CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Customer churn is when a company's customers stop doing business with that company. Businesses are very keen on measuring churn because keeping an existing customer is far less expensive than acquiring a new customer. New business involves working leads through a sales funnel, using marketing and sales budgets to gain additional customers. Existing customers will often have a higher volume of service consumption and can generate additional customer referrals.

Customer retention can be achieved with good customer service and products. But the most effective way for a company to prevent attrition of customers is to truly know them. The vast volumes of data collected about customers can be used to build churn prediction models. Knowing who is most likely to defect means that a company can priority focused marketing efforts on that subset of their customer base.

Preventing customer churn is critically important to the telecommunications sector, as the barriers to entry for switching services are so low.

1.2 OBJECTIVES OF THE PROJECT

- **Identify At-Risk Customers:** By analyzing customer data and identifying patterns, businesses can proactively address the needs of these at-risk customers before they decide to leave.
- **Understand Churn Drivers:** Determining the underlying factors that contribute to customer churn is crucial. This involves analyzing customer behaviour, feedback, and engagement. Understanding these drivers enables businesses to make informed decisions on how to improve their products or services.
- **Develop Targeted Retention Strategies:** Based on the insights gained from churn prediction models, businesses can design and implement specific strategies aimed at retaining at-risk customers. This could include personalized offers, improved customer

support, and tailored marketing campaigns to address the unique needs and concerns of these customers.

- **Reduce Customer Acquisition Costs:** By focusing on retaining existing customers, businesses can significantly reduce the costs associated with acquiring new ones. Effective churn management ensures that marketing and sales efforts are more efficient, thereby lowering overall customer acquisition costs.
- **Enhance Overall Customer Satisfaction and Loyalty:** Addressing the factors that lead to churn can improve the overall customer experience. By continuously refining products and services based on customer feedback and behaviour, businesses can enhance customer satisfaction and build long-term loyalty.

1.3 SCOPE OF THE PROJECT

- **Data Collection and Integration:** Gather and unify customer data from CRM systems, analytics services, and feedback platforms into a centralized database.
- **Feature Engineering and Selection:** Identify and engineer key features such as customer demographics, transaction history, and usage patterns to improve model accuracy.
- **Model Development and Validation:** Develop and validate machine learning models using historical data to predict customer churn, ensuring high performance through metrics like accuracy and recall.
- **Deployment and Monitoring:** Implement the churn prediction model in real-time systems and continuously monitor its performance to maintain accuracy.
- **Actionable Insights and Strategies:** Utilize model insights to inform retention strategies, customer engagement initiatives, and optimize customer service to reduce churn rates.

1.4 APPLICATIONS

- **Targeted Retention Campaigns:** By predicting which customers are at risk of churning, companies can launch targeted retention campaigns to address these customers' specific needs and concerns.
- **Personalized Customer Interactions:** Customer service representatives can use churn predictions to tailor their interactions and offer personalized solutions or promotions to at-risk customers.

- **Optimized Marketing Strategies:** Insights from churn models can help businesses understand why customers leave and adjust their marketing strategies to improve customer satisfaction and retention.
- **Product Improvement:** Churn analysis helps in identifying the features or services that lead to customer dissatisfaction, allowing businesses to make informed improvements to their offerings.
- **Revenue Forecasting:** Accurate churn predictions enable better revenue forecasting by providing insights into potential future losses and helping to mitigate them through proactive strategies.

CHAPTER 2

LITERATURE SURVEY

- **Chen and Ann Huang**
  Label encoding and one-hot encoding are standard techniques. Label encoding assigns a unique integer to each category, while one-hot encoding creates binary columns for each category.

- **Jiawei Han**
  Standardization and normalization are used to bring numerical features to a similar scale, which is crucial for models like SVC and KNN.

- **Annelies Verbeke**
  Methods such as mean imputation, median imputation, and predictive imputation are commonly used. The impact of different imputation techniques on model performance.

- **Lemmens and Croux**
  Decision trees provide an interpretable model structure, while random forests offer improved accuracy by averaging multiple trees. The advantages of ensemble methods like random forests in churn prediction.

- **Coussement and Van den Poel**
  SVC is effective in high-dimensional spaces and robust to overfitting. Shows that SVC can achieve high accuracy in churn prediction tasks.

- **Idris**
  KNN is a simple, instance-based learning algorithm. Demonstrate its application in churn prediction, highlighting its sensitivity to the choice of k and distance metrics.

CHAPTER 3

REQUIREMENTS SPECIFICATIONS

3.1 HARDWARE REQUIREMENTS

Processor (CPU):

- Recommendation: Intel Core i7 or i9, AMD Ryzen 7 or 9.
- Rationale: These processors offer high performance necessary for data processing and running complex machine learning algorithms.

Memory (RAM):

- Recommendation: 16 GB minimum, 32 GB or more preferred.
- Rationale: Adequate memory is essential for handling large datasets and performing data-intensive operations efficiently.

Storage:

- Recommendation: SSD with at least 1 TB of storage.
- Rationale: Solid State Drives (SSDs) provide faster read/write speeds, which are crucial for data access and processing.

Graphics Processing Unit (GPU):

- Recommendation: NVIDIA GTX 1080 Ti or higher, such as the RTX 30 series.
- Rationale: GPUs accelerate machine learning tasks, especially deep learning model training.

Network:

- Recommendation: High-speed internet connection (1 Gbps or higher).
- Rationale: Necessary for downloading large datasets, software packages, and for cloud-based operations.

3.2 SOFTWARE REQUIREMENTS

Operating System:

- Recommendation: Windows 10/11, Ubuntu 20.04 LTS, or macOS.
- Rationale: These operating systems support a wide range of data science tools and libraries.

Programming Languages:

- Recommendation: Python 3.8 or higher, R.
- Rationale: Python and R are widely used in data science for their extensive libraries and community support.

Integrated Development Environment (IDE):

- Recommendation: Jupyter Notebook, PyCharm, VS Code.
- Rationale: These IDEs offer features tailored for data science workflows, such as interactive coding and debugging tools.

Python Libraries:

- scikit-learn: For general machine learning algorithms.
- pandas: For data manipulation and analysis.
- numpy: For numerical operations.
- TensorFlow/Keras or PyTorch: For deep learning models.
- matplotlib and seaborn: For data visualization..
- Catboost: designed for supervised learning tasks such as classification and regression

Database Management System:

- Recommendation: MySQL, PostgreSQL, or MongoDB.
- Rationale: These databases offer robust data storage solutions that are scalable and efficient for handling large datasets.

CHAPTER 4

SYSTEM DESIGN

4.1 System Architecture

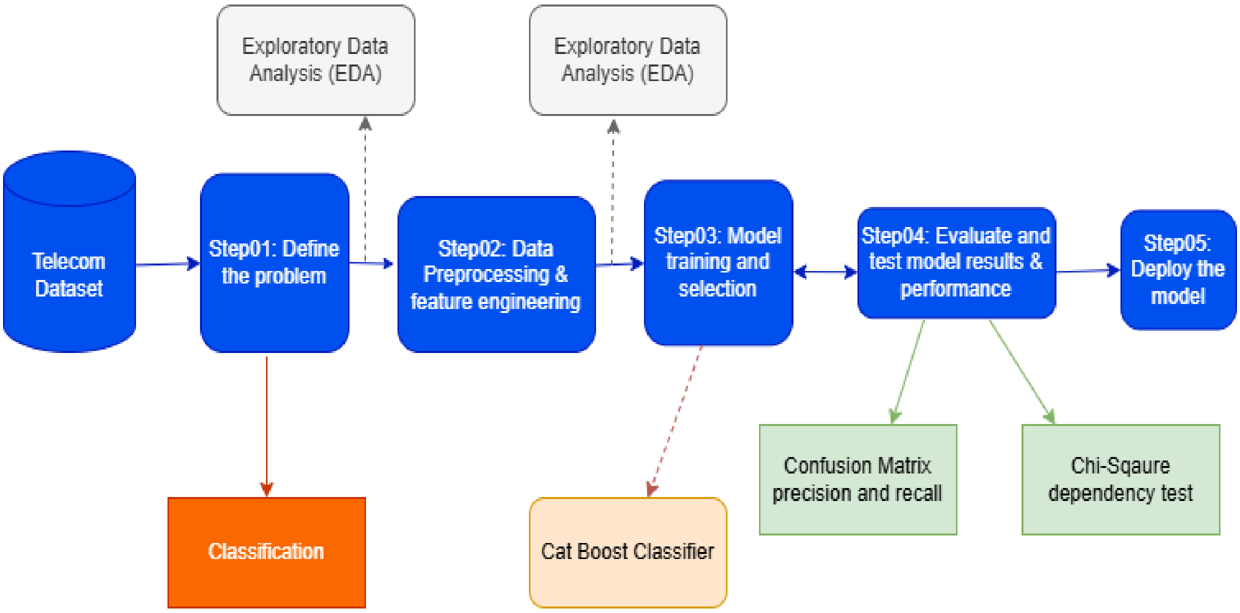


Fig no. 4.1 System Architecture

Data Collection and Storage:

- Data Sources: Collect data from various sources like CRM systems, transaction databases, web analytics, and customer feedback.
- Storage Solutions: Use data lakes or warehouses such as Amazon S3, Google Cloud Storage, or Azure Data Lake for storing large volumes of raw and processed data.

Data Processing:

- ETL (Extract, Transform, Load): Tools like Apache NiFi, AWS Glue, or Talend are used to extract data from different sources, transform it into a suitable format, and load it into storage.
- Data Preprocessing: Preprocessing involves cleaning, normalizing, and aggregating data. This can be done using Python with libraries like Pandas and NumPy or using cloud services like AWS Lambda or Google Cloud Functions.

Model Training:

- Environment: Use cloud-based environments like Amazon Sage Maker, Google AI Platform, or Azure ML Studio for scalable model training.
- Algorithms: Common algorithms include logistic regression, decision trees, random forests, gradient boosting (e.g., XGBoost), and deep learning models. Libraries like Cat Boost, scikit-learn, TensorFlow, or PyTorch are typically used.
- Feature Engineering: This involves creating meaningful features from raw data to improve model performance. Tools for feature engineering include Python with scikit-learn or dedicated platforms like Feature tools.

Model Evaluation and Tuning:

- Evaluation Metrics: Use metrics like accuracy, precision, recall, F1 score, and AUC-ROC to evaluate model performance.
- Hyperparameter Tuning: Automated hyperparameter tuning can be done using tools like Hyperopt, Optuna, or the built-in tuning features of SageMaker, Azure ML, or Google AI Platform.

Deployment:

- Model Serving: Deploy models using containerized services like Docker and Kubernetes or cloud-native solutions like SageMaker Endpoints, Google Cloud AI Platform Prediction, or Azure ML Service.
- Monitoring and Management: Use monitoring tools to track model performance and retrain models periodically. Tools like Prometheus, Grafana, and cloud-specific monitoring services are essential.

4.2 Methodology

Data Preparation:

- Collect relevant data points such as customer demographics, transaction history, engagement metrics, and customer feedback.
- Preprocess data to handle missing values, normalize features, and create new derived features.

Exploratory Data Analysis (EDA):

- Perform EDA to understand data distributions, correlations, and potential predictors of churn. Use visualization tools like Matplotlib, Seaborn, or Tableau.

Model Building:

- Select Algorithms: Choose suitable algorithms based on the data and business requirements. Often, a combination of models like logistic regression, random forests, and gradient boosting is used.
- Train Models: Use frameworks like TensorFlow, PyTorch, or scikit-learn to train models. Utilize cloud services for scalable training.

Model Validation:

- Validate models using cross-validation techniques and evaluate them against a hold-out test set to ensure they generalize well to unseen data.

Model Deployment:

- Deploy models to production using cloud services or on-premises solutions. Ensure models can scale to handle real-time predictions if needed.

Monitoring and Updating:

- Continuously monitor model performance in production and retrain models periodically with new data to maintain accuracy.

4.3 Cat Booster Classifier

CatBoost is a gradient boosting library that excels in handling categorical variables automatically during training, without the need for extensive preprocessing. It integrates advanced techniques to optimize training speed and reduce overfitting, including ordered boosting and L2 regularization. CatBoost provides insights into feature importance and is suitable for both classification and regression tasks, making it a powerful choice for various machine learning applications, particularly with large datasets.

CatBoost is a machine learning library known for its robust handling of categorical variables in data. Write about the principles and workings of the CatBoost Classifier, highlighting its key features and advantages.

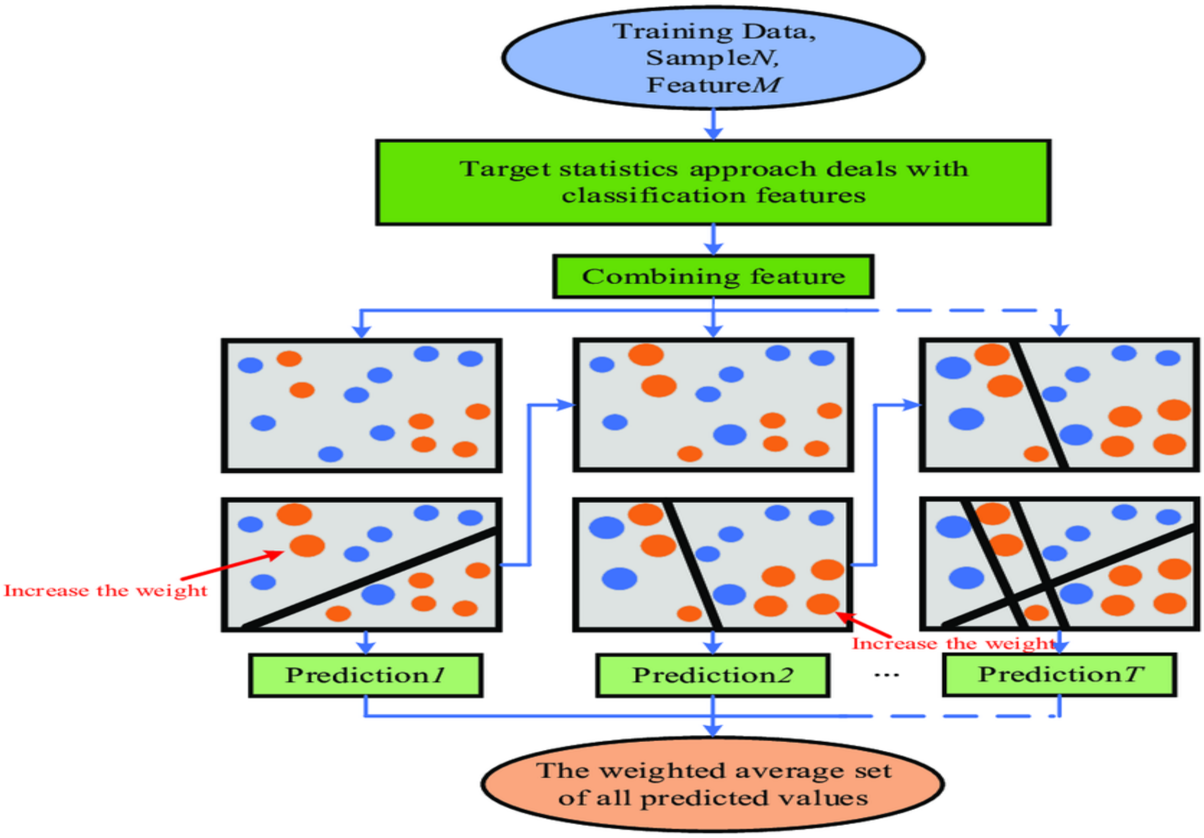


Fig no. 4.3 Cat Boost Classifier

Fig no. 4.3 depicts the Cat Boost Classifier Model

CHAPTER 5

IMPLEMENTATION

5.1 Loading Libraries and Data

```
1.Loading libraries and data

!pip install catboost

Collecting catboost
  Downloading catboost-1.2.5-cp310-cp310-manylinux2014_x86_64.whl (98.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.2/98.2 MB 6.2 MB/s eta 0:00:00
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from catboost) (0.20.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from catboost) (3.7.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.25.2)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-packages (from catboost) (2.0.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from catboost) (1.11.4)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (from catboost) (5.15.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (3.1.2)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly->catboost) (8.4.2)
Installing collected packages: catboost
Successfully installed catboost-1.2.5
```

Fig no. 5.1.1 Importing Libraries

```
[ ] import pandas as pd
    import numpy as np
    import missingno as msno
    import matplotlib.pyplot as plt
    import seaborn as sns
    import plotly.express as px
    import plotly.graph_objects as go
    from plotly.subplots import make_subplots
    import warnings
    warnings.filterwarnings('ignore')
```

Fig no. 5.1.2 Loading the Libraries

```python
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from xgboost import XGBClassifier

from catboost import CatBoostClassifier
from sklearn import metrics
from sklearn.metrics import roc_curve
from sklearn.metrics import recall_score, confusion_matrix, precision_score, f1_score, accuracy_score, classification_report
```

Fig no. 5.1.3 Loading the Libraries

```python
[ ]  #loading data
     df = pd.read_csv('/content/churndata.csv')

[ ]  from google.colab import drive
     drive.mount('/content/drive')
```

Fig no. 5.1.4 Loading the Dataset

5.2 Understanding the Data

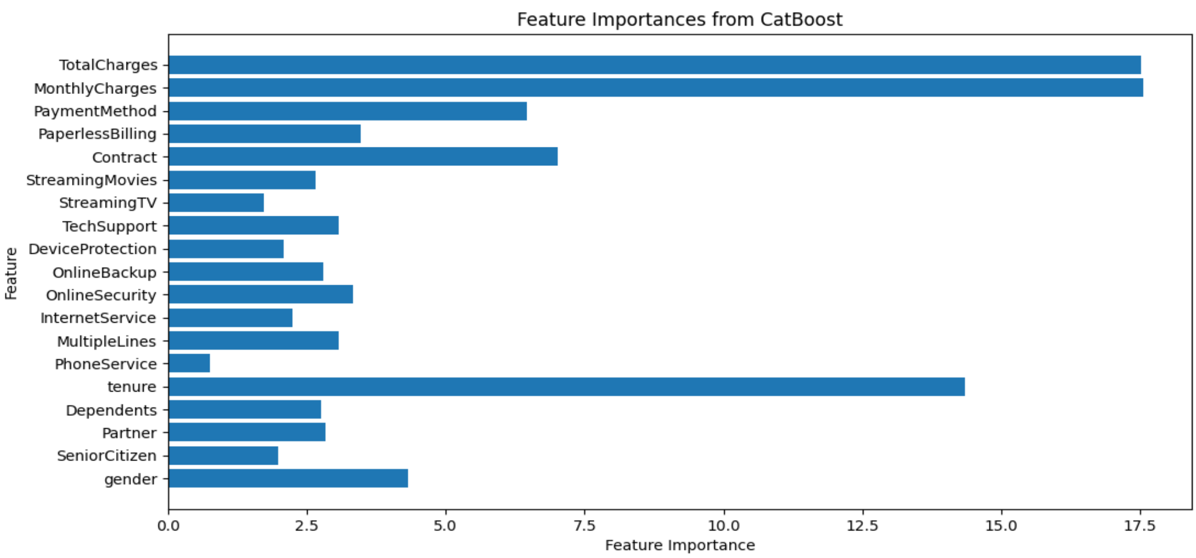


Fig 5.2.1 Bar Graph for Distribution of Services

The Figure 5.2.1 illustrates the distribution of services in a bar graph.

2. Understanding the data

Each row represents a customer, each column contains customer's attributes described on the column Metadata.

```
[ ] df.head()
```

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

5 rows × 21 columns

Fig no. 5.2.2 Understanding the Data

Fig no. 5.2.2 involves analyzing and interpreting the dataset to gain insights into its structure, patterns, and key characteristics relevant to the analysis or modeling task.

2. Understanding the data

Each row represents a customer, each column contains customer's attributes described on the column Metadata.

```
df.head()
```

InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
DSL	No	...	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
DSL	Yes	...	Yes	No	No	No	One year	No	Mailed check	56.95	1889.5	No
DSL	Yes	...	No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
DSL	Yes	...	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
Fiber optic	No	...	No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes

Fig no. 5.2.2 Understanding the Data

```
[ ] df.shape
(7043, 21)
```

Fig no. 5.2.3 Data frame Shape

Fig no. 5.2.3 returns a tuple representing the dimensions of a Dataframe, indicating the number of rows and columns it contains.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

Fig no. 5.2.4 Data Frame Information

Fig no. 5.2.4 provides a concise summary of a Pandas Data frame, showing the data types, non-null counts, and memory usage of each column.

```
df.dtypes
```

```
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
```

Fig no. 5.2.5 Data Frame Types

Fig no. 5.2.6 is a Pandas attribute that displays the data types of each column in a Data frame.

5.3 Visualize Missing Values

```
[ ] # Visualize missing values as a matrix
    msno.matrix(df);
```

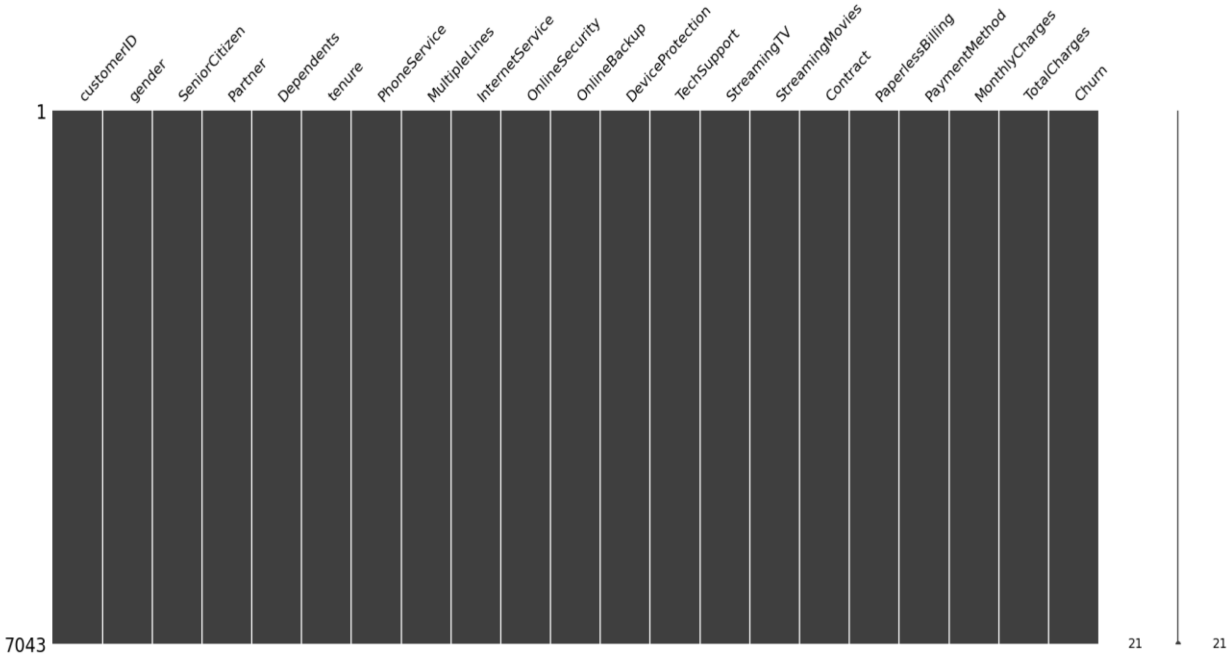


Fig no. 5.3.1 Visualizing Missing Data

5.4 Data Preprocessing

5. Data Preprocessing

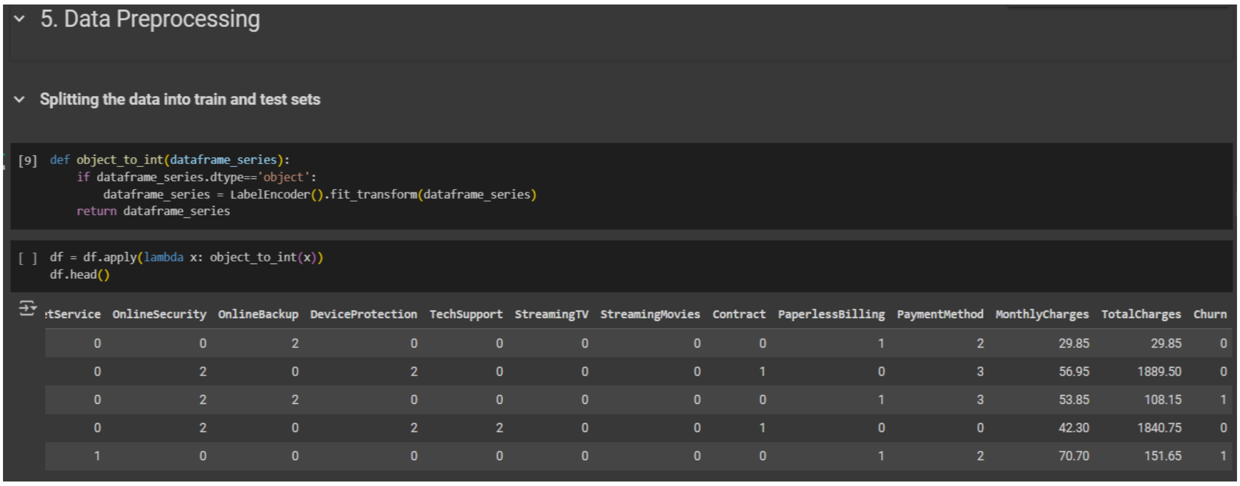
Splitting the data into train and test sets

```
[9] def object_to_int(dataframe_series):
        if dataframe_series.dtype=='object':
            dataframe_series = LabelEncoder().fit_transform(dataframe_series)
        return dataframe_series
```

```
[ ] df = df.apply(lambda x: object_to_int(x))
    df.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV
0	0	0	1	0	1	0	1	0	0	2	0	0	0
1	1	0	0	0	34	1	0	0	2	0	2	0	0
2	1	0	0	0	2	1	0	0	2	2	0	0	0
3	1	0	0	0	45	0	1	0	2	0	2	2	0
4	0	0	0	0	2	1	0	1	0	0	0	0	0

Fig no. 5.4.1 Data Preprocessing



5. Data Preprocessing

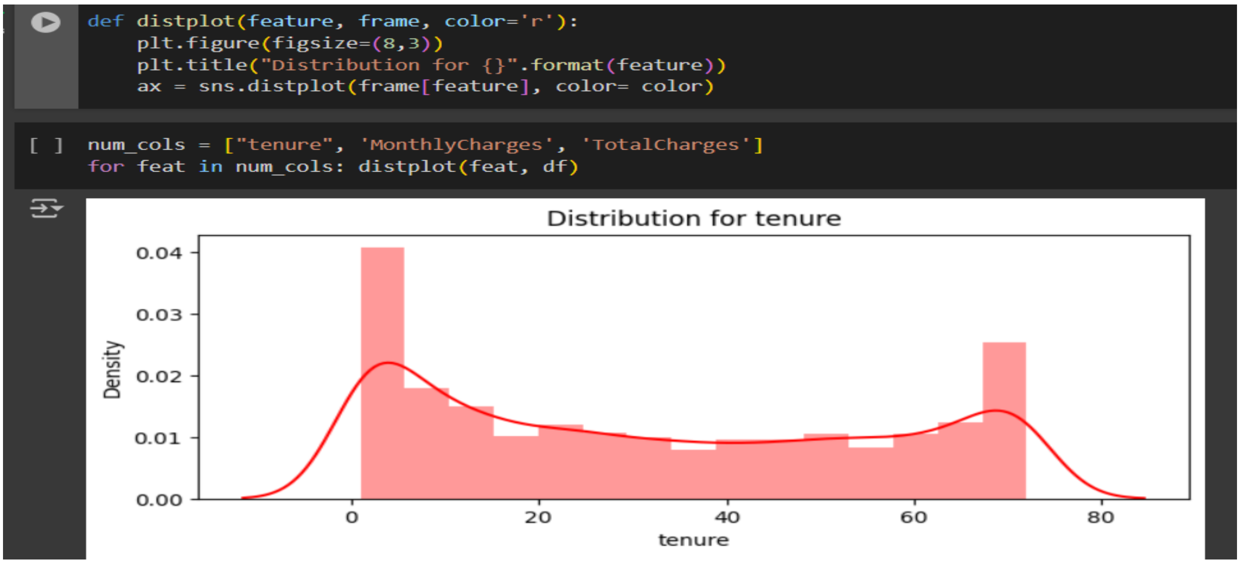
Splitting the data into train and test sets

```
[9] def object_to_int(dataframe_series):
        if dataframe_series.dtype=='object':
            dataframe_series = LabelEncoder().fit_transform(dataframe_series)
        return dataframe_series
```

```
[ ] df = df.apply(lambda x: object_to_int(x))
    df.head()
```

:tService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	0	2	0	0	0	0	0	1	2	29.85	29.85	0
0	2	0	2	0	0	0	1	0	3	56.95	1889.50	0
0	2	2	0	0	0	0	0	1	3	53.85	108.15	1
0	2	0	2	2	0	0	1	0	0	42.30	1840.75	0
1	0	0	0	0	0	0	0	1	2	70.70	151.65	1

Fig no. 5.4.2 Splitting the Data into Train and Test Sets



```
def distplot(feature, frame, color='r'):
    plt.figure(figsize=(8,3))
    plt.title("Distribution for {}".format(feature))
    ax = sns.distplot(frame[feature], color= color)
```

```
[ ] num_cols = ["tenure", 'MonthlyCharges', 'TotalCharges']
    for feat in num_cols: distplot(feat, df)
```


Fig no. 5.4.3 Distribution of Tenure

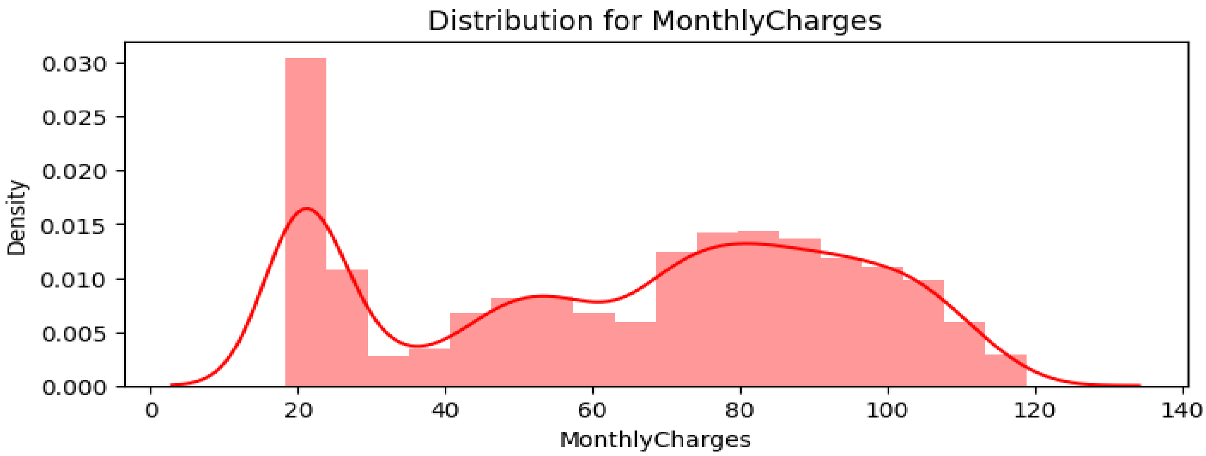


Fig no. 5.4.4 Distribution of Monthly Charges

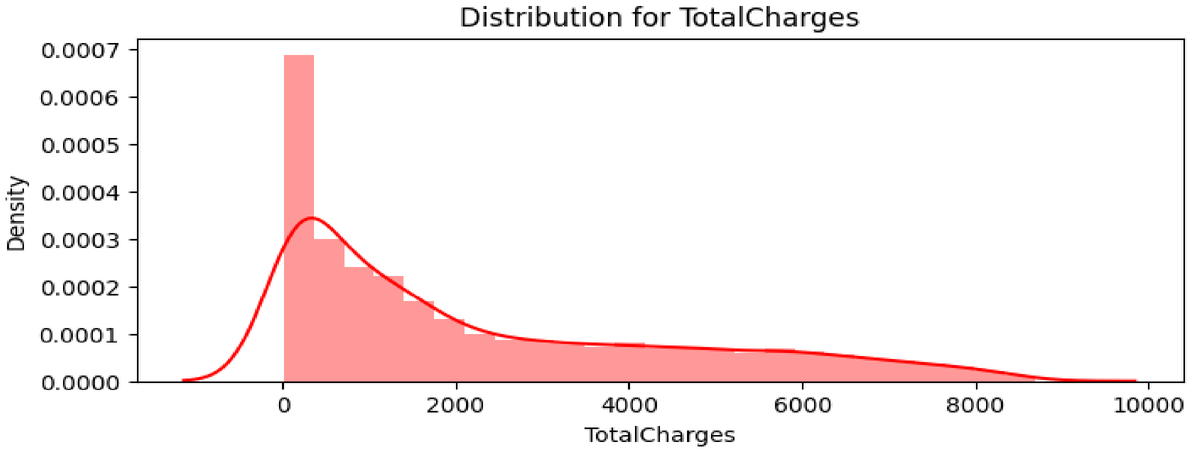


Fig no. 5.4.5 Distribution of Total Charges

5.5 Model Creation

```
import pandas as pd
import numpy as np
from catboost import CatBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import joblib
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
```

Fig no. 5.5.1 Importing Libraries

Fig no. 5.5.1 describes importing necessary libraries to develop a machine learning models

```
# Load data
data = pd.read_csv(r"C:\Users\Samyuktha\Desktop\churn data backup\data.csv")
```

Fig 5.5.2 Importing Dataset

Fig 5.5.2 shows the code snippet loads a dataset from a CSV file and displays the first few rows.

```
# Preprocess data
data = data.drop(["customerID"], axis=1)
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors='coerce')
data = data.dropna()
```

Fig no. 5.5.3 Preprocess data

Fig 5.5.3 describes preprocessing data involves cleaning and transforming raw data into a format suitable for analysis or model training

```
# Encode categorical features
label_encoders = {}
for column in data.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    data[column] = le.fit_transform(data[column])
    label_encoders[column] = le
```

Fig no. 5.5.4 Encode categorical features

Fig no. 5.5.4 features involve converting non-numeric data into a numerical format suitable for machine learning algorithms.

```
# Split data into features and target
X = data.drop("Churn", axis=1)
y = data["Churn"]
```

Fig no. 5.5.5 Split Data into Features and Target

Fig no. 5.5.5 involves separating the dataset into independent variables (features) and the dependent variable (target) that you want to predict

```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Fig no. 5.5.6 Split Data into Training and Test Sets

Fig no. 5.5.6 involves dividing the dataset into two parts: one for training the model and the other for evaluating its performance.

```
# Train CatBoostClassifier
catboost_model = CatBoostClassifier(iterations=1000, depth=6, learning_rate=0.1, loss_function='Logloss', verbose=False)
catboost_model.fit(X_train, y_train)
```

Fig no. 5.5.7 Train Cat Boost Classifier

Fig no. 5.5.7 Training a Cat Boost Classifier involves fitting the Cat Boost algorithm to the training data to create a model that can predict categorical target values.

```
# Evaluate the model
y_pred = catboost_model.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Fig no.5.5.8 Evaluate the Model

Fig no.5.5.8 involves assessing its performance on test data using metrics such as accuracy, precision, recall, and F1-score to determine how well it makes predictions.

```
# Save the model
joblib.dump(catboost_model, 'catboost_model.pkl')
```

Fig no.5.5.9 Save the Model

Fig no.5.5.9 involves storing the trained model to a file so it can be reused or deployed without retraining.

```
# Plot feature importances
feature_importances = catboost_model.get_feature_importance()
plt.figure(figsize=(12, 6))
plt.barh(X.columns, feature_importances)
plt.xlabel("Feature Importance")
plt.ylabel("Feature")
plt.title("Feature Importances from CatBoost")
plt.show()
```

Fig no. 5.6 Plot feature importances

Fig no. 5.6 involves creating a visual representation that shows the relative importance of each feature in making predictions within the model.

CHAPTER 6

RESULTS

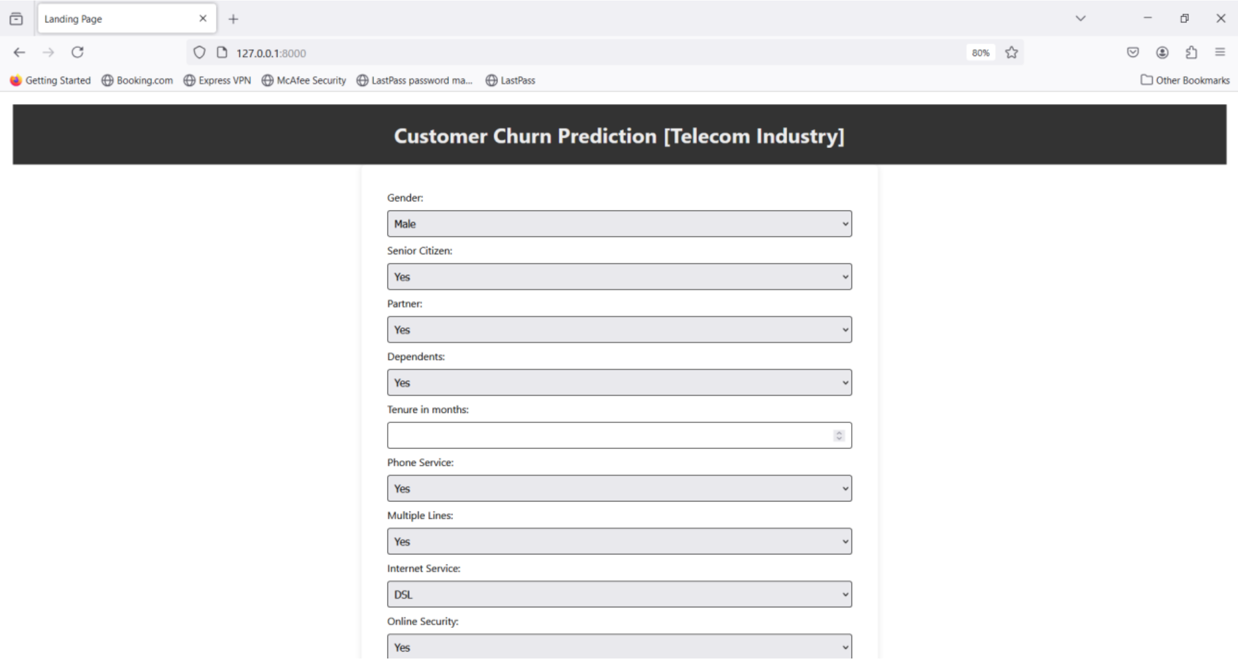


Fig no. 6.1.1 GUI used for Customer Churn Prediction

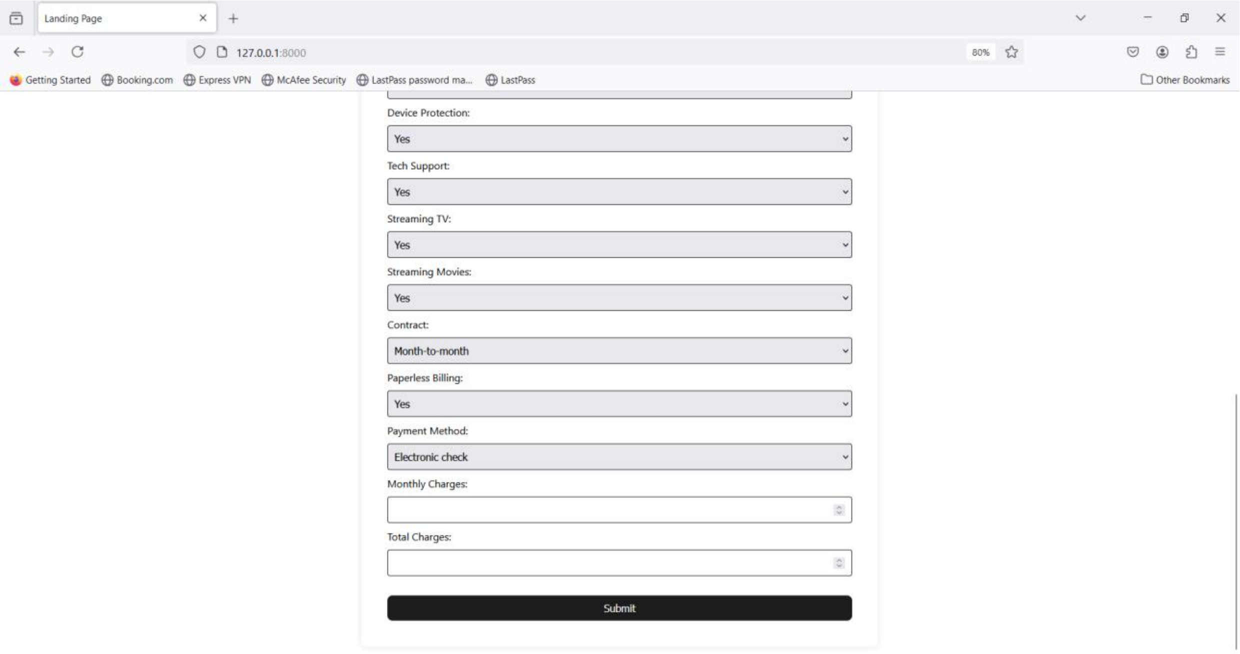


Fig no. 6.1.2 GUI used for Customer Churn Prediction

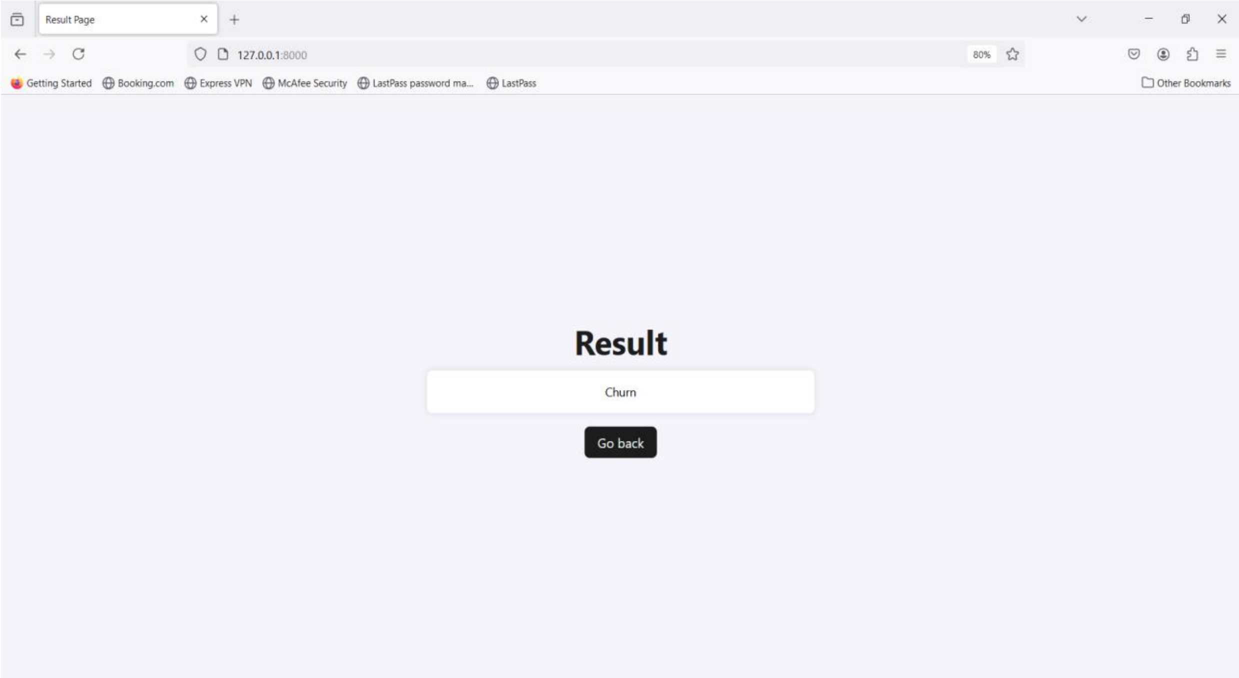


Fig no. 6.1.3 Result/Output

Fig no. 6.1.3 shows the output predicting if a customer uses all the services provided by the company the result shows Churn

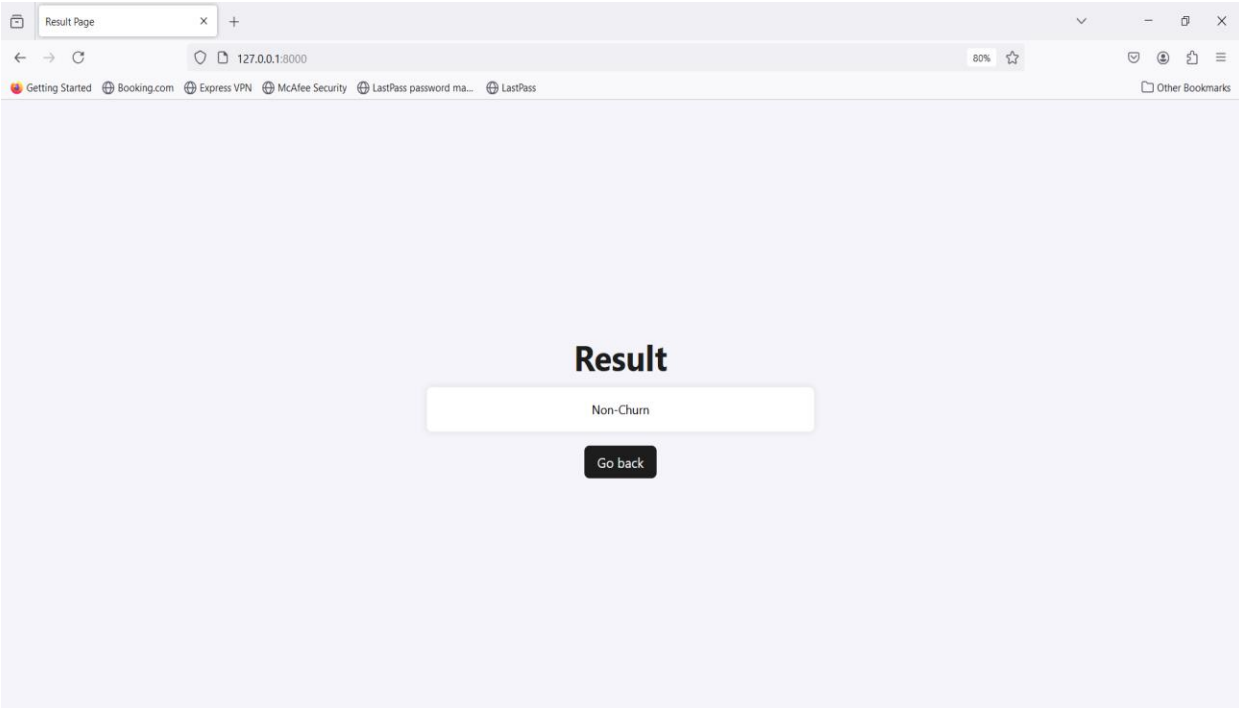


Fig no. 6.1.4 Result/Output

Fig no. 6.1.4 shows the output predicting that If he won't use the services of company then the results shows as Non-Churn

CHAPTER 7

CONCLUSION

Customer churn modeling is crucial for improving customer retention and business profitability. A structured approach involving data collection, preprocessing, model building, evaluation, and deployment, supported by modern tools and cloud platforms, enhances the effectiveness and scalability of churn prediction models. This process enables businesses to identify and address factors leading to customer churn, ultimately leading to better customer retention strategies and increased revenue. By following a structured approach and leveraging modern tools and platforms, businesses can significantly enhance their ability to predict and mitigate customer churn, ultimately leading to improved customer retention and increased profitability.

- Integrate the model into the company's CRM system for real-time churn prediction.
- Explore additional features and data sources to improve prediction accuracy.
- Conduct A/B testing to evaluate the effectiveness of targeted retention strategies based on model predictions.
- Key features influencing churn include contract type, tenure, and monthly charges.
- The predictive model enables the business to identify at-risk customers and implement targeted retention strategies.

REFERENCES

- https://aws.amazon.com/blogs/machine-learning/build-tune-and-deploy-an-end-to-end-churn-prediction-model-using-amazon-sagemaker-pipelines
- https://link.springer.com/article/10.1007/s10207-019-00435-4
- https://www.sciencedirect.com/science/article/abs/pii/S1877050916304862
- https://cloud.google.com/blog/products/ai-machine-learning/predicting-customer-churn-with-ai-and-machine-learning
- https://www.researchgate.net/publication/368319203/figure/fig1/AS:11431281118505127@1675782612905/Flow-chart-of-CatBoost-algorithm.png
- telecom customer churn model.drawio.png - Google Drive