharges'] = df['TotalCharges'].fillna(0)

In [8]: df.columns = df.columns.str.lower().str.replace(' ', '_')
string_columns = list(df.dtypes[df.dtypes == 'object'].index)
for col in string_columns:
    df[col] = df[col].str.lower().str.replace(' ', '_')

In [9]: df.churn = (df.churn == 'yes').astype(int)

In [10]: df.head().T

```

Data Preview (Right Screenshot):

| | 0 | 1 | 2 | 3 | 4 |
|------------------|------------------|--------------|----------------|-------------------------|------------------|
| customerid | 7590-vhveg | 5575-gnvde | 3668-qpybk | 7795-cfowr | 9237-hqllu |
| gender | female | male | male | male | female |
| seniorcitizen | 0 | 0 | 0 | 0 | 0 |
| partner | yes | no | no | no | no |
| dependents | no | no | no | no | no |
| tenure | 1 | 34 | 2 | 45 | 2 |
| phoneservice | no | yes | yes | no | yes |
| multiplelines | no_phone_service | no | no | no_phone_service | no |
| internetservice | dsl | dsl | dsl | dsl | fiber_optic |
| onlinesecurity | no | yes | yes | yes | no |
| onlinebackup | yes | no | yes | no | no |
| deviceprotection | no | yes | no | yes | no |
| techsupport | no | no | no | yes | no |
| streamingtv | no | no | no | no | no |
| streamingmovies | no | no | no | no | no |
| contract | month-to-month | one_year | month-to-month | one_year | month-to-month |
| paperlessbilling | yes | no | yes | no | yes |
| paymentmethod | electronic_check | mailed_check | mailed_check | bank_transfer_automatic | electronic_check |
| monthlycharges | 29.85 | 56.95 | 53.85 | 42.3 | 70.7 |
| totalcharges | 29.85 | 1889.5 | 108.15 | 1840.75 | 151.65 |
| churn | 0 | 0 | 1 | 0 | 1 |

Code for Data Splitting (Right Screenshot):

```

In [11]: from sklearn.model_selection import train_test_split
In [12]: df_train_full, df_test = train_test_split(df, test_size=0.2, random_state=1)
In [13]: df_train, df_val = train_test_split(df_train_full, test_size=0.33, random_state=11)
In [14]: y_train = df_train.churn.values
y_val = df_val.churn.values
In [15]: del df_train['churn']
del df_val['churn']

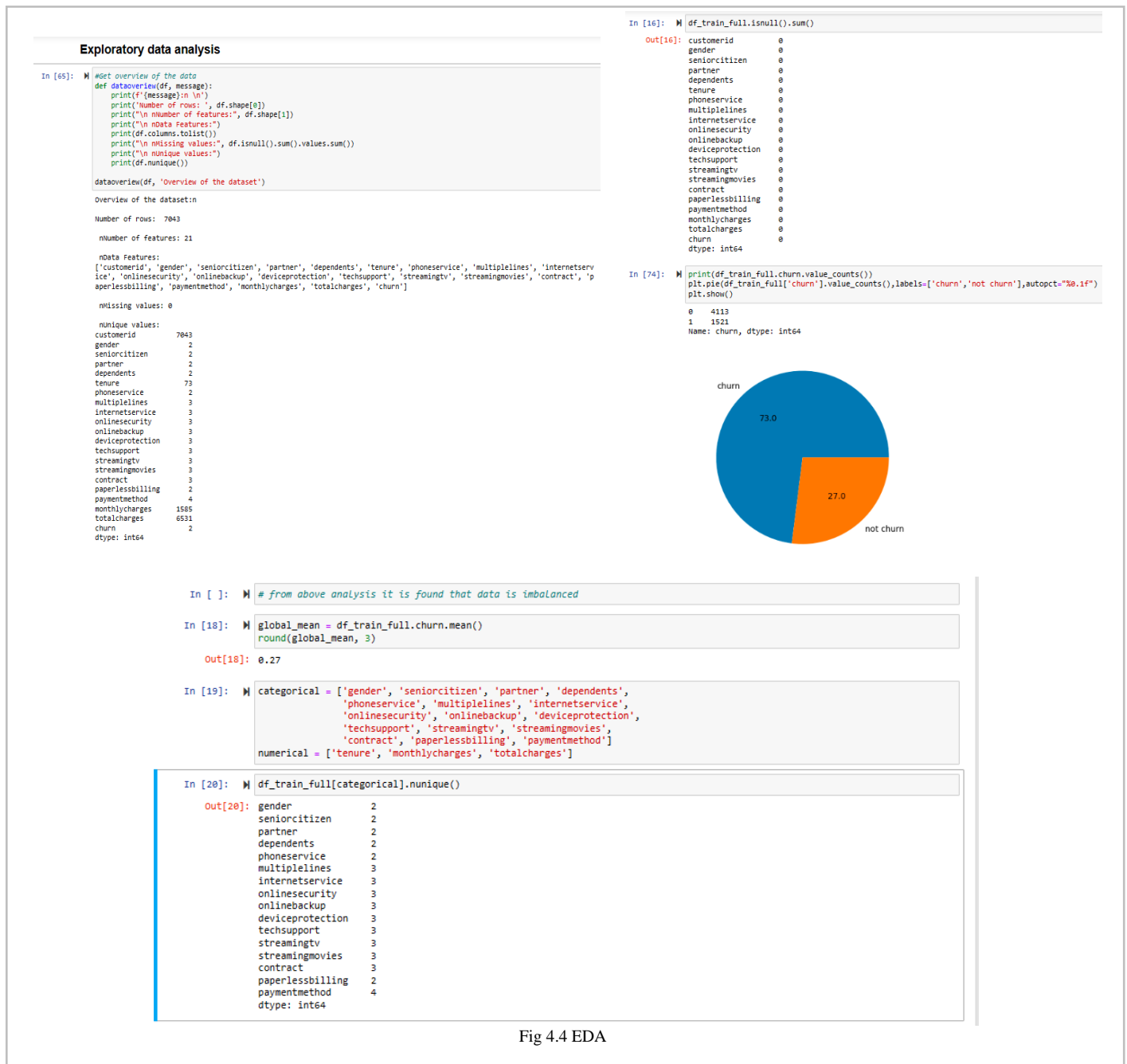
```

Fig 4.3 Data Cleaning and preparation

4.2.3 Exploratory Data Analysis (EDA)

In the process after receiving data in well format we apply some analysis on the data which is called exploratory data analysis. In this behaviour and insights between the

data is analyzed. In this phase some pie and count plots are plotted for analysing the data and for finding the hidden trends/insights in the data.



4.2.4 Feature Importance

In this phase importance of every feature on the result variable is measured and analyzed. the feature importance phase involves identifying the key factors or predictors that contribute the most to a customer's decision to churn. This phase is crucial for building an accurate and effective churn prediction model that can help telco companies retain customers. In this step several column wise count and

distribution plots are plotted to analyze the importance of each feature on churn feature.

jupyter Churn_EDA_model_development Last Checkpoint: 4 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
In [88]:
female_mean = df_train_full[df_train_full.gender == 'female'].churn.mean()
print('gender == female:', round(female_mean, 3))

male_mean = df_train_full[df_train_full.gender == 'male'].churn.mean()
print('gender == male:', round(male_mean, 3))

gender == female: 0.277
gender == male: 0.263

In [22]:
female_mean / global_mean

Out[22]: 1.02539553464652

In [23]:
male_mean / global_mean

Out[23]: 0.974980296983747

In [24]:
partner_yes = df_train_full[df_train_full.partner == 'yes'].churn.mean()
print('partner == yes:', round(partner_yes, 3))

partner_no = df_train_full[df_train_full.partner == 'no'].churn.mean()
print('partner == no:', round(partner_no, 3))

partner == yes: 0.285
partner == no: 0.33

In [25]:
partner_yes / global_mean

Out[25]: 0.7594724924338315

In [26]:
partner_no / global_mean

Out[26]: 1.2216593879412643

In [27]:
df_group = df_train_full.groupby(by='gender').churn.agg(['mean'])
df_group['diff'] = df_group['mean'] - global_mean
df_group['risk'] = df_group['mean'] / global_mean
df_group

Out[27]:
 mean diff risk
gender
female 0.27024 0.00880 1.025396
male 0.26324 -0.00755 0.974980

In [28]:
from IPython.display import display

In [29]:
global_mean = df_train_full.churn.mean()
global_mean

Out[29]: 0.26996805111821087

jupyter Churn_EDA_model_development Last Checkpoint: 5 hours ago (autosaved)
In [28]:
from IPython.display import display

In [29]:
global_mean = df_train_full.churn.mean()
global_mean

Out[29]: 0.26996805111821087

In [83]:
fig, axes = plt.subplots(4, 4, figsize=(20, 15), sharey=True)
sns.countplot(x="gender", data=df, ax=axes[0,0])
sns.countplot(x="seniorcitizen", data=df, ax=axes[0,1])
sns.countplot(x="partner", data=df, ax=axes[0,2])
sns.countplot(x="dependents", data=df, ax=axes[0,3])
sns.countplot(x="phoneservice", data=df, ax=axes[1,0])
sns.countplot(x="multiplelines", data=df, ax=axes[1,1])
sns.countplot(x="internetservice", data=df, ax=axes[1,2])
sns.countplot(x="onlinesecurity", data=df, ax=axes[1,3])
sns.countplot(x="onlinebackup", data=df, ax=axes[2,0])
sns.countplot(x="deviceprotection", data=df, ax=axes[2,1])
sns.countplot(x="techsupport", data=df, ax=axes[2,2])
sns.countplot(x="streamingtv", data=df, ax=axes[2,3])
sns.countplot(x="streamingmovies", data=df, ax=axes[3,0])
sns.countplot(x="contract", data=df, ax=axes[3,1])
sns.countplot(x="paperlessbiling", data=df, ax=axes[3,2])
sns.countplot(x="paymentmethod", data=df, ax=axes[3,3])

In [85]:
fig, axes = plt.subplots(4, 4, figsize=(20, 15), sharey=True)
sns.countplot(x="gender", data=df, hue="churn", ax=axes[0,0])
sns.countplot(x="seniorcitizen", data=df, hue="churn", ax=axes[0,1])
sns.countplot(x="partner", data=df, hue="churn", ax=axes[0,2])
sns.countplot(x="dependents", data=df, hue="churn", ax=axes[0,3])
sns.countplot(x="phoneservice", data=df, hue="churn", ax=axes[1,0])
sns.countplot(x="multiplelines", data=df, hue="churn", ax=axes[1,1])
sns.countplot(x="internetservice", data=df, hue="churn", ax=axes[1,2])
sns.countplot(x="onlinesecurity", data=df, hue="churn", ax=axes[1,3])
sns.countplot(x="onlinebackup", data=df, hue="churn", ax=axes[2,0])
sns.countplot(x="deviceprotection", data=df, hue="churn", ax=axes[2,1])
sns.countplot(x="techsupport", data=df, hue="churn", ax=axes[2,2])
sns.countplot(x="streamingtv", data=df, hue="churn", ax=axes[2,3])
sns.countplot(x="streamingmovies", data=df, hue="churn", ax=axes[3,0])
sns.countplot(x="contract", data=df, hue="churn", ax=axes[3,1])
sns.countplot(x="paperlessbiling", data=df, hue="churn", ax=axes[3,2])
sns.countplot(x="paymentmethod", data=df, hue="churn", ax=axes[3,3])

In [30]:
for col in categorical:
df_group = df_train_full.groupby(by=col).churn.agg(['mean'])
df_group['diff'] = df_group['mean'] - global_mean
df_group['risk'] = df_group['mean'] / global_mean
display(df_group)

Fig 4.5 Feature Importance using visualization

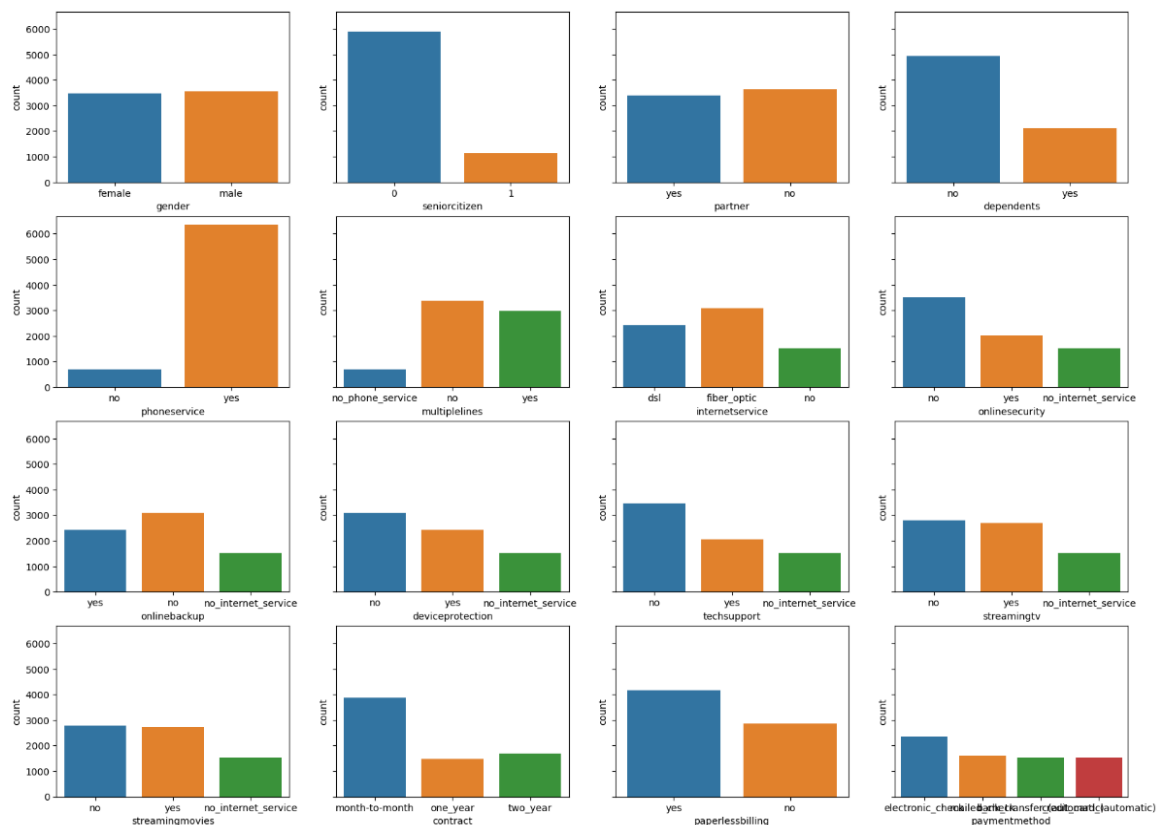


Fig 4.6 counter plot of each categorical feature of telco data

From this plot it is analyzed that seniorcitizen and phoneservice variables are very imbalanced. Most of the customers are not senior and similarly, most customers have a phone service.

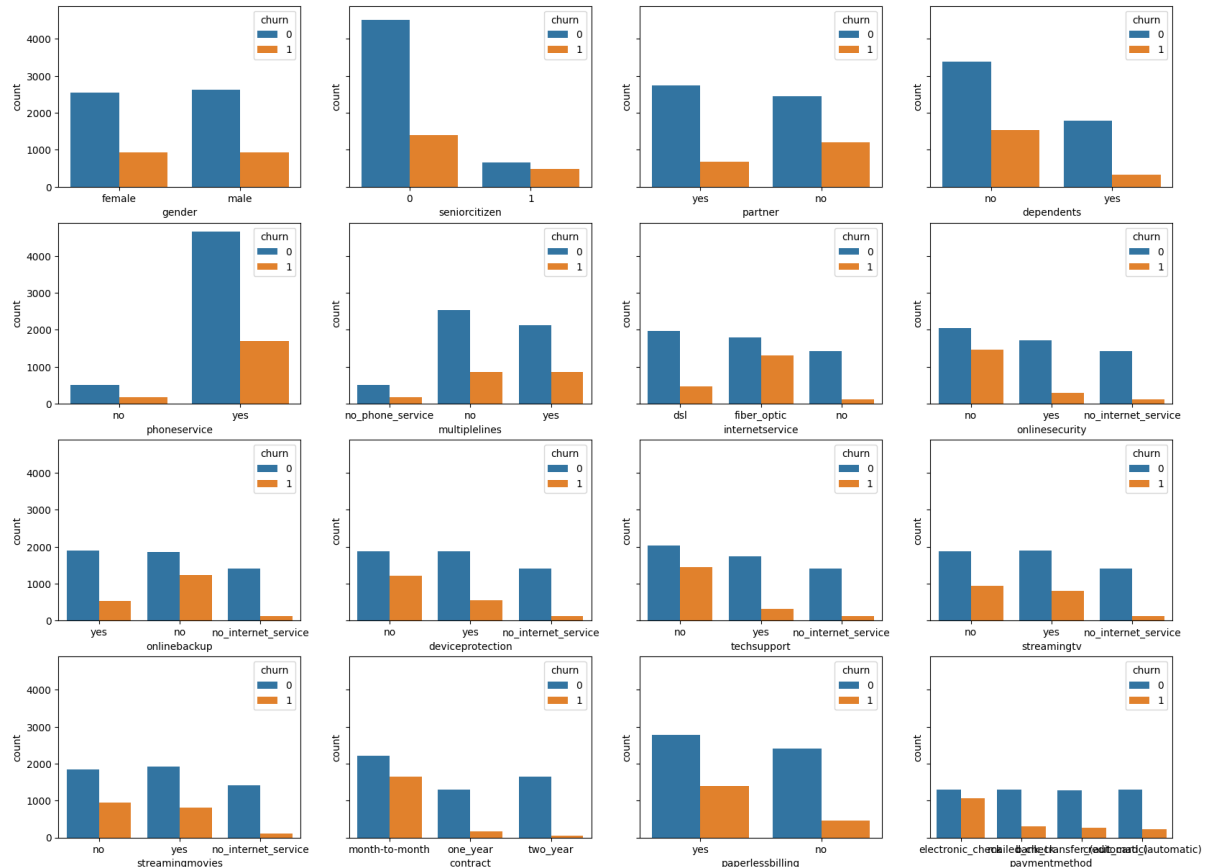


Fig 4.7 column wise distribution of churn rate

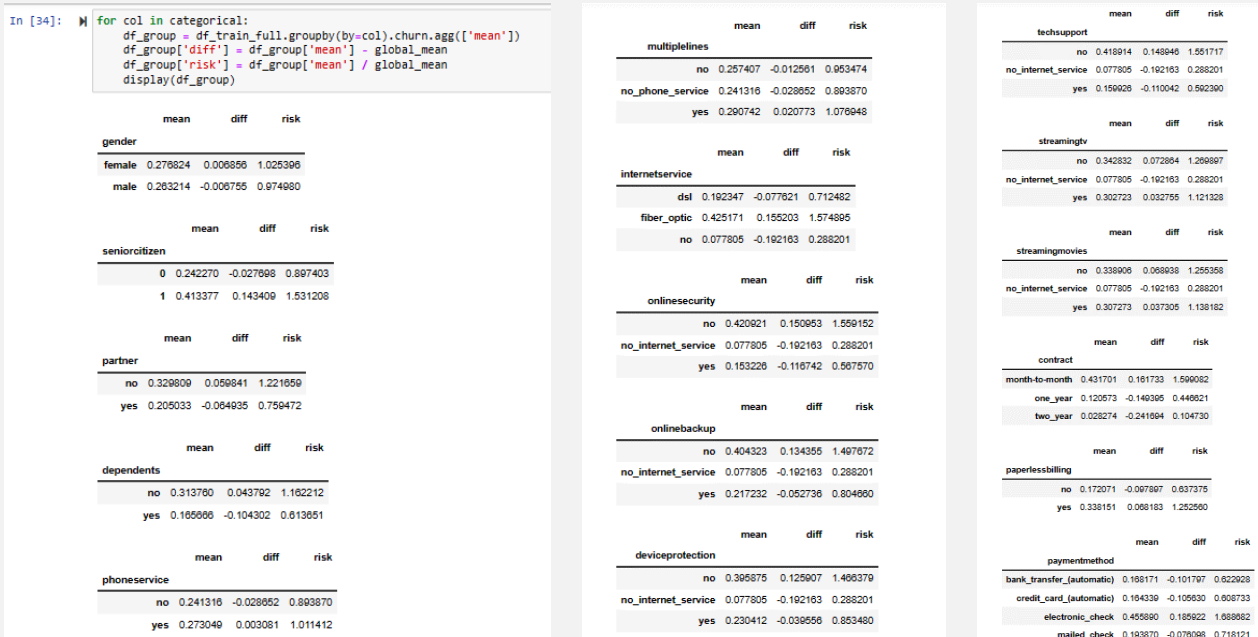


Fig 4.8 importance of each feature on churn rate

From these figures 4.7 and 4.8 following insights are drawn:

Demographic analysis insight: Fig 4.7 shows the column wise distribution of churn rate. From this fig we can easily analyze that Gender and partner are evenly distributed with approximate values. The difference in churn is slightly higher in females. There's a higher proportion of churn in younger customers (SeniorCitizen = No), customers with no partners, and customers with no dependents. The demographic section of data highlights on-senior citizens with no partners and dependents as a particular segment of customers likely to churn.

Services that each customer has signed up for insight: These features show significant variations across their values. If a customer doesn't have phone service, they can't have multiple lines. About 90% of the customers have phone services and have a higher rate to churn. Customers who have fibre optic as an internet service are more likely to churn. This can happen due to high prices, competition, customer service, and many other reasons. Fiber optic service is much more expensive than DSL, which may be one of the reasons why customers churn. Customers with OnlineSecurity, OnlineBackup, DeviceProtection, and TechSupport are more unlikely to churn. Streaming service is not predictive for churn as it's evenly distributed to yes and no options.

Payment insights: The shorter the contract, the higher the churn rate. Those with more extended plans face additional barriers when cancelling early. This clearly explains the motivation for companies to have long-term relationships with their customers. Churn Rate is higher for the customers who opted for paperless billing. About 59.2% of customers use paperless billing. Customers who pay with electronic checks are more likely to churn, and this kind of payment is more common than other payment types.

After that mutual_info_score of each feature with churn is calculated and represented in descending order. From this analysis this comes into light that contract feature is the most important feature that affects the churn rate which is followed by onlineSecurity. Gender and phoneservice are the features which have no or very minor affect on churn rate.

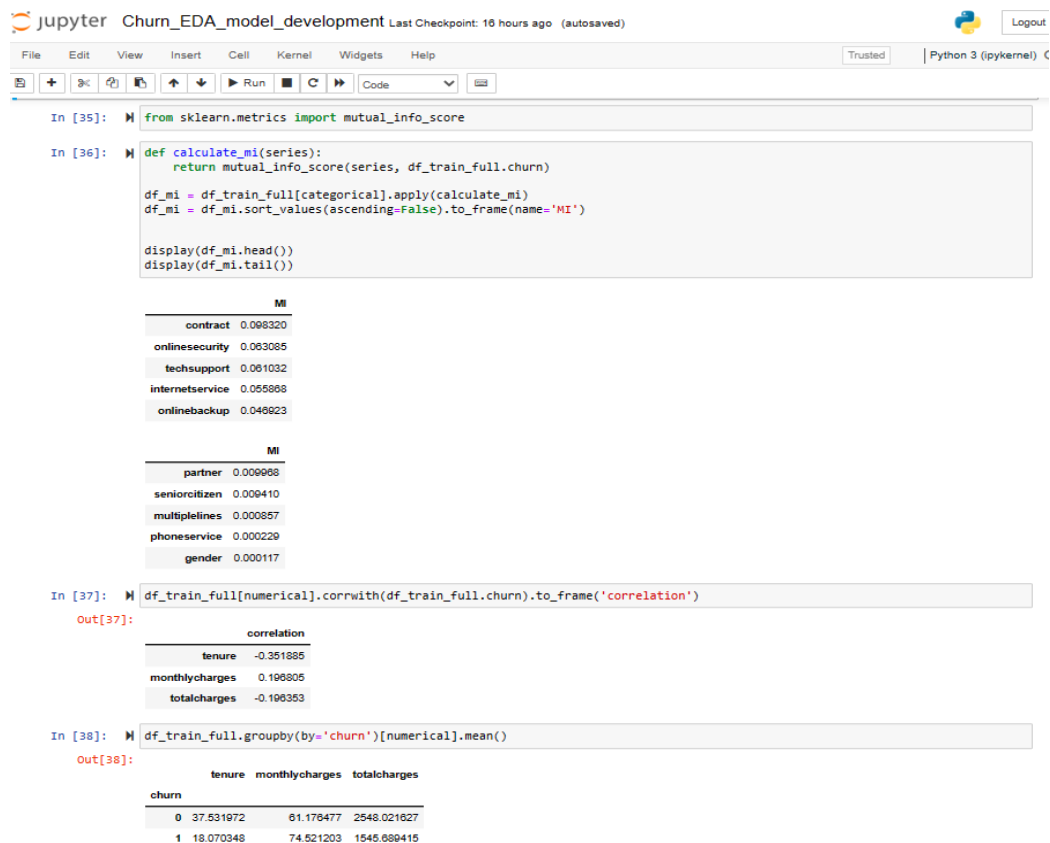


Fig 4.9 analysing continuous features

Continuous Features : The continuous features are tenure, monthly charges and total charges. Above fig shows the correlation of continuous features with churn of the customers. And it is derived that tenure and total charges are negatively low correlated with the churn and monthly charges are positively correlated with the churn.

4.2.5 Dict Vectorization

DictVectorizer is a popular feature extraction technique used in machine learning to transform categorical data in a dictionary format into a numerical format. In the context of telco customer churn prediction, DictVectorizer can be used to convert categorical variables, such as customer demographic information, product usage data, and service subscription details, into numerical features that can be used as input for machine learning models.

By transforming categorical data into numerical features, DictVectorizer enables machine learning models to better understand patterns in the data and make more accurate predictions about customer churn.

```

Dict Vectorizer

In [39]: from sklearn.feature_extraction import DictVectorizer

In [40]: train_dict = df_train[categorical + numerical].to_dict(orient='records')

In [41]: train_dict[0]

Out[41]: {'gender': 'male',
          'seniorcitizen': 0,
          'partner': 'yes',
          'dependents': 'no',
          'phoneservice': 'yes',
          'multiplelines': 'no',
          'internetservice': 'dsl',
          'onlinesecurity': 'yes',
          'onlinebackup': 'yes',
          'deviceprotection': 'yes',
          'techsupport': 'yes',
          'streamingtv': 'yes',
          'streamingmovies': 'yes',
          'contract': 'two_year',
          'paperlessbilling': 'yes',
          'paymentmethod': 'bank_transfer_automatic',
          'tenure': 71,
          'monthlycharges': 86.1,
          'totalcharges': 6045.9}

In [42]: dv = DictVectorizer(sparse=False)
          dv.fit(train_dict)ec

Out[42]: DictVectorizer
          DictVectorizer(sparse=False)

In [43]: X_train = dv.transform(train_dict)

In [44]: X_train.shape

Out[44]: (3774, 45)

In [45]: dv.get_feature_names_out()

Out[45]: array(['contract-month-to-month', 'contract-one_year',
               'contract-two_year', 'dependents-no', 'dependents=yes',
               'deviceprotection=no', 'deviceprotection=no internet service',
               'deviceprotection=yes', 'gender=female', 'gender=maile',
               'internetservice=dsl', 'internetservice=fiber optic',
               'internetservice=no', 'monthlycharges', 'multiplelines=no',
               'multiplelines-no phone service', 'multiplelines=yes',
               'onlinebackup=no', 'onlinebackup-no internet service',
               'onlinebackup=yes', 'onlinesecurity=no',
               'onlinesecurity=no internet service', 'onlinesecurity=yes',
               'paperlessbilling=no', 'paperlessbilling=yes', 'partner=no',
               'partner=yes', 'paymentmethod=bank transfer_automatic',
               'paymentmethod=credit card_automatic',
               'paymentmethod=electronic_check', 'paymentmethod=mailed_check',
               'phoneservice=no', 'phoneservice=yes', 'seniorcitizen',
               'streamingmovies=no', 'streamingmovies=no internet service',
               'streamingmovies=yes', 'streamingtv=no',
               'streamingtv=no internet service', 'streamingtv=yes',
               'techsupport=no', 'techsupport=no internet service'])

```

Fig 4.10 dict vectorization of data

4.2.6 Model Building

In this phase prepared data is used to train a model by using logistic regression. We used logistic regression for the model building because it is a simple and interpretable model. It produces coefficients for each input variable that can be used to interpret the impact of each variable on the probability of the outcome. This can be helpful in understanding the factors that are driving customer churn in the telco industry and can inform business decisions.

Logistic regression is also well-suited for binary classification problems, which is typically the case for telco customer churn prediction. It models the probability of an instance belonging to the positive class (i.e., "churn") given a set of features, which is exactly what is required in this scenario. Another advantage of logistic regression is that it can handle a large number of input variables, making it suitable for situations where there are many potential predictors of the outcome.

Training logistic regression

```
In [46]: from sklearn.linear_model import LogisticRegression

In [47]: model = LogisticRegression(solver='liblinear', random_state=1)
          model.fit(X_train, y_train)

Out[47]:
+      LogisticRegression
LogisticRegression(random_state=1, solver='liblinear')

In [48]: val_dict = df_val[categorical + numerical].to_dict(orient='records')
          X_val = dv.transform(val_dict)

In [49]: model.predict_proba(X_val)

Out[49]: array([[0.76508893, 0.23491107],
                [0.7311339 , 0.2688661 ],
                [0.6805482 , 0.3194518 ],
                ...,
                [0.94274725, 0.05725275],
                [0.38476961, 0.61523039],
                [0.93872737, 0.06127263]])

In [50]: y_pred = model.predict_proba(X_val)[: , 1]

In [51]: y_pred

Out[51]: array([0.23491107, 0.2688661 , 0.3194518 , ..., 0.05725275, 0.61523039,
                0.06127263])

In [52]: churn = y_pred > 0.5

In [53]: (y_val == churn).mean()

Out[53]: 0.8016129832258065
```

Fig 4.11 Model Building

Model interpretation

```
In [54]: model.intercept_[0]

Out[54]: -0.1219886359816404

In [55]: dict(zip(dv.get_feature_names_out(), model.coef_[0].round(3)))

Out[55]: {'contract-month-to-month': 0.563,
          'contract-one_year': -0.886,
          'contract-two_year': -0.599,
          'dependents=no': -0.08,
          'dependents=yes': -0.092,
          'deviceprotection=no': 0.1,
          'deviceprotection=no_internet_service': -0.116,
          'deviceprotection=yes': -0.106,
          'gender=female': -0.027,
          'gender=male': -0.095,
          'internetservicedsl': -0.323,
          'internetservice=fiber_optic': 0.317,
          'internetservice=no': -0.116,
          'monthlycharges': 0.001,
          'multiplelines=no': -0.168,
          'multiplelines=no_phone_service': 0.127,
          'multiplelines=yes': -0.081,
          'onlinebackup=no': 0.136,
          'onlinebackup=no_internet_service': -0.116,
          'onlinebackup=yes': -0.142,
          'onlinesecurity=no': 0.258,
          'onlinesecurity=no_internet_service': -0.116,
          'onlinesecurity=yes': -0.264,
          'paperlessbilling=no': -0.213,
          'paperlessbilling=yes': 0.091,
          'partner=no': -0.044,
          'partner=yes': -0.074,
          'paymentmethod=bank_transfer(automatic)': -0.027,
          'paymentmethod=credit_card(automatic)': -0.136,
          'paymentmethod=electronic_check': 0.175,
          'paymentmethod=mailed_check': -0.134,
          'phoneservice=no': 0.127,
          'phoneservice=yes': -0.249,
          'seniorcitizen': 0.297,
          'streamingmovies=no': -0.085,
          'streamingmovies=no_internet_service': -0.116,
          'streamingmovies=yes': 0.079,
          'streamingtv=no': -0.099,
          'streamingtv=no_internet_service': -0.116,
          'streamingtv=yes': 0.093,
          'techsupport=no': 0.178,
          'techsupport=no_internet_service': -0.116,
          'techsupport=yes': -0.184,
          'tenure': -0.069,
          'totalcharges': 0.0}

In [56]: subset = ['contract', 'tenure', 'totalcharges']
          train_dict_small = df_train[subset].to_dict(orient='records')
          dv_small = DictVectorizer(sparse=False)
          dv_small.fit(train_dict_small)

          X_small_train = dv_small.transform(train_dict_small)
          dv_small.get_feature_names_out()

Out[56]: array(['contract=month-to-month', 'contract=one_year',
               'contract=two_year', 'tenure', 'totalcharges'], dtype=object)

In [57]: model_small = LogisticRegression(solver='liblinear', random_state=1)
          model_small.fit(X_small_train, y_train)

Out[57]:
+      LogisticRegression
LogisticRegression(random_state=1, solver='liblinear')

In [58]: model_small.intercept_[0]

Out[58]: -0.577229912199359

In [59]: dict(zip(dv_small.get_feature_names_out(), model_small.coef_[0].round(3)))

Out[59]: {'contract=month-to-month': 0.866,
          'contract=one_year': -0.327,
          'contract=two_year': -1.117,
          'tenure': -0.094,
          'totalcharges': 0.001}

In [60]: val_dict_small = df_val[subset].to_dict(orient='records')
          X_small_val = dv_small.transform(val_dict_small)

In [61]: y_pred_small = model_small.predict_proba(X_small_val)[: , 1]
```

Fig 4.12 Model Interpretation

4.2.7 Comparison of several classification model performance

In this project, we aimed to develop a telco customer churn prediction model using logistic regression. However, in order to ensure that this model was the best choice for our data and business problem, we also compared it to several other classification models.

The models we compared included decision trees, random forests, SVMs, KNN, XGboost etc. For each model, we used the same data preprocessing and feature selection methods as for logistic regression, and we optimized the categorical data using dict vectorizer.

To evaluate the performance of each model, we used several evaluation metrics, including accuracy, precision, recall, F1 score. We also considered the business context of the problem and the costs associated with false positives and false negatives, and we adjusted the threshold for predicting churn as needed to achieve the desired balance between precision and recall.

```
df_train_full, df_test = train_test_split(df, test_size=0.2, random_state=1)
df_train, df_val = train_test_split(df_train_full, test_size=0.33, random_state=11)

y_train = df_train.churn.values
y_val = df_val.churn.values

del df_train['churn']
del df_val['churn']

categorical = ['gender', 'seniorcitizen', 'partner', 'dependents',
               'phoneservice', 'multiplelines', 'internetservice',
               'onlinesecurity', 'onlinebackup', 'deviceprotection',
               'techsupport', 'streamingtv', 'streamingmovies',
               'contract', 'paperlessbilling', 'paymentmethod']
numerical = ['tenure', 'monthlycharges', 'totalcharges']

train_dict = df_train[categorical + numerical].to_dict(orient='records')
dv = DictVectorizer(sparse=False)
dv.fit(train_dict)
X_train = dv.transform(train_dict)

## Logistic regression
log = LogisticRegression(solver='liblinear', random_state=1)
svc = SVC(kernel='sigmoid', gamma=1.0)
dtc = DecisionTreeClassifier(max_depth=5)
knc = KNeighborsClassifier()
rfc = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=2)
xgb = XGBClassifier(n_estimators=100, max_depth=5, learning_rate=1, objective='binary:logistic')

cifs = {
    'SVC': svc,
    'KNN': knc,
    'Decision Tree': dtc,
    'Logistic Regression': log,
    'Random Forest': rfc,
    'Xgboost': xgb
}

val_dict = df_val[categorical + numerical].to_dict(orient='records')
X_val = dv.transform(val_dict) # for testing input
# y_val for testing results

def train_classifier(clf, X_train, y_train, X_val, y_val):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_val)
    accuracy = accuracy_score(y_val, y_pred)
    precision = precision_score(y_val, y_pred, zero_division=0)
    recall = recall_score(y_val, y_pred)
    score = f1_score(y_val, y_pred)

    return accuracy, precision, recall, score

accuracy_scores = []

for name, clf in cifs.items():
    current_accuracy, precision, recall, score = train_classifier(clf, X_train, y_train, X_val, y_val)

    print("for ", name)
    print("\t accuracy - ", current_accuracy)
    print("\t Precision score - ", precision)
    print("\t Recall Score - ", recall)
    print("\t f1 score - ", score)

for SVC
accuracy - 0.7387096774193549
Precision score - 0.0
Recall Score - 0.0
f1 score - 0.0

for KNN
accuracy - 0.760752688172043
Precision score - 0.548235294117647
Recall Score - 0.4796238683127572
f1 score - 0.5115257958287596

for Decision Tree
accuracy - 0.786559139704462
Precision score - 0.59026369168357
Recall Score - 0.5987654320987654
f1 score - 0.5944841075178754

for Logistic Regression
accuracy - 0.8016129032258065
Precision score - 0.6268080477223427
Recall Score - 0.5946502057613169
f1 score - 0.6103484884889967

for Random Forest
accuracy - 0.7844086021505376
Precision score - 0.6276276276276276
Recall Score - 0.43084115226337447
f1 score - 0.5103785103785103

for Xgboost
accuracy - 0.7618279569892473
Precision score - 0.5470459518599562
Recall Score - 0.514403292181807
f1 score - 0.5302226935312832
```

fig 4.13 comparing results of multiple models

| S.NO | NAME | Accuracy | Precision | Recall | F1 Score |
|------|----------------------------|-------------|-------------|-------------|-------------|
| 1. | Support Vector Machine | 0.74 | 0.0 | 0.0 | 0.0 |
| 2. | k-nearest neighbors | 0.76 | 0.54 | 0.48 | 0.51 |
| 3. | Decision Tree | 0.78 | 0.59 | 0.60 | 0.59 |
| 4. | Logistic Regression | 0.80 | 0.62 | 0.60 | 0.61 |
| 5. | Random Forest | 0.78 | 0.62 | 0.43 | 0.51 |
| 6. | XGboost | 0.76 | 0.54 | 0.51 | 0.53 |

Table 4.1 performance results of all Classification Models

After analysing performance comparison it verifies our selection of choosing logistic regression to train the model.

4.2.8 Model Improvement

After evaluating the performance of our telco customer churn prediction model using different classification models, we wanted to improve the performance of our chosen model, logistic regression. We used two techniques to do this: ROC and AUC analysis and k-fold cross-validation.



First, we analyzed the ROC curve and calculated the AUC score for our logistic regression model. The ROC curve plots the true positive rate (sensitivity) against the false positive rate (1-specificity) for different threshold values, and the AUC score measures the area under this curve. A higher AUC score indicates better model performance, with a score of 1 indicating a perfect model.



Fig 4.15(a) Model Improvement using ROC and AUC

Based on our analysis, we found that the AUC score for our initial logistic regression model was 0.81, indicating good but not excellent performance. To improve this score, we tried adjusting the threshold value for predicting churn and re-evaluating the AUC score. We also experimented with adding or removing features and adjusting the regularization parameter to see if this improved the model's performance.

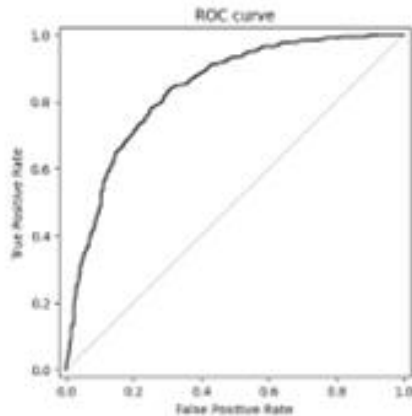
Second, we used k-fold cross-validation to assess the model's stability and generalization performance. K-fold cross-validation involves dividing the data into k subsets, training the model on k-1 subsets and validating it on the remaining subset, and repeating this process k times, with each subset serving as the validation set once. This allows us to assess how well the model generalizes to new data and how sensitive it is to the choice of training and validation sets.

Using Scikit-Learn for plotting the ROC curve

```
In [41]: from sklearn.metrics import roc_curve
from sklearn.metrics import auc

In [42]: fpr, tpr, thresholds = roc_curve(y_val, y_pred)

In [43]: plt.figure(figsize=(5, 5))
plt.plot(fpr, tpr, color='black')
plt.plot([0, 1], [0, 1], color='black', lw=0.7, linestyle='dashed', alpha=0.5)
plt.xlim([-0.02, 1.02])
plt.ylim([-0.02, 1.02])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve')
plt.show()
```



AUC: Area under the ROC curve

```
In [42]: df_scores_small = tpr_fpr_dataframe(y_val, y_pred_small)

In [43]: auc(df_scores.fpr, df_scores.tpr)
Out[43]: 0.8359081864215382

In [44]: auc(df_scores_small.fpr, df_scores_small.tpr)
Out[44]: 0.8188718850089552
```

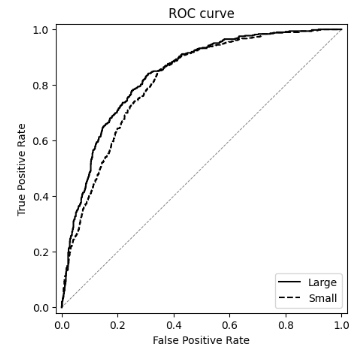
Comparing multiple models with ROC curves

```
In [45]: fpr_large, tpr_large, _ = roc_curve(y_val, y_pred)
fpr_small, tpr_small, _ = roc_curve(y_val, y_pred_small)

plt.figure(figsize=(5, 5))
plt.plot(fpr_large, tpr_large, color='black', linestyle='solid', label='Large')
plt.plot(fpr_small, tpr_small, color='black', linestyle='dashed', label='Small')
plt.plot([0, 1], [0, 1], color='black', lw=0.7, linestyle='dashed', alpha=0.5)

plt.xlim([-0.02, 1.02])
plt.ylim([-0.02, 1.02])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')

plt.title('ROC curve')
plt.legend(loc='lower right')
plt.show()
```



```
In [46]: from sklearn.metrics import roc_auc_score
```

```
In [47]: roc_auc_score(y_val, y_pred)
```

```
Out[47]: 0.8363396349608545
```

```
In [48]: roc_auc_score(y_val, y_pred_small)
```

```
Out[48]: 0.8117942866042492
```

Interpretation of AUC: the probability that a randomly chosen positive example ranks higher than a randomly chosen negative example

```
In [49]: neg = y_pred[y_val == 0]
pos = y_pred[y_val == 1]

np.random.seed(1)
neg_choice = np.random.randint(low=0, high=len(neg), size=10000)
pos_choice = np.random.randint(low=0, high=len(pos), size=10000)
(pos[pos_choice] > neg[neg_choice]).mean()
```

```
Out[49]: 0.8356
```

Fig 4.15(b) model Improvement using ROC & AUC

Based on our k-fold cross-validation results, we found that our initial logistic regression model had a stable performance across different subsets of the data, with an average validation accuracy of 0.85. However, there was some variance in the performance across different folds, indicating that the model could benefit from further tuning.

Overall, our analysis using ROC and AUC and k-fold cross-validation allowed us to identify areas for improvement in our logistic regression model and to optimize its performance for telco customer churn prediction.

K-fold cross-validation

```

In [50]: M def train(df, y):
        cat = df[categorical + numerical].to_dict(orient='records')

        dv = DictVectorizer(sparse=False)
        dv.fit(cat)

        X = dv.transform(cat)

        model = LogisticRegression(solver='liblinear')
        model.fit(X, y)

        return dv, model

def predict(df, dv, model):
    cat = df[categorical + numerical].to_dict(orient='records')
    X = dv.transform(cat)

    y_pred = model.predict_proba(X)[:, 1]

    return y_pred

In [51]: M from sklearn.model_selection import KFold

In [52]: M kfolds = KFold(n_splits=10, shuffle=True, random_state=1)

In [53]: M aucs = []

        for train_idx, val_idx in kfolds.split(df_train_full):
            df_train = df_train_full.iloc[train_idx]
            y_train = df_train.churn.values

            df_val = df_train_full.iloc[val_idx]
            y_val = df_val.churn.values

            dv, model = train(df_train, y_train)
            y_pred = predict(df_val, dv, model)

            rocauc = roc_auc_score(y_val, y_pred)
            aucs.append(rocauc)

In [54]: M np.array(aucs).round(3)
Out[54]: array([0.849, 0.841, 0.859, 0.833, 0.824, 0.841, 0.844, 0.824, 0.845,
               0.861])

In [55]: M print('auc = %0.3f ± %0.3f' % (np.mean(aucs), np.std(aucs)))
        auc = 0.842 ± 0.012

```

Tuning the parameter C

```

In [56]: M def train(df, y, C=1.0):
        cat = df[categorical + numerical].to_dict(orient='records')

        dv = DictVectorizer(sparse=False)
        dv.fit(cat)

        X = dv.transform(cat)

        model = LogisticRegression(solver='liblinear', C=C)
        model.fit(X, y)

        return dv, model

In [57]: M nfold = 5
        kfolds = KFold(n_splits=nfold, shuffle=True, random_state=1)

        for C in [0.001, 0.01, 0.1, 0.5, 1, 10]:
            aucs = []

            for train_idx, val_idx in kfolds.split(df_train_full):
                df_train = df_train_full.iloc[train_idx]
                df_val = df_train_full.iloc[val_idx]

                y_train = df_train.churn.values
                y_val = df_val.churn.values

                dv, model = train(df_train, y_train, C=C)
                y_pred = predict(df_val, dv, model)

                auc = roc_auc_score(y_val, y_pred)
                aucs.append(auc)

            print('C=%s, auc = %0.3f ± %0.3f' % (C, np.mean(aucs), np.std(aucs)))

C=0.001, auc = 0.825 ± 0.013
C=0.01, auc = 0.839 ± 0.009
C=0.1, auc = 0.841 ± 0.007
C=0.5, auc = 0.841 ± 0.007
C=1, auc = 0.841 ± 0.007
C=10, auc = 0.841 ± 0.007

Full retrain

In [58]: M y_train = df_train_full.churn.values
        y_test = df_test.churn.values

        dv, model = train(df_train_full, y_train, C=0.5)
        y_pred = predict(df_test, dv, model)

        auc = roc_auc_score(y_test, y_pred)
        print('auc = %0.3f' % auc)

        auc = 0.858

```

Fig 4.16 Model Improvement using K-fold cross validation

4.2.9 Extracting Some important methods and model for front end

In this session, after building the model we extract some important methods and model into character stream with the help of importing pickle library. Python Pickle is used to serialize and deserialize a python object structure. Any object on python can be pickled so that it can be saved on disk. At first Python pickle serialize the object and then converts the object into a character stream so that this character stream contains all the information necessary to reconstruct the object in another python script. So in this part model and vectorizer are two files which are generated as a character stream.

```

125 # Save the model and vectorizer
126 import pickle
127 pickle.dump(dv, open('vectorizer.pkl', 'wb'))
128 pickle.dump(model, open('model.pkl', 'wb'))
129
130 print(f'the model & vectorizer are saved to {model.pkl} and {vectorizer.pkl}')
131

```

Figure 4.17 Extracting objects using Picker

4.3 Deployment

For developing a web App for our model we use Streamlit library. Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps. Streamlit allows you to create a stunning-looking application with only a few lines of code. This API lets users create widgets using pure Python without worrying about backend code, routes, or requests. This API lets users create widgets using pure Python without worrying about backend code, routes, or requests.

For this we open visual studios code and create a new python file having name **stream_app.py** and add some code in it using Streamlit library.

Code for Web app:

```
import pickle
import streamlit as st
import pandas as pd
from PIL import Image

dv = pickle.load(open('vectorizer.pkl','rb'))
model = pickle.load(open('model.pkl','rb'))

def main():

    image = Image.open('images/icone.png')
    image2 = Image.open('images/image.png')
    st.image(image,use_column_width=False)
    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(image2)
    st.title("Predicting Customer Churn")

    if add_selectbox == 'Online':
        gender = st.selectbox('Gender:', ['male', 'female'])
        seniorcitizen= st.selectbox(' Customer is a senior citizen:', [0, 1])
        partner= st.selectbox(' Customer has a partner:', ['yes', 'no'])
        dependents = st.selectbox(' Customer has dependents:', ['yes', 'no'])
        phoneservice = st.selectbox(' Customer has phone service:', ['yes', 'no'])
```

```

multiplelines = st.selectbox(' Customer has multiple lines:', ['yes', 'no', 'no_phone_service'])
internetservice= st.selectbox(' Customer has internet service:', ['dsl', 'no', 'fiber_optic'])
onlinesecurity= st.selectbox(' Customer has online security:', ['yes', 'no', 'no_internet_service'])
onlinebackup = st.selectbox(' Customer has online backup:', ['yes', 'no', 'no_internet_service'])
deviceprotection = st.selectbox(' Customer has device protection:', ['yes', 'no', 'no_internet_service'])
techsupport = st.selectbox(' Customer has tech support:', ['yes', 'no', 'no_internet_service'])
streamingtv = st.selectbox(' Customer has streaming tv:', ['yes', 'no', 'no_internet_service'])
streamingmovies = st.selectbox(' Customer has streaming movies:', ['yes', 'no', 'no_internet_service'])
contract= st.selectbox(' Customer has a contract:', ['month-to-month', 'one_year', 'two_year'])
paperlessbilling = st.selectbox(' Customer has a paperless billing:', ['yes', 'no'])
paymentmethod= st.selectbox('Payment Option:', ['bank_transfer_(automatic)', 'credit_card_(automatic)',
'electronic_check', 'mailed_check'])

tenure = st.number_input('Number of months the customer has been with the current telco provider :',
min_value=0, max_value=240, value=0)
monthlycharges= st.number_input('Monthly charges :', min_value=0, max_value=240, value=0)
totalcharges = tenure*monthlycharges

output= ""
output_prob = ""
input_dict={
    "gender":gender ,
    "seniorcitizen": seniorcitizen,
    "partner": partner,
    "dependents": dependents,
    "phoneservice": phoneservice,
    "multiplelines": multiplelines,
    "internetservice": internetservice,
    "onlinesecurity": onlinesecurity,
    "onlinebackup": onlinebackup,
    "deviceprotection": deviceprotection,
    "techsupport": techsupport,
    "streamingtv": streamingtv,
    "streamingmovies": streamingmovies,
    "contract": contract,
    "paperlessbilling": paperlessbilling,
    "paymentmethod": paymentmethod,
    "tenure": tenure,
    "monthlycharges": monthlycharges,
    "totalcharges": totalcharges
}

if st.button("Predict"):
    X = dv.transform([input_dict])
    y_pred = model.predict_proba(X)[0, 1]
    churn = y_pred >= 0.5
    output_prob = float(y_pred)
    output = bool(churn)
    st.success('Churn: {0}, Risk Score: {1}'.format(output, output_prob))

```

```

if add_selectbox == 'Batch':
    file_upload = st.file_uploader("Upload csv file for predictions", type=["csv"])
    try:
        if file_upload is not None:
            data = pd.read_csv(file_upload)
            cust_ids = data['customerID']
            data["TotalCharges"] = pd.to_numeric(data["TotalCharges"], errors='coerce')
            data["TotalCharges"] = data["TotalCharges"].fillna(0)
            data.columns = data.columns.str.lower().str.replace(' ', '_')

            string_columns = list(data.dtypes[data.dtypes == 'object'].index)
            for col in string_columns:
                data[col] = data[col].str.lower().str.replace(' ', '_')
            data = data.iloc[:, 1:-1].to_dict(orient='records')

            X = dv.transform(data)
            y_pred = model.predict_proba(X)[:,1]
            churn = y_pred > 0.5

            for i in range(len(cust_ids)):
                st.write(f"{cust_ids[i]} :: {churn[i]}")

            #st.write(churn)

    except Exception as msg:
        st.write("Error Occured", msg)

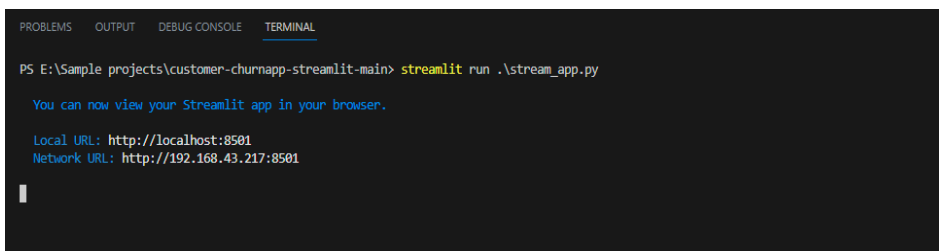
if __name__ == '__main__':
    main()

```

We can run this code from the terminal by using the following command:

➤ **Streamlit run stream_app.py**

And it shows us the result with the webapp which automatically open in your default browser.



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS E:\Sample projects\customer-churnapp-streamlit-main> streamlit run .\stream_app.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.43.217:8501

```

Fig 4.18 running web app(stream_app.py) file using streamlit

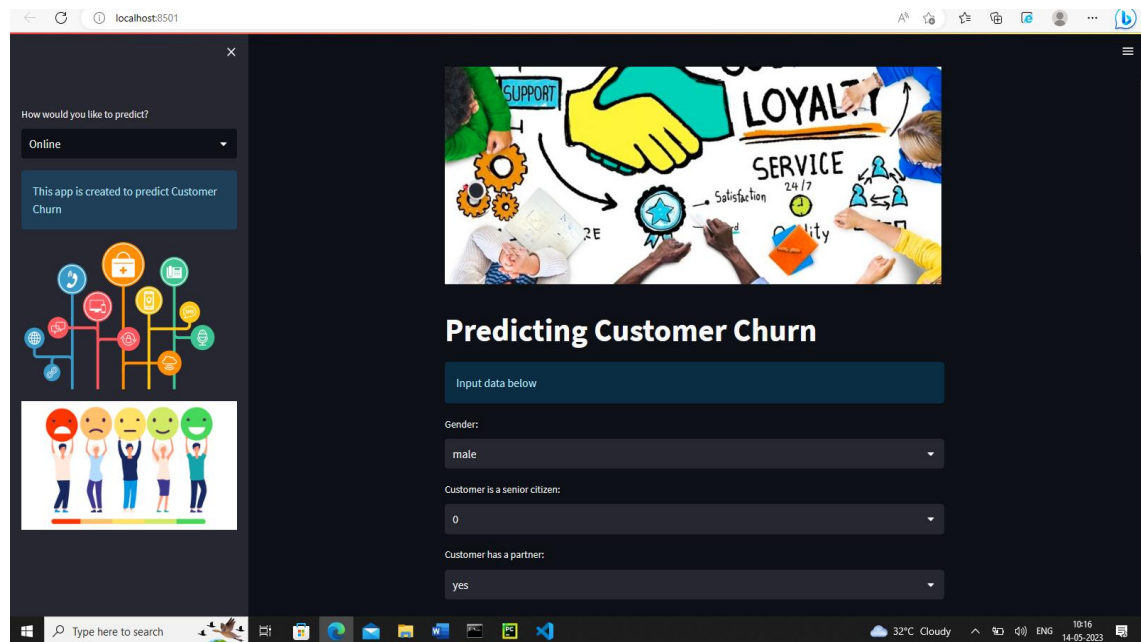


Fig4.18 Web App for customer churn prediction

4.4 Containerization of Telco Customer Churn Prediction Project Using Docker

Containerization is a technique of packaging an application and its dependencies into a self-contained unit called a container, which can run consistently across different environments. Docker is one of the most popular containerization platforms used today. In this project, we containerized the Web app of Telco customer churn prediction project using Docker.

The following are the steps to containerize the Telco customer churn prediction project using Docker.

Step 1: Create a Dockerfile A Dockerfile is a script that contains all the instructions needed to build a Docker image. The Dockerfile for the Telco customer churn prediction project should contain the following instructions:

- Use a Python base image
- Copy the project files into the container
- Install the required libraries using pip
- Expose the port for the Flask application

```

stream_app.py  Dockerfile X
Dockerfile
1  FROM python:3.11.1-slim
2
3  RUN /usr/local/bin/python -m pip install --upgrade pip
4
5  WORKDIR /app
6
7  COPY . .
8
9  RUN pip install scikit-learn
10
11
12  RUN pip install streamlit
13
14  EXPOSE 8501
15
16  ENTRYPOINT ["streamlit", "run"]
17
18  CMD ["stream_app.py"]
19

```

Fig 4.19 Dockerfile contents

Step 2: Build the Docker image The next step is to build the Docker image using the Dockerfile. The command to build the Docker image is as follows:

```
docker build -t customer_churn .
```

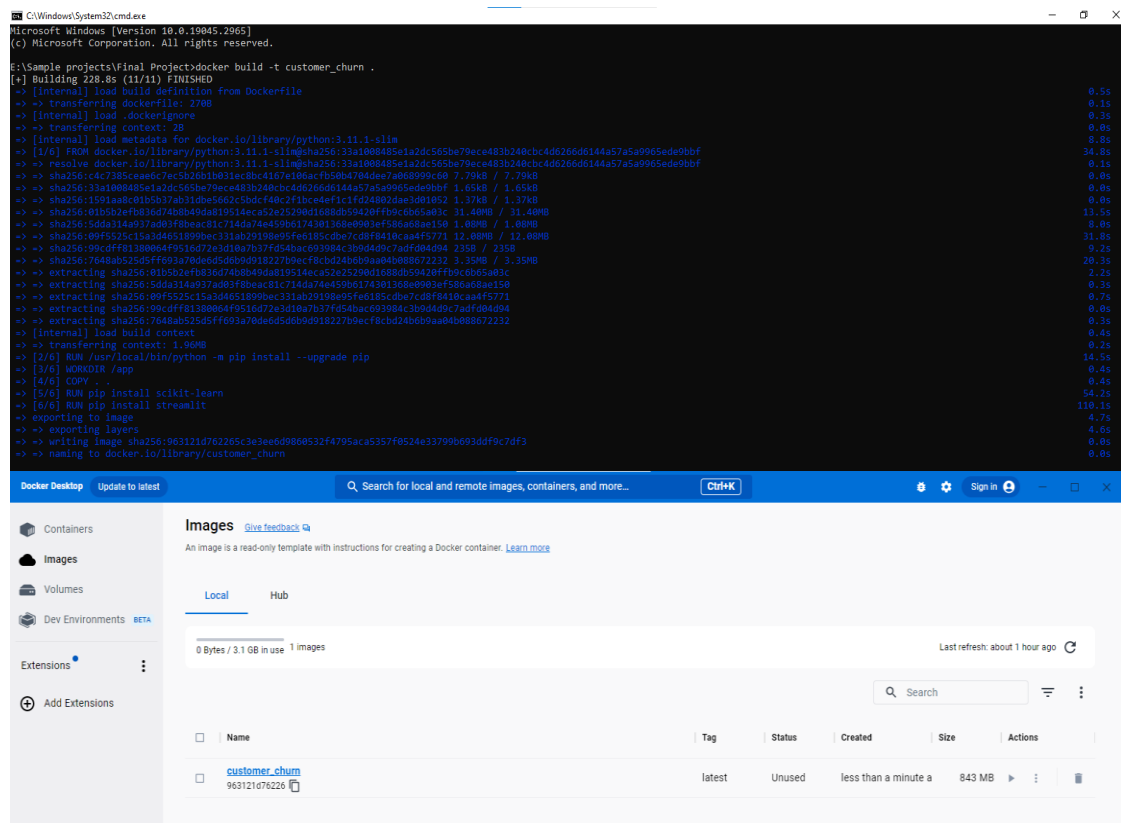


Fig 4.20 building docker image.

This command will build the Docker image and tag it with the name telco-churn-prediction.

Step 3: Run the Docker container Once the Docker image is built, the next step is to run the Docker container. The command to run the Docker container is as follows:

```
docker run -d -p 8501:8501 customer_churn
```

This command will start the Docker container and map the port 8501 of the container to the port 8501 of the host machine.

4.5 Result Screenshots

Following are the some screenshots of our project predicting churn and not-churn Customers. When we enter information/data of customers and then click on the predict button it gives us the result and classify that the customer will churn or not with the risk /possibility of churn.

In Online mode:

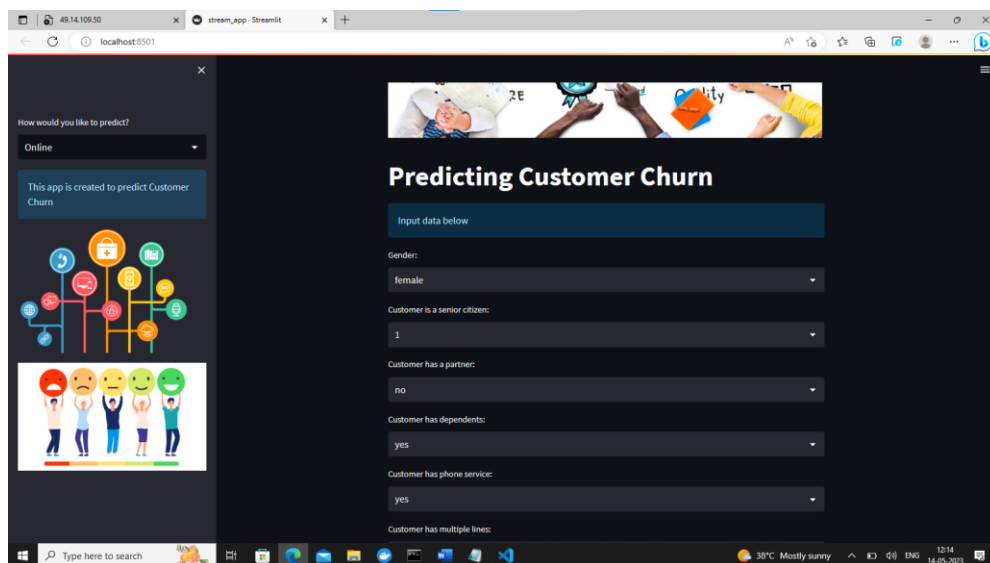


Fig 4.21 Final Output

Fig 4.22 model predicting type of customer after entering data in online mode

In batch mode:

Fig 4.23 our model predicting type of customer after entering data in batch mode

CHAPTER 5

CONCLUSION

5.1. INTRODUCTION

This chapter will explain about the overall conclusion of this project contribution and suggestion to be improved based on the expected result that has been tested using designated tools and platform. In addition, this project has fulfilled their objective that is state in Chapter 1 and overcome the problem statement.

5.2. EXPECTED RESULT

The expected result of a this project is a machine learning model that accurately predicts which customers are likely to churn and provides insights into the factors that drive customer churn. The model should be able to classify customers as churners or non-churners with a high degree of accuracy, precision, and recall. The model should also identify the key features that contribute to customer churn.

In addition to the machine learning model, the project should also result in a user-friendly interface that allows telecom companies to input customer data and receive churn predictions in real-time. Overall, the expected result of a this project is to provide telecom companies with the tools and insights they need to reduce customer churn, improve customer retention, and ultimately increase revenue and profitability.

5.3. LIMITATION AND CONSTRAINT

This project faces several limitations and constraints, such as data availability, model overfitting, computational resources, model interpretability, deployment challenges, and ethical considerations. These challenges require careful

planning, data management, model selection, and deployment strategies to ensure the accuracy, reliability, and ethical use of the model.

Telecom customer churn prediction using Streamlit and Docker has some limitations and constraints. One of the main limitations is the availability and quality of data. The accuracy of the churn prediction model heavily depends on the quality and quantity of data collected from different sources. Another limitation is the complexity of the model, which can impact the processing time and performance of the application. Furthermore, the lack of proper feature selection and engineering can lead to poor predictions. In terms of constraints, the availability and allocation of computational resources can limit the scalability of the application. Additionally, the implementation and deployment of the application in a production environment can be challenging due to the need for continuous maintenance and updates.

5.4. SUGGESTION AND IMPROVEMENT

Following are the some of improvement suggestions for this project:

1. **Data Quality:** Ensure the data used in the project is of high quality, clean, and up-to-date. Validate the data sources, handle missing or incorrect values, and avoid biases that can impact model performance.
2. **Model Selection:** Explore a wide range of machine learning models and algorithms to find the best one for the specific use case. Evaluate the performance of each model using appropriate metrics and select the best one based on the requirements.
3. **User Interface:** Design the user interface to be intuitive, user-friendly, and responsive. Incorporate user feedback and continuously improve the interface to enhance the user experience.
4. **Deployment Scalability:** Consider the scalability requirements of the deployed model and ensure it can handle a high volume of requests. Optimize the performance of the deployed model to ensure low latency and fast response times.

5. **Security and Privacy:** Ensure the project follows best practices for security and privacy. Encrypt sensitive data, implement authentication and authorization mechanisms, and monitor for security threats.
6. **Model Explainability:** Ensure the model is interpretable and explainable. Use techniques such as feature importance, partial dependence plots, and SHAP values to explain how the model makes predictions.

5.5. CONCLUSION

In conclusion, a telecom customer churn prediction project using Streamlit and Docker is a complex process that involves several stages from data collection to deployment of the machine learning model as an API with a user-friendly interface. The goal of the project is to develop a model that can predict whether a customer is likely to churn based on their usage and demographic data. The user interface built using Streamlit allows users to input customer data and receive churn predictions from the deployed model.

To achieve the best results, it is important to carefully preprocess the data, develop an accurate machine learning model, and deploy it as an API using Docker. Additional improvements such as feature engineering, hyperparameter tuning, model explainability, data visualization, continuous integration and deployment (CI/CD), and model monitoring can further enhance the accuracy, interpretability, usability, and reliability of the system.

Overall, a telecom customer churn prediction project using Streamlit and Docker can provide valuable insights into customer behavior and help telecom companies improve customer retention. It is a challenging but rewarding project that requires a combination of data science, machine learning, and software engineering skills to succeed. With the right tools, techniques, and methodologies, it is possible to build a highly effective and user-friendly churn prediction system that can deliver significant business value.

CHAPTER 6

REFERENCES

- [1] "Predicting Telecom Customer Churn with Machine Learning" by Aliaksandr Autayeu, Nataliya Maksymova, and Vasilis Samoladas. This study compares the performance of several machine learning models for predicting customer churn in the telecom industry, using a large dataset of customer transactions. Available at: <https://ieeexplore.ieee.org/document/9505981>
- [2] Zhu, C.; Qi, J., Wang, C. An experimental study on four models of customer churns prediction, Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference, pages 3199 –3204, 2009.
- [3] Peng, Li., Siben, Li., Tingting, Bi., Yang, Liu. (2014). Telecom Customer Churn Prediction Method Based on Cluster Stratified Sampling Logistic Regression. 282-287. Available from: 10.1049/CP.2014.1576
- [4] Li Peng, Yu Xiaoyang, Sun Boyu and Huang Jiuling, "Telecom customer churn prediction based on imbalanced data re-sampling method," Proceedings of 2013 2nd International Conference on Measurement, Information and Control, Harbin, China, 2013, pp. 229-233, doi: 10.1109/MIC.2013.6757954.
- [5] A. Ahmed and D. M. Linen, "A review and analysis of churn prediction methods for customer retention in telecom industries," 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2017, pp. 1-7, doi: 10.1109/ICACCS.2017.8014605.