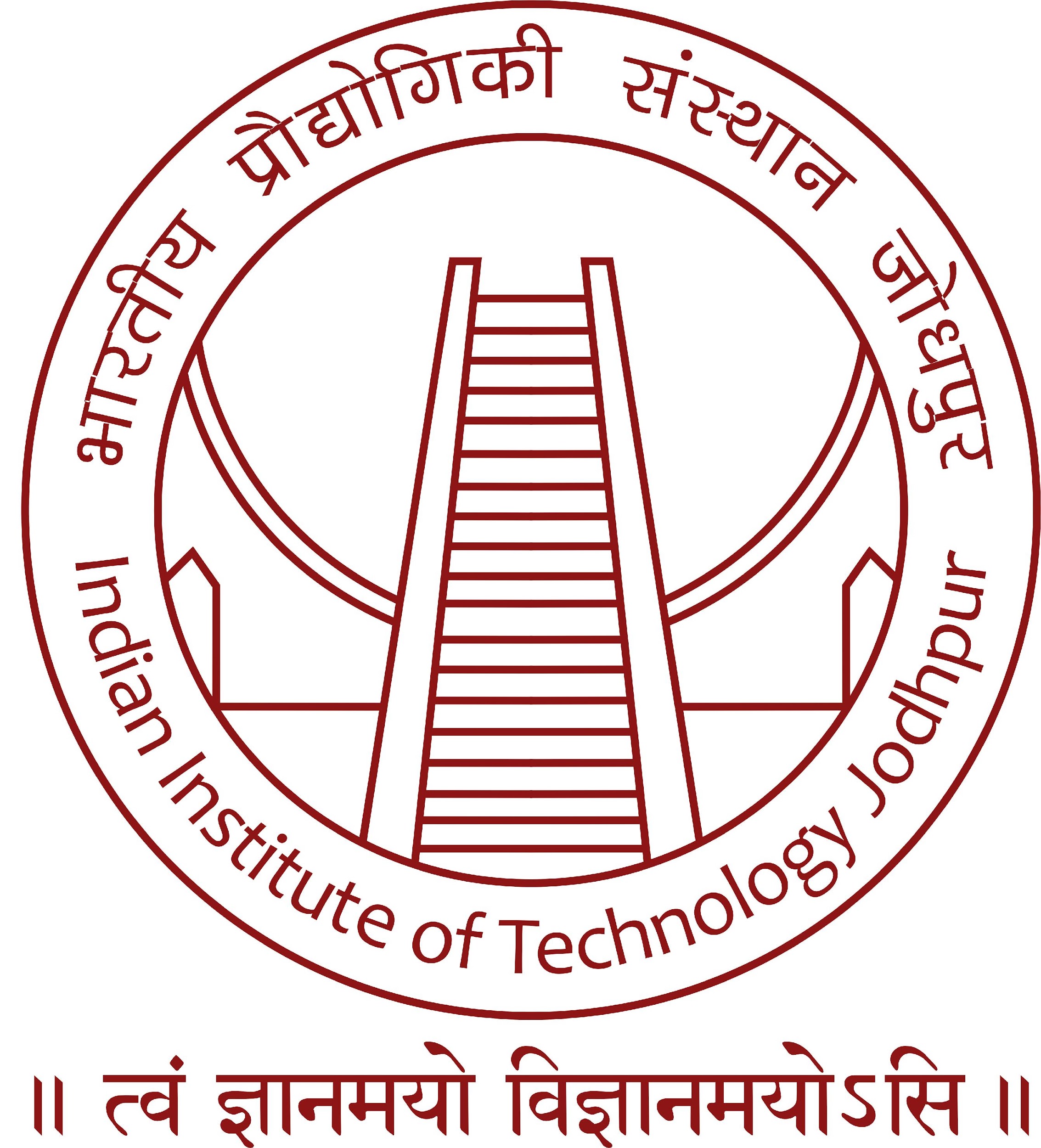
**Customer Churn Prediction Model**

Project Report

Submitted For

Machine Learning

Under Supervision of  
Prof. Sandeep Yadav



Submitted By

| Name | Roll Number |
| --- | --- |
| Atul Singh | G23AI2104 |
| Mayank Goyal | G23AI2120 |
| Rahul Verma | G23AI2039 |
| Suraj Mourya | G23AI2116 |
| Vidyut Bhaskar | G23AI2128 |

### CONTENTS

[**CONTENTS 2**](#_qwbmnsd643xa)

[**PROBLEM STATEMENT 3**](#_27p02sfb1kgs)

[**OBJECTIVE 3**](#_b2ysdzy5hgg5)

[**PROBLEM DEFINITION 4**](#_5aeaxi7221cr)

[**1. Task 4**](#_vgck08iw5stu)

[**2. Performance 4**](#_ok8kx59ukb70)

[**3. Experience 4**](#_ec0nci2ps78f)

[**DATASET USED 4**](#_5k864kszaael)

[**PROJECT SETUP 5**](#_px28cjyh94s7)

[**EXPLORATORY DATA ANALYSIS 5**](#_5py7lc71okc2)

[Unbalanced Dataset 9](#_x2gi5u8zbk2)

[Solution to Unbalanced Dataset 9](#_g5gncj771d3t)

[**FEATURE ENGINEERING 9**](#_ll9pkpxi6o4w)

[Selection of Encoding Technique 10](#_sgda5zuwwigp)

[Label Encoder 10](#_1gf2hwux4k9x)

[Standardization 11](#_9r7ig0z4bhcm)

[**FEATURE SELECTION AND EXTRACTION 12**](#_hfu7rtr2b146)

[Methods of Feature Extraction: 12](#_32qvj38ydswd)

[**MODELS 14**](#_76erfhbpxs55)

[Dataset Balancing and Splitting Train and Test Dataset 14](#_nri0acx79qog)

[Original Dataset : 14](#_8aht8iwwm4pf)

[After Balancing : 14](#_ylb31iieg09m)

[Train, Test Split 15](#_y20jpre40yqd)

[LOGISTIC REGRESSION 15](#_qb6ldi7whsvj)

[Model 16](#_n62bc74b9m49)

[Model Performance Metrics 17](#_s5vldbsrz1e3)

[Analysis of Confusion Matrix 17](#_nn27ks1i4bjs)

[Model Exporting 18](#_jvp57120142g)

[SUPPORT VECTOR CLASSIFIER 18](#_8s16xn99wt3a)

[Model 18](#_yesnjx74npr7)

[Model Performance Metrics 19](#_ljkwgmpf0pq)

[Model Exporting 21](#_w6ecaclzf010)

[DECISION TREE CLASSIFICATION - RANDOM FOREST 21](#_1xdh5hp37ph4)

[Model 21](#_x84g8clnq63)

[Model Performance Metrics 22](#_f9qpq3ojmovy)

[**REFERENCES 23**](#_46d9d9aggohj)

### PROBLEM STATEMENT

In the contemporary business landscape, customer retention is critical for sustaining long-term profitability. The cost associated with acquiring new customers is significantly higher than retaining existing ones. Consequently, understanding and mitigating customer churn—where customers discontinue their use of a company’s product or service—has become a priority for businesses across various industries.

### OBJECTIVE

The goal of this project is to develop a robust machine learning model that can accurately predict customer churn. This model will utilize historical customer data, including behavioral, transactional, and demographic information, to identify patterns that precede churn events.

Some machine learning algorithms that can be used for churn prediction include:

* Logistic regression
* Bayes algorithm
* Decision trees
* Support vector machines
* Linear discriminant analysis
* Random forest
* K-means
* Hierarchical clustering

In this project we have used Logistic regression, Support Vector Machine and Random Forest algorithm to predict the Target Variable.

A 5 step process will be followed to build the Customer churn prediction model.

1. Exploratory Data Analysis
2. Data Preprocessing
3. Feature Extraction and Selection
4. Customer Classification and Prediction
5. Model benchmarking

At the end, the goal is to benchmark algorithms for customer churn as given above and to suggest the best approach based on different KPIs. If required upon analysis if the project team feels to suggest a phased approach the same shall be documented and presented.

**Assumption - Training and Test Data are sampled from the same dataset.**

### PROBLEM DEFINITION

#### 1. Task

* **Objective:** The primary task is to predict which customers are likely to churn, i.e., stop using a service or product within a specific timeframe.
* **Action:** Develop a predictive model that classifies customers into "likely to churn" and "not likely to churn" categories.

#### 2. Performance

For evaluating the model performance we use some metrics evaluation such as:  
  
Feature Importance— List the top features to make a prediction. Feature weights are used for the Random forest model

Confusion Matrix — Show the true and false then compare to prediction and actual values.

Accuracy Score — Show the accuracy model from training and testing to measure that a model is suitable for prediction.

F1 Score — This metric is usually better for accuracy, especially if it has an imbalanced data set target distribution.

#### 3. Experience

* **Customer Retention:** The goal of the model is to improve prediction learning from experiences i.e., improve customer retention rates by reducing churn. This involves not just correctly predicting churn but also identifying important features of customers to implement retention strategy.
* **Business Impact:** A positive experience with the churn model leads to reduced revenue loss, increased customer lifetime value, and better overall customer satisfaction.

### DATASET USED

This project aims to develop a robust machine learning model using the IBM Telco Customer Churn dataset. 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 20 features.

### PROJECT SETUP

The production application of this project is hosted on **Streamlit** application. **Streamlit** is an open-source Python framework for data scientists and AI/ML engineers to deliver interactive data **apps**.

The link to application is <https://iitj-ml-assingment-2.streamlit.app/>

The sandbox application is created in a flask application. The code repository is available at

<https://github.com/DSvidyutbhaskariitj/Model>

The dataset used in the project is hosted in Azure data container as blob storage. The url to access this dataset is “<https://vbfilestorage.blob.core.windows.net/datasets/Telco-Customer-Churn.csv>”

### EXPLORATORY DATA ANALYSIS

Dataset comprises of following features

--- ------ -------------- -----

0 customerID 7043 non-null object

1 gender 7043 non-null object

2 SeniorCitizen 043 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 7032 non-null float64

20 Churn 7043 non-null object

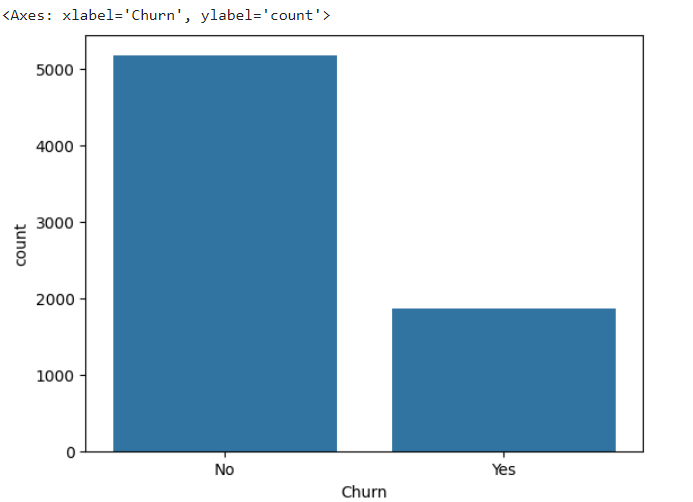
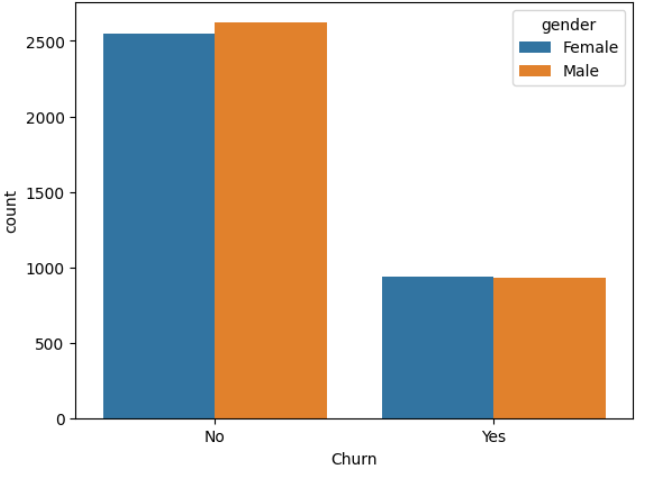
There are 3 Interval Features which are as follows -

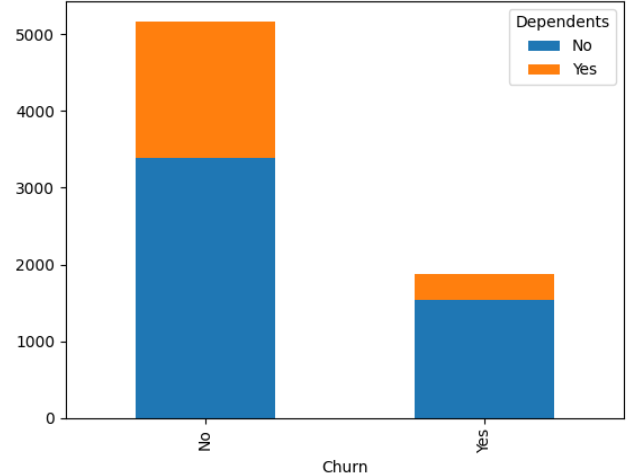
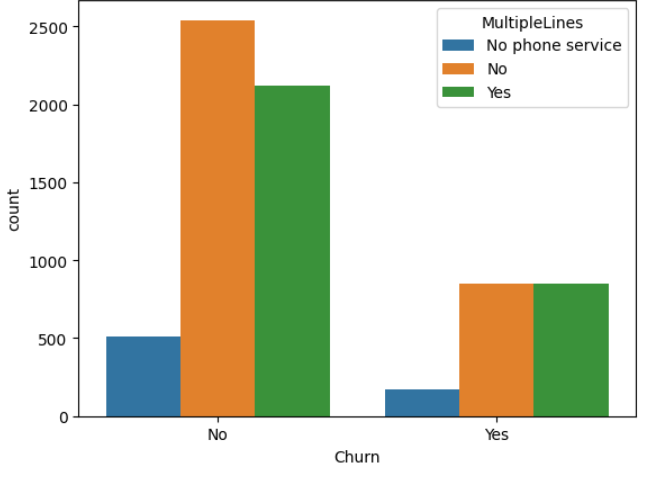
* Tenure, MonthlyCharges, TotalCharges

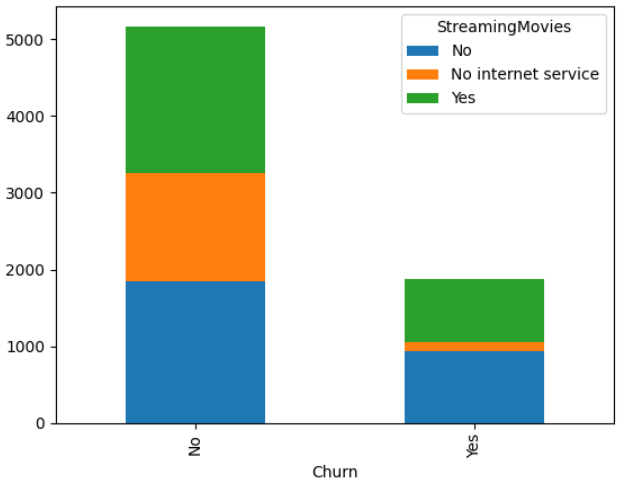
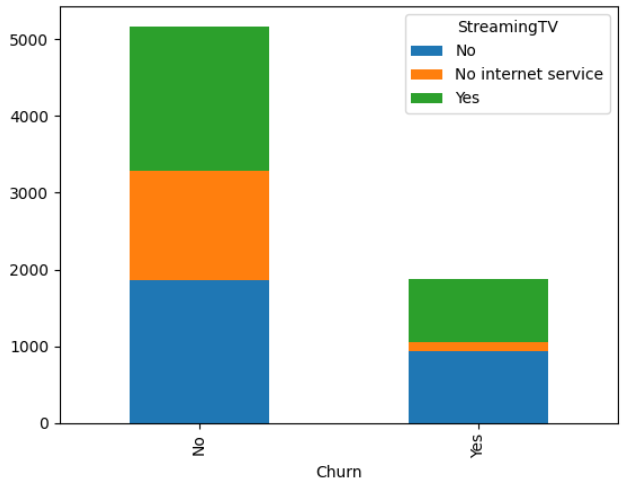
There are 18 Categorical features which are as follows -

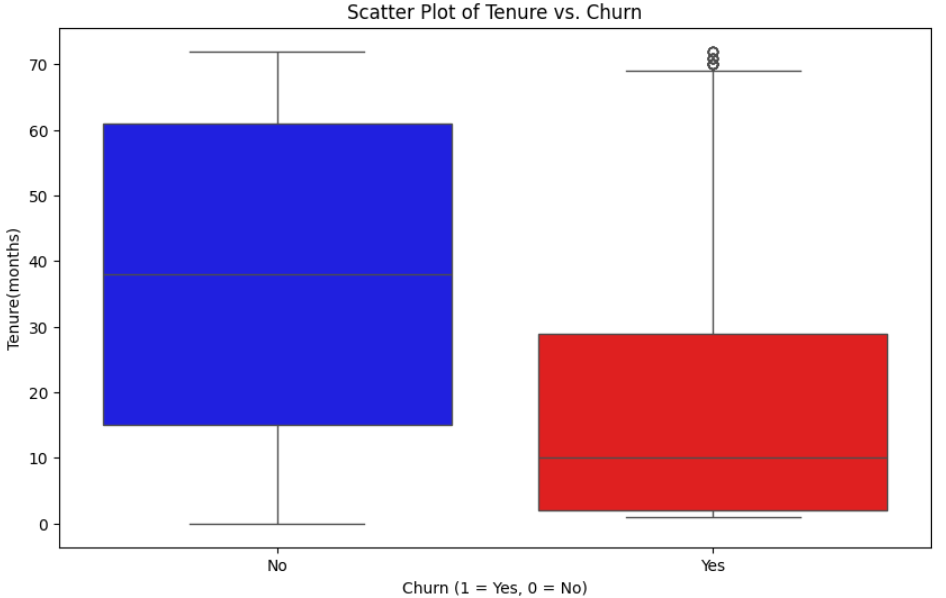
* Gender, Senior Citizen, Partner, Dependents, PhoneService, MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies, Contract, PaperlessBilling, PaymentMethod

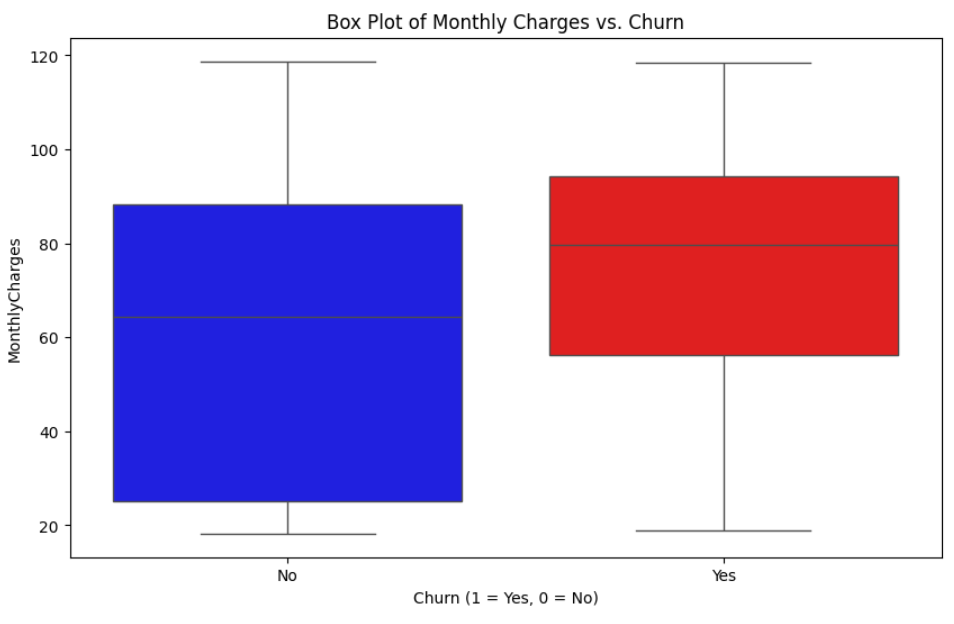
**Target Variable - Churn**

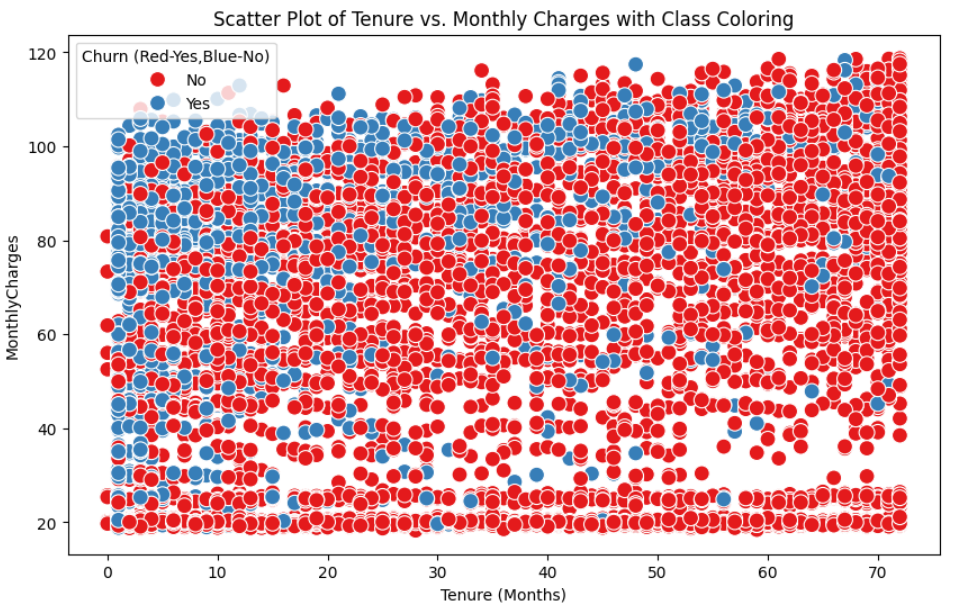
** **

****

****

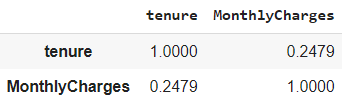
****

****

****

**Analysis-**

1. This is an unbalanced dataset
2. The dataset comprises multiple categorical feature sets, hence requiring Encoding during feature engineering.
3. Churn is almost same across both genders ‘Male’ and ‘Female’
4. The %age of Customers who had dependents added to the service Churned less compared to the ones who hadn’t
5. Customers who didn’t Churn had higher Average tenure compared to who Churned
6. Customers who didn’t Churn payed lower Average monthly charges compared to who Churned
7. There is less significant correlation between Monthly Charges and Tenure. Also there is no visual segregation with respect to Churn class (Yes and No)



1. There are no missing values in the dataset. Hence does not require handling missing values.

#### Unbalanced Dataset

Unbalanced datasets are those where one class or category significantly outnumbers others. It can lead to several problems in the Machine Learning model

1. High Bias - Bias towards majority class
2. Poor Generalization of model
3. Overfitting

#### Solution to Unbalanced Dataset

When the dataset is unbalanced we apply techniques to balance the dataset. Some of which are

1. Data Augmentation
   1. Generating data with mean, median or mode especially for continuous data.
2. Resampling
   1. Oversample with techniques such as SMOTE
   2. Undersample

3. Ensembling

* 1. Bagging
  2. Boosting - Adaptive Boosting or Gradient based boosting

In this project, SMOTE technique is used to synthetically create data which is used in Logistic Regression and SVM algorithms.

In this project Ensembling technique is used in Decision Tree Classification.

### FEATURE ENGINEERING

For a machine learning model the input features are represented in form of vector like

X’ = {x1​,x2​,...,xn​}

Where,

X: The vector or set of features that describe each instance (observation) in the dataset.

x1,x2,...,xn​: The individual features or variables that make up the vector X

In a general form

Y-hat = h(X,θ)

Hence, it is required to convert categorical variables to an encoded variable first.

In this project we have mapped binary variables from **‘Yes’ to 1 and ‘No’ to 0**

#### Selection of Encoding Technique

Label encoding may introduce unintended relationships between values that are not truly ordinal. For such cases sparse matrix can be used. Example sparse matrix for the dataset.

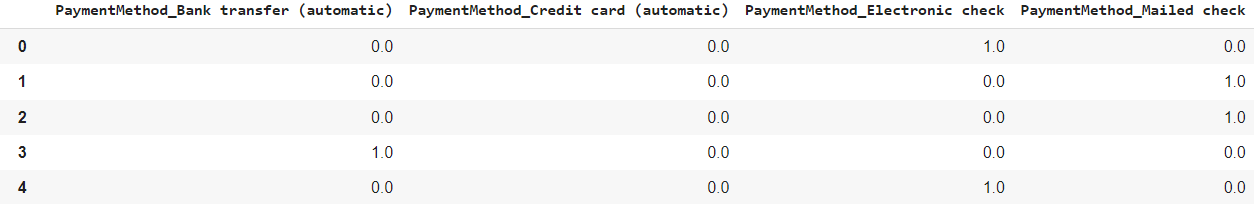
*Code Snippet*

from sklearn.preprocessing import OneHotEncoder

ohe\_encoder=OneHotEncoder()

encoded = ohe\_encoder.fit\_transform(df\_original[['PaymentMethod']]).toarray()

encoder\_df = pd.DataFrame(encoded,columns=ohe\_encoder.get\_feature\_names\_out())

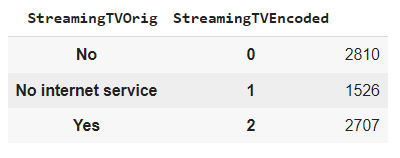
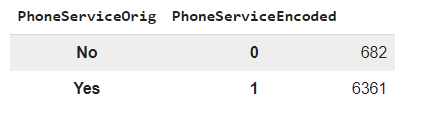


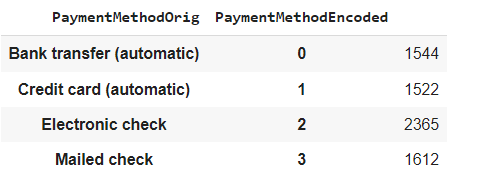
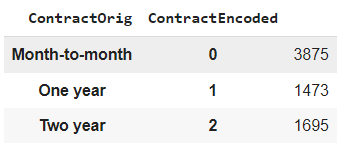
The OneHotEncoder technique increases the dimension of the dataset. This may introduce complexity in the model convergence. It makes it unable to handle new categorical data.

Hence we choose Label Encoder as the solution.

#### Label Encoder

For categorical variables, we have used Label Encoder. Label Encoding assigns a unique integer value to each distinct category (or label) in the dataset. For example:

#### Standardization

The **Standard Scaler** is a widely used technique for feature scaling in machine learning. It ensures that the features (or input data) have similar scales, which can significantly improve the performance of many machine learning models. The Standard Scaler achieves this by transforming the features to have a **mean of 0** and a **standard deviation of 1**.

It standardizes features with the help of z-score. In the project we standardize numerical data,

*CodeSnippet*

*from sklearn.preprocessing import StandardScaler*

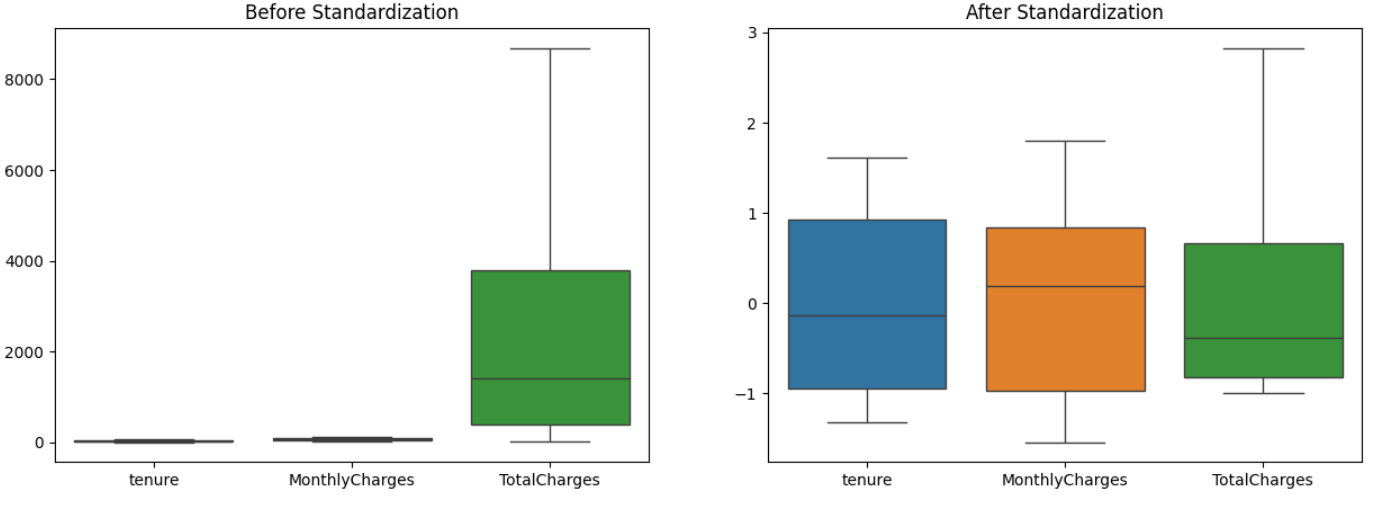
*scaler = StandardScaler()*

*df[['tenure', 'MonthlyCharges', 'TotalCharges']] = scaler.fit\_transform(df[['tenure', 'MonthlyCharges', 'TotalCharges']])*

*# Dropping customerID since it has no predictive value*

*df = df.drop(columns=['customerID'])*

Results of Data Standardization

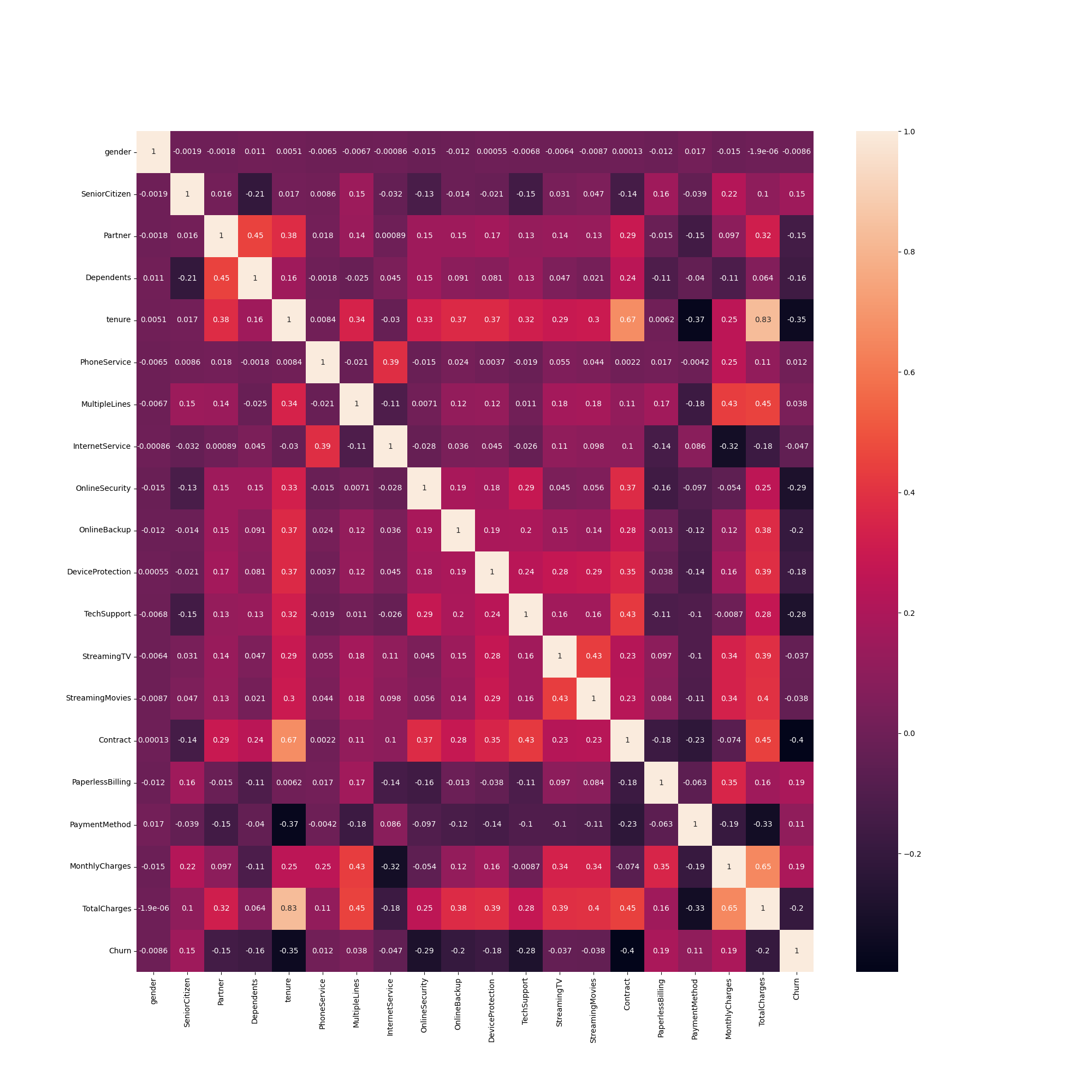


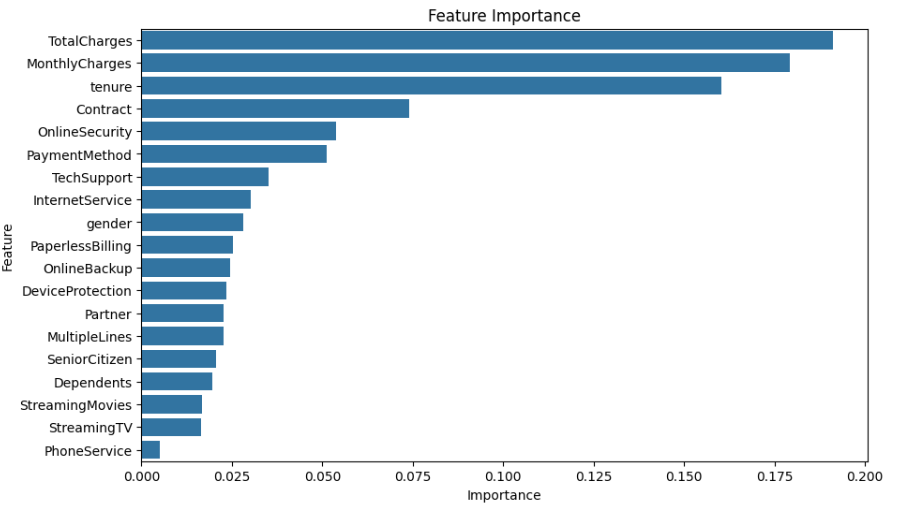
### FEATURE SELECTION AND EXTRACTION

**Feature extraction** involves transforming the data into a new space where it can be more effectively analyzed or modeled. Unlike feature selection, which reduces the number of features, feature extraction creates new features from the original data.

#### **Methods of Feature Extraction:**

1. **Principal Component Analysis (PCA)**: PCA reduces the dimensionality of the data by projecting it onto a lower-dimensional space while preserving as much variance as possible.
2. **Linear Discriminant Analysis (LDA)**: LDA is used for dimensionality reduction and classification. It maximizes the separation between multiple classes.
3. **Correlation Threshold** - Features who have high correlation with Target variable can be omitted via threshold technique. **This correlation threshold can be set to be 0.9**

In this dataset, as we can see the correlation plot, there is no high correlation of feature with target variable hence, there is no elimination required based on this

G

Predicted by Random Forest Model

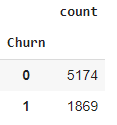
### MODELS

#### Dataset Balancing and Splitting Train and Test Dataset

Using SMOTE technique, dataset is balanced and results are as follows

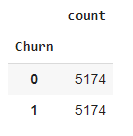
##### Original Dataset :

In original dataset the Churn count is not balanced with **No as 5174 and Yes as 1869**



##### After Balancing :

The Churn count is balanced with **Yes as 5174 and No as 5174**. That means we have synthesized the data to introduce scenarios where Customer may churn. Oversampling is done for the dataset



*CodeSnippet*

*from imblearn.over\_sampling import SMOTE*

*oversample=SMOTE(sampling\_strategy='minority')*

*X\_smote,y\_smote=oversample.fit\_resample(df[['gender', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',*

*'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaymentMethod','tenure','MonthlyCharges','TotalCharges']],df['Churn'])*

#### Train, Test Split

In this project, the dataset is split for a test with 25% i.e., 0.25.

*Code Snippet*

*from sklearn.model\_selection import train\_test\_split*

*from sklearn.linear\_model import LogisticRegression*

*from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix*

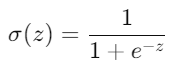
*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_smote, y\_smote, test\_size=0.25, random\_state=42)*

#### LOGISTIC REGRESSION

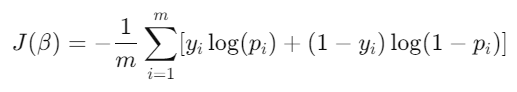
**Logistic Regression** is a widely used statistical method for binary classification problems. It predicts the probability of a binary outcome based on one or more predictor variables. Despite its name, logistic regression is used for classification, not regression.

**Binary Classification**: Logistic Regression is designed for problems where the outcome variable is binary like in our problem where Target variable is Customer Churn - Yes or No

Sigmoid function in logistic regression



The goal of the logistic regression is to minimize the Cost function or Loss Function given by



Where



##### Model

Used default parameters

Penalty 'l2'

Tolerance 0.0001,

Regularization Strength C 1.0,

Fit\_intercept True,

Class\_weight None

Solver 'lbfgs'

from sklearn.linear\_model import LogisticRegression

Model Coefficients

array([[-0.27149303, 0.06756817, -0.02905543, -0.31552086, -0.16077064,

-0.07383471, -0.26232631, 0.05514369, 0.0463841 , -1.23082227,

-0.03904518, -1.49504082, 0.51133148, 0.9143863 ]])

##### Model Performance Metrics

Logistic Regression Results:

Accuracy: 0.7811594202898551

Confusion Matrix:

[[741 280]

[173 876]]

Classification Report:

precision recall f1-score support

0 0.81 0.73 0.77 1021

1 0.76 0.84 0.79 1049

accuracy 0.78 2070

macro avg 0.78 0.78 0.78 2070

weighted avg 0.78 0.78 0.78 2070

##### Analysis of Confusion Matrix

**True Negatives (TN)**: 741

**False Positives (FP)**: 280

**False Negatives (FN)**: 173

**True Positives (TP)**: 876

**Accuracy: 0.78**

**Precision for True Positives : 0.76**

**Precision for True Negatives : 0.81**

##### Model Exporting

*CodeSnippet*

*import pickle*

*pickle.dump(lr,open('LogisticRegression.pkl','wb'))*

The LogisticRegression.pkl model is exported as.pkl files that is later used by our application to predict Churn.

#### SUPPORT VECTOR CLASSIFIER

The **Support Vector Classifier (SVC)** is a machine learning technique used for classification tasks. It is part of the Support Vector Machine (SVM) family, which aims to find the best boundary (or hyperplane) that separates data into different classes. The SVC algorithm is known for its effectiveness in high-dimensional spaces and its robustness against overfitting, especially in cases where the number of dimensions exceeds the number of samples.

**Kernel Trick**:

* For non-linearly separable data, SVC can use different kernels to map the data into higher-dimensional spaces where a linear separator might be possible.
* Common kernels include:
  + **Linear Kernel**: No transformation, used for linearly separable data.
  + **Polynomial Kernel**: Maps data into polynomial feature space.
  + **Radial Basis Function (RBF) Kernel**: Maps data into an infinite-dimensional space using the Gaussian function.
  + **Sigmoid Kernel**: Uses a sigmoid function to map data.

We will calculate the model accuracy with SVC with above kernel tricks

##### Model

from sklearn.svm import SVC

SVC includes a regularization parameter C that controls the trade-off between maximizing the margin and minimizing classification error.

A high value of CCC aims to classify all training examples correctly (less margin, more focus on classification accuracy), while a low value of CCC creates a larger margin (more generalization, potentially allowing some misclassifications).

We have used C = 1.0 (default)

##### Model Performance Metrics

**Model with Radial Basis Function (RBF) Kernel**

Support Vector Results:

Accuracy: 0.7974487823734054

Confusion Matrix:

[[ 990 308]

[ 216 1073]]

Classification Report:

precision recall f1-score support

0 0.82 0.76 0.79 1298

1 0.78 0.83 0.80 1289

accuracy 0.80 2587

macro avg 0.80 0.80 0.80 2587

weighted avg 0.80 0.80 0.80 2587

**Model with Sigmoid Kernel**

Support Vector Sigmoid Results:  
Accuracy: 0.6443757247777349  
Confusion Matrix:  
 [[842 456]

[464 825]]

Classification Report:

precision recall f1-score support

0 0.64 0.65 0.65 1298

1 0.64 0.64 0.64 1289

accuracy 0.64 2587  
 macro avg 0.64 0.64 0.64 2587  
weighted avg 0.64 0.64 0.64 2587

**Model with Linear Kernel**

Support Vector Linear Results:

Accuracy: 0.7692307692307693

Confusion Matrix:

[[ 907 391]

[ 206 1083]]

Classification Report:

precision recall f1-score support

0 0.81 0.70 0.75 1298

1 0.73 0.84 0.78 1289

accuracy 0.77 2587

macro avg 0.77 0.77 0.77 2587

weighted avg 0.77 0.77 0.77 2587

**Model with Poly Kernel**

Support Vector Results:  
Accuracy: 0.7966756861229223  
Confusion Matrix:  
 [[ 954 344]

[ 182 1107]]

Classification Report:

precision recall f1-score support

0 0.84 0.73 0.78 1298

1 0.76 0.86 0.81 1289

accuracy 0.80 2587

macro avg 0.80 0.80 0.80 2587

weighted avg 0.80 0.80 0.80 2587

**SVC Result Summary**

| Kernel Type | Accuracy | Precision (Positive) | Precision (Negative) |
| --- | --- | --- | --- |
| **RBF** | **0.80** | **0.82** | **0.78** |
| Sigmoid | 0.64 | 0.64 | 0.64 |
| Linear | 0.77 | 0.73 | 0.81 |
| Poly | 0.80 | 0.76 | 0.84 |

**We select Radial Basis Function for model in Support Vector Classifier**

##### Model Exporting

import pickle

pickle.dump(svc,open('SupportVector.pkl','wb'))

#### DECISION TREE CLASSIFICATION - RANDOM FOREST

**Random Forest** is a powerful and versatile ensemble learning method used for both classification and regression tasks. It builds multiple decision trees and combines their outputs to make more accurate and robust predictions

##### Model

*CodeSnippet*

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()

rfc.fit(X\_train, y\_train)

name = 'Random Forest'

y\_pred = rfc.predict(X\_test)

print(f'{name} Results:')

print('Accuracy:', accuracy\_score(y\_test, y\_pred))

print('Confusion Matrix:\n', confusion\_matrix(y\_test, y\_pred))

print('Classification Report:\n', classification\_report(y\_test, y\_pred))

print('-' \* 50)

n\_estimators: Number of decision trees in the forest.

max\_depth: Maximum depth of each tree.

min\_samples\_split: Minimum number of samples required to split an internal node.

min\_samples\_leaf: Minimum number of samples required to be at a leaf node.

max\_features: Number of features to consider when looking for the best split.

##### Model Performance Metrics

Random Forest Results:

Accuracy: 0.8341708542713567

Confusion Matrix:

[[1054 244]

[ 185 1104]]

Classification Report:

precision recall f1-score support

0 0.85 0.81 0.83 1298

1 0.82 0.86 0.84 1289

accuracy 0.83 2587

macro avg 0.83 0.83 0.83 2587

weighted avg 0.83 0.83 0.83 2587

**Accuracy : 0.83**

**Precision (Negative) : 0.85**

**Precision (Positive) : 0.82**

### REFERENCES

1. <https://scikit-learn.org/stable/user_guide.html>
2. <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>
3. <https://scikit-learn.org/stable/modules/ensemble.html>
4. <https://github.com/ashishpatel26/Andrew-NG-Notes>
5. <https://ieeexplore.ieee.org/document/9249211>