**Report on Telecom Customer Churn Prediction**

**1. Project Overview**

The primary objective of this project is to develop a predictive model that can identify customers at risk of churning for a telecommunications company. By analysing customer data, we aim to enable the company to take proactive measures to retain these customers.

**2. Libraries Used**

**2.1. Warnings:**

Used to Ignore the warnings which are shown at execution of the code.

**2.2. Pandas:**

It is a fast, powerful, flexible and easy to use library for data analysis and manipulation.

**2.3. Numpy:**

Which is mostly used for working with arrays and faster scientific calculations.

**2.4. Matplotlib:**

It is used to create static, animated and interactive visualizations in python.

**2.5. Seaborn:**

Seaborn is built on top of the matplotlib and can create advanced visualizations.

**2.6. Scikit-learn:**

Open-source machine learning library for python, which have built in models like LogisticRegression, RandomForestClassifier, GradientBoostingClassifier etc.

**3. Data Collection and Preprocessing**

The dataset used for this project was sourced from Kaggle and contains information on 7,043 customers, including 21 feature columns and one target column (`Churn`).

The columns of the dataset are as follows:

* customerID: Customer ID
* gender: Whether the customer is a male or a female
* SeniorCitizen: Whether the customer is a senior citizen or not (1, 0)
* Partner: Whether the customer has a partner or not (Yes, No)
* Dependents: Whether the customer has dependents or not (Yes, No)
* tenure: Number of months the customer has stayed with the company
* PhoneService: Whether the customer has a phone service or not (Yes, No)
* MultipleLines: Whether the customer has multiple lines or not (Yes, No, No phone service)
* InternetService: Customer’s internet service provider (DSL, Fiber optic, No)
* OnlineSecurity: Whether the customer has online security or not (Yes, No, No internet service)
* OnlineBackup: Whether the customer has online backup or not (Yes, No, No internet service)
* DeviceProtection: Whether the customer has device protection or not (Yes, No, No internet service)
* TechSupport: Whether the customer has tech support or not (Yes, No, No internet service)
* StreamingTV: Whether the customer has streaming TV or not (Yes, No, No internet service)
* StreamingMovies: Whether the customer has streaming movies or not (Yes, No, No internet service)
* Contract: The contract term of the customer (Month-to-month, One year, Two year)
* PaperlessBilling: Whether the customer has paperless billing or not (Yes, No)
* PaymentMethod: The customer’s payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))
* MonthlyCharges: The amount charged to the customer monthly
* TotalCharges: The total amount charged to the customer
* Churn: Whether the customer churned or not (Yes or No)

The preprocessing steps included:

**3.1. Object to Numeric datatype**: *Totalcharges* is stored in object (String) format, so we are using pd.to\_numeric() function to convert into numeric datatype

**3.2. Handling missing values**: Imputed missing *TotalCharges* values with the median.

**3.3. Outliers**: By using Box plot and quantile() function, we tried to find the outliers. The output showed that there are no outliers form the columns *tenure, MonthlyCharges, TotalCharges.*

**4. Exploratory Data Analysis (EDA)**

**4.1. Initial Summary Statistics**

**Numerical Features:**

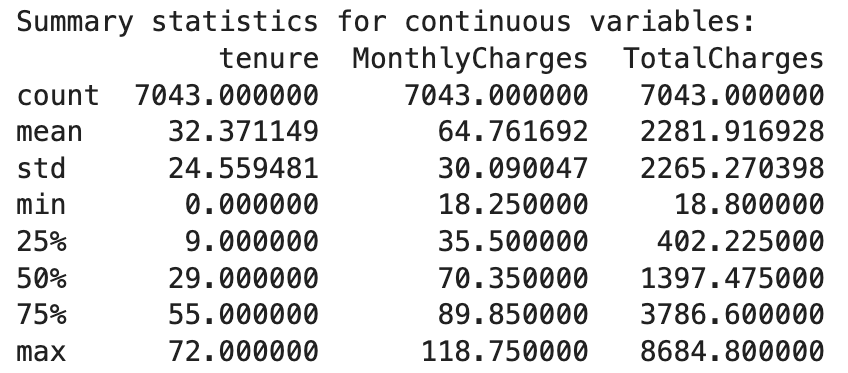
*tenure*: Number of months a customer has stayed with the company.

*MonthlyCharges*: Monthly charges for the customer.

*TotalCharges*: Total charges incurred by the customer.

**Categorical Features**:

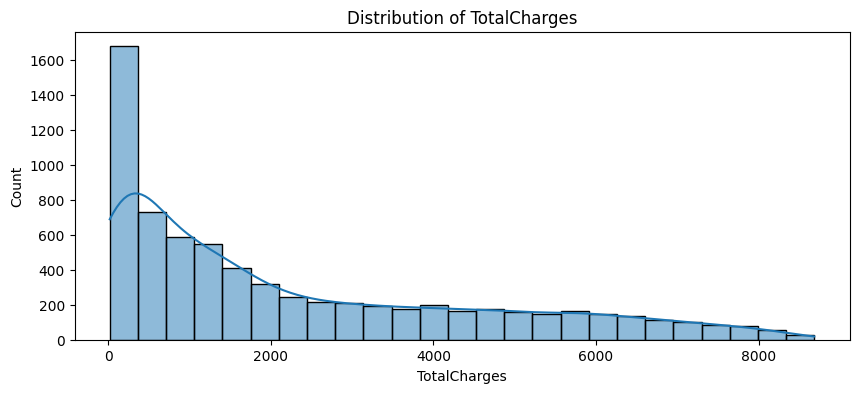
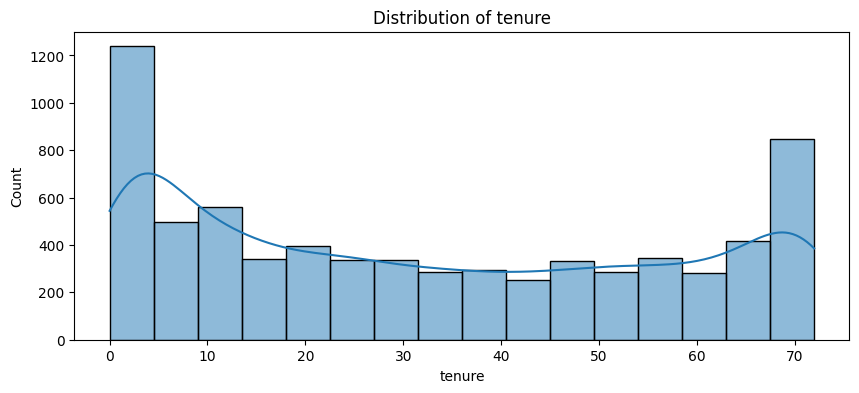
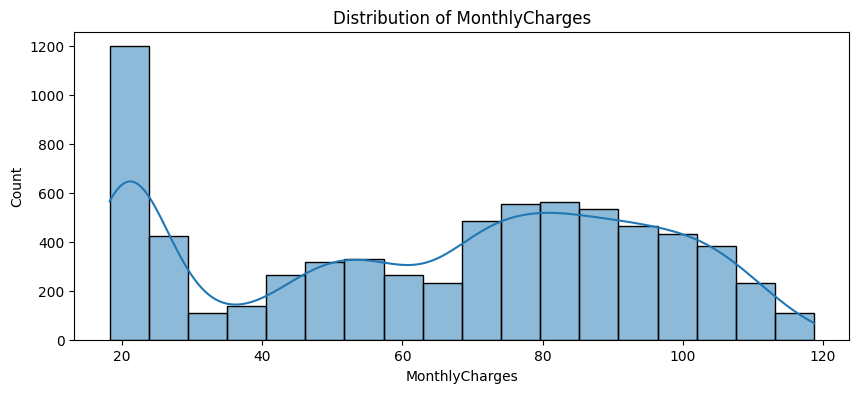
gender, Partner, Dependents, PhoneService, MultipleLines, InternetService, OnlineSecurity, OnlineBackup , DeviceProtection , TechSupport , StreamingTV , StreamingMovies, Contract, PaperlessBilling, PaymentMethod.



From the table, we can see there is a gradual increase in the values, which indicated the data is not having any outliers and data is in good form.

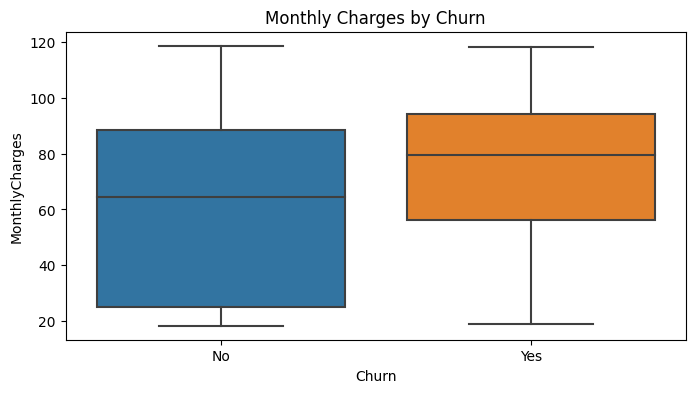
We also checked the Churn percentage from the dataset, which is turn out to be 26.53% of the customers are turned to be churns, which is a huge problem for the company.

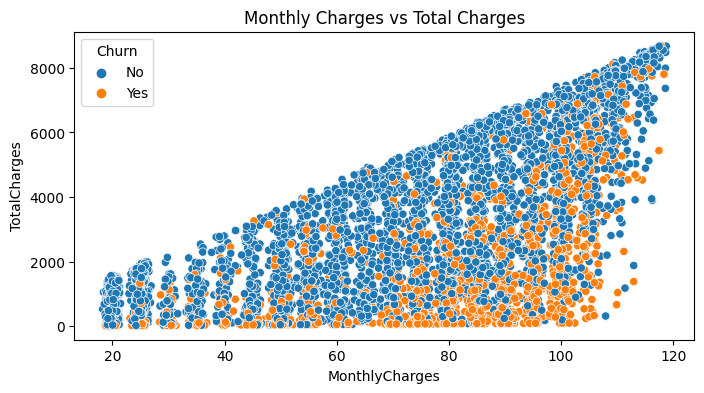
**4.2. Univariate Analysis**

**We are Plotting the distribution graphs for *tenure, MonthlyCharges, TotalCharges* where we got the following insights*:*

1. Tenure: Most of the data is at range 0-10 and 65-70
2. MonthlyCharges: Most of the customers are likely to be at 20-40 Monthly Charges
3. TOtalCharges: Most of the customers in range 0-2000 which shows that customers like to stay at lower charges

**4.3. Bivariate Analysis**

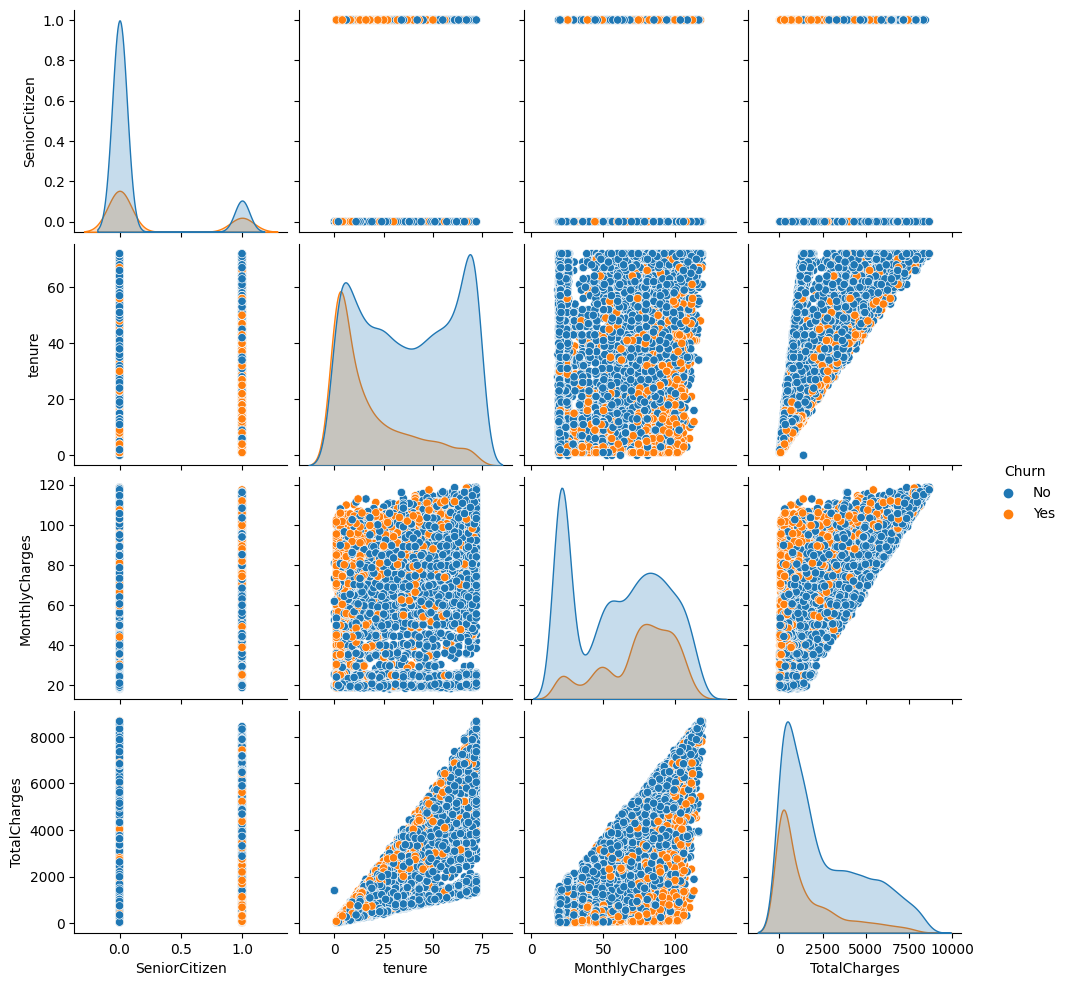
We used box plot on *MonthlyCharges and Churn* variables to see how this both are related.



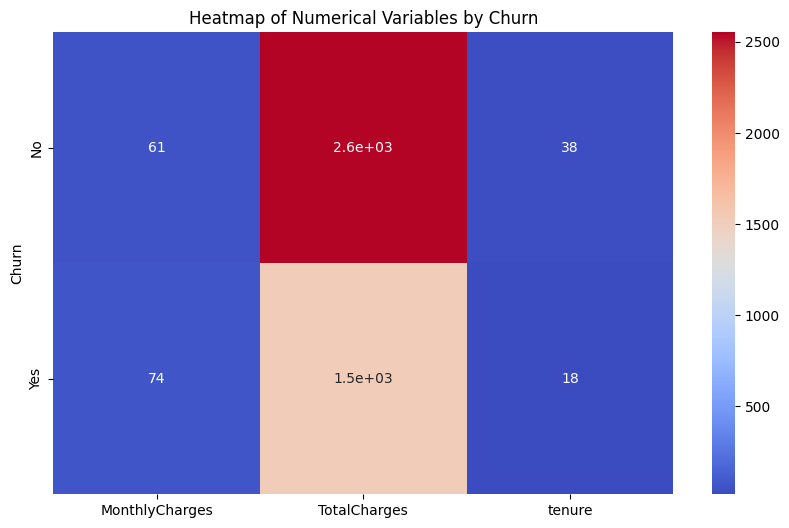
The insights we can get from the two graphs that customers are more likely to turn churn are the one who are having high monthly charges.

**4.4. Multivariate analysis:**

We first used pairplot to understand the pair wise relations between the columns *SeniorCitizen, tenure, MonthlyCharges, TotalCahrges.*

**

And we used Heapmap on *, tenure, MonthlyCharges, TotalCahrges,*  to view the correlation between the columns.



**5. Feature Selection**

**5.1. Conversion of data and datatypes**

**5.1.1. Converting Columns which are binary variables**

Binary variables: This are the columns where the data in this are in form binary i.e yes or no. So inorder to make this data machine readable, we need to convert this into the form of 1 or 0.

We used a function convertToBinary(), which is used to convert the Yes/No into 1/0 values in the columns of the telecom dataset.

**5.1.2. Converting Categorical Columns to numerical columns**

By using get\_dummies() function from pandas, we are converting categorical variables into numerical as one-hot encoding the categorical variables.

The following columns are underwent the one-hot encoding:

'gender', 'MultipleLines','InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaymentMethod'.

**5.2. Drop unnecessary columns:**

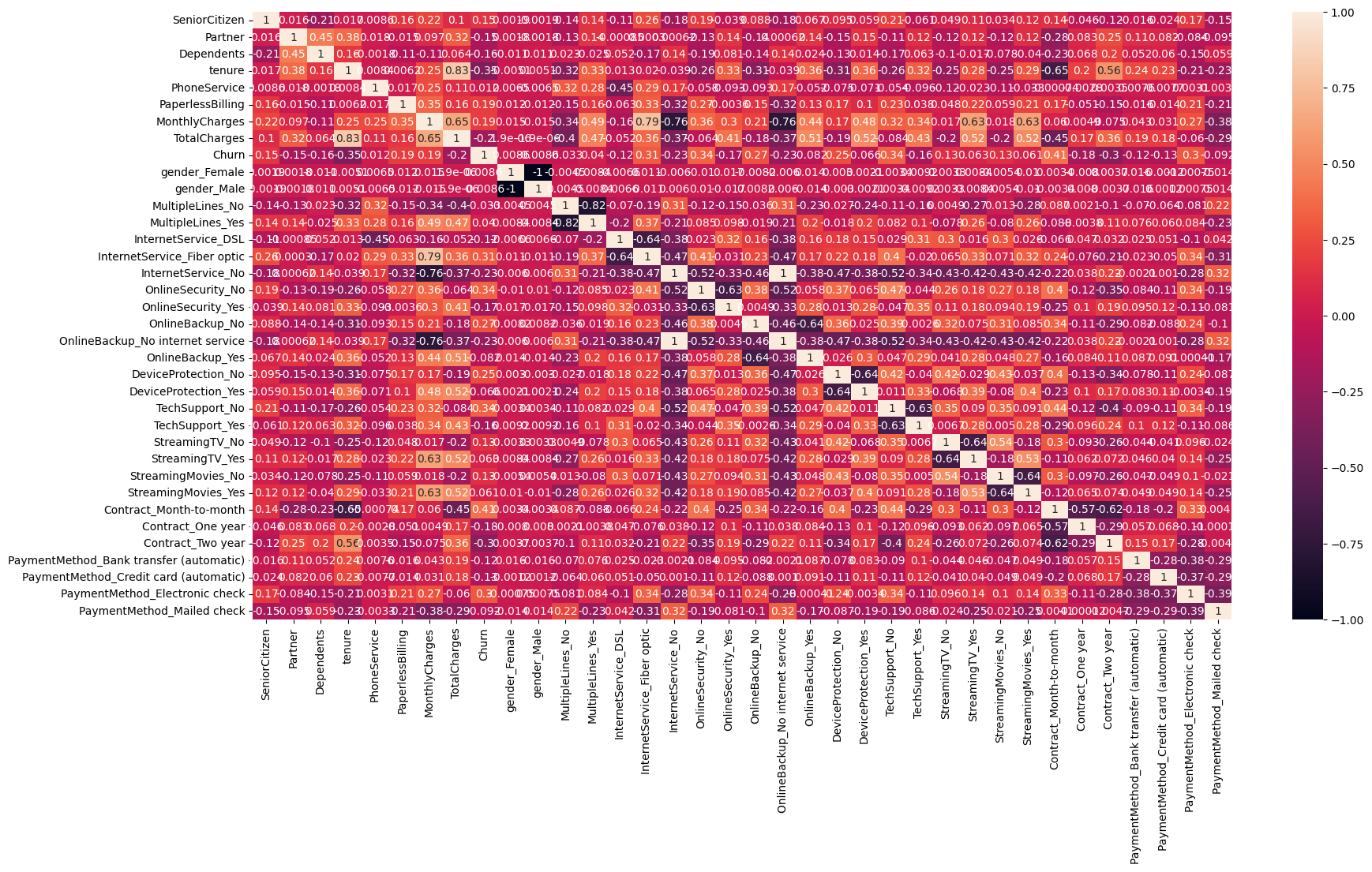
After performing one-hot encoding, we need to remove any unnecessary columns which does not give any importance in model training.

The following are the columns which are not much important, which we can say by observing the variables:

'customerID', 'MultipleLines\_No phone service', 'OnlineSecurity\_No internet service', 'DeviceProtection\_No internet service', 'TechSupport\_No internet service', 'StreamingTV\_No internet service', 'StreamingMovies\_No internet service'

**5.3. Correlation Analysis:**

By using heatmap(), we can see which variables are correlated to which other variables. High correlation variables are not much use for the model training, because the correlated variables have same type of ordered values, which leads to passing two same order variables 2 times to the model training.



After careful observation, we can say that the following variables are highly correlated and makes no meaning to keep them for model training:

*'gender\_Female', 'Contract\_Month-to-month', 'InternetService\_Fiber optic', 'MultipleLines\_No','OnlineSecurity\_No', 'OnlineBackup\_No', 'DeviceProtection\_No', 'TechSupport\_No','StreamingTV\_No', 'StreamingMovies\_No', 'OnlineBackup\_No internet service'*

**5.4. Feature Scaling:**

We need to standardise the continuous values, as they will cause in low accurecy in model training.

We are using sklearn StandardScaler() on the continuous variables i.e. *'tenure', 'MonthlyCharges', 'TotalCharges'.*

Viewing the first 5 rows of the *'tenure', 'MonthlyCharges', 'TotalCharges':*

A screenshot of a graph

Description automatically generated

**6. Model Train-Test Split**

We are storing the all input data variables in X variable and in y we are storing the *churn* variable.

By using sklearn.model\_selection -> train\_test\_split, we are dividing the X, y variables into 4 variables i.e.

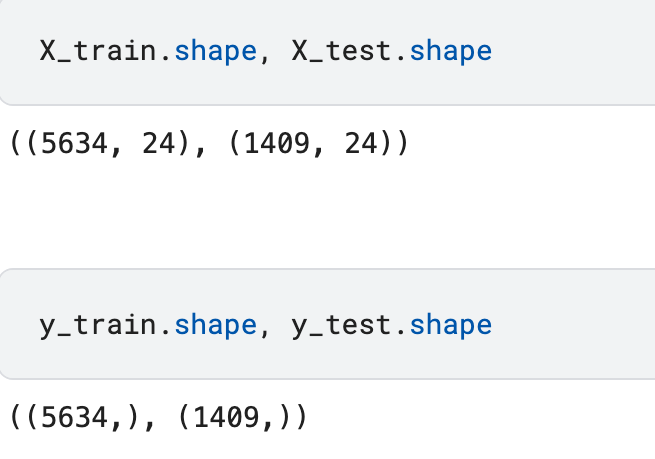
X\_train : which stores the trainable input data, 80% of the data,

X\_test : which stores for testing the model, 20% of the data,

y\_train : Which stores the trainable output data, 20% of the data,

y\_test : Which stores the testing the output model, 20% of the data.

The shape of the variables as follows:



**7. Model Building**

In this section, we will try on training the model on 3 different models i.e.

a. Logistic Regression

b. Random Forest Classifier

c. Gradient Boosting Classifier

**7.1. Logistic Regression**

We defined the lr as LogisticRegression() model, where we imported the Recursive Feature Elimination (RFE) to select only top features for the model training.

By passing lr to RFE(estimator=lr), we are creating an instance of RFE with a logistic regression estimator.

Then after fitting the RFE model to the training data: rfe.fit(X\_train, y-train).

Based on the X\_train.columns[rfe.support\_], we can train our lr model only on selected columns.

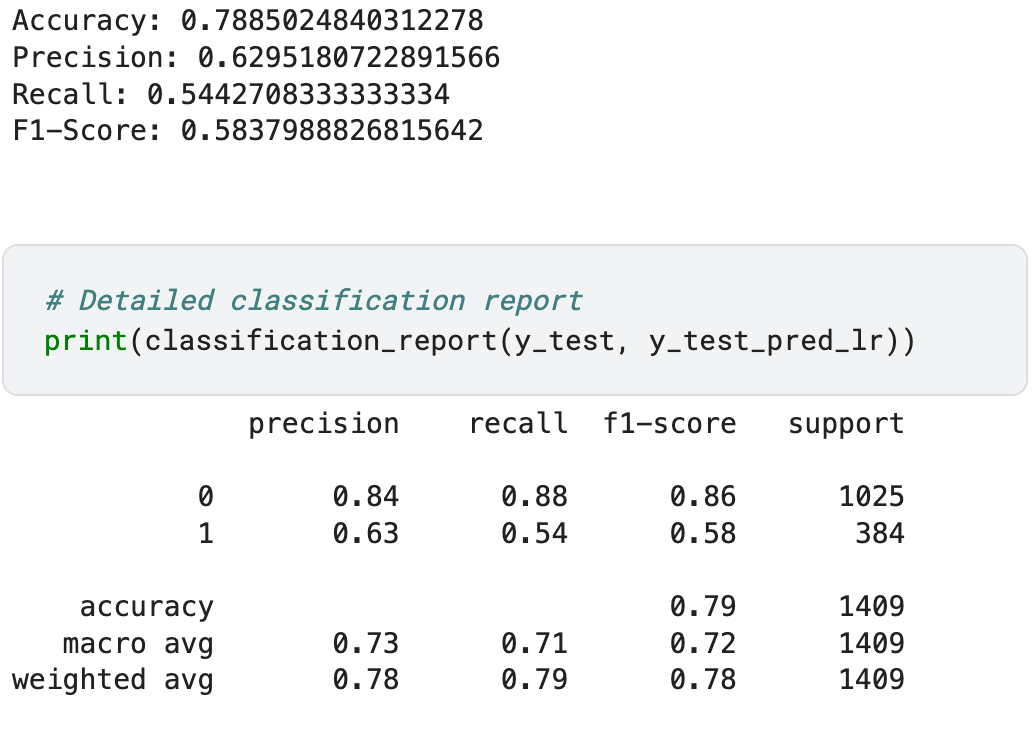
Training the selected columns of X\_train and y\_train, we start predicting the y\_train\_pred\_lr and y\_test\_pred\_lr.

The following data shows the model's Training Accuracy and Test Accuracy:

* Training Accuracy: 0.8052893148739794
* Test Accuracy: 0.7885024840312278

**7.1.1. Model Evaluation**

The following image shows the model's Accuracy, Precision, Recall, F1-Score



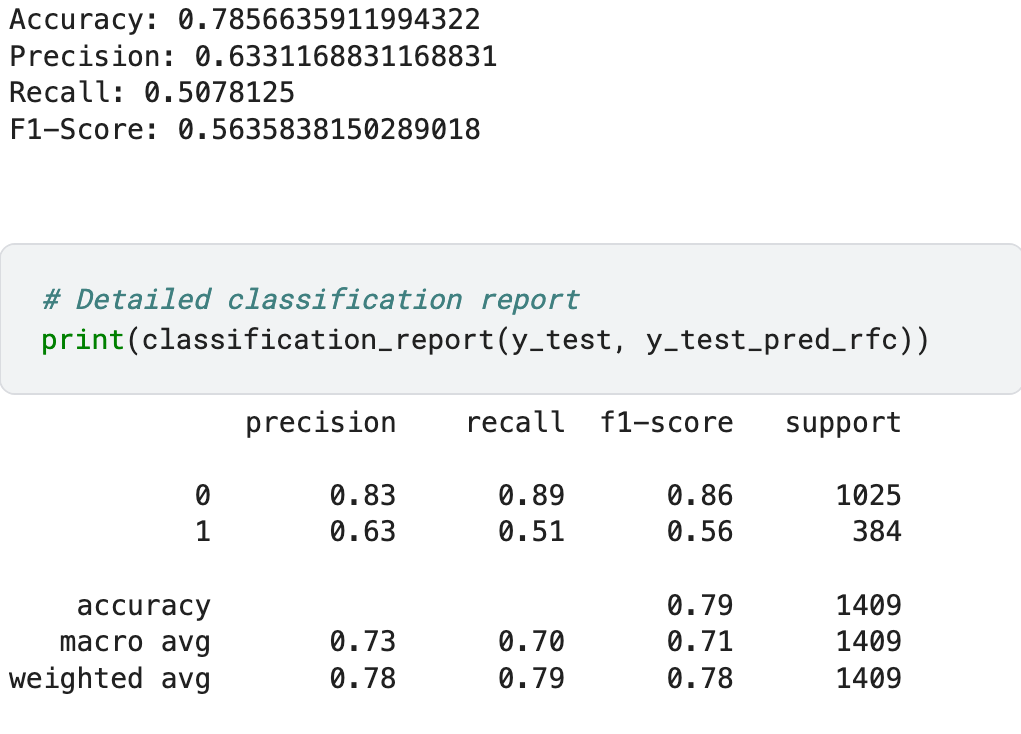
**7.2. Random Forest Classifier**

Similar to Logistic Regression, we are using Recursive Feature Elimination (RFE) to select only top features for the model training. Once we get the selected columns, we train the rfc model on selected columns, and predit the y\_train\_pred\_rfc and y\_test\_pred\_rfc.

The following data shows the model's Training Accuracy and Test Accuracy:

* Training Accuracy: 0.9971600993965212
* Test Accuracy: 0.7856635911994322

**7.2.1. Model Evaluation**

The following image shows the model's Accuracy, Precision, Recall, F1-Score

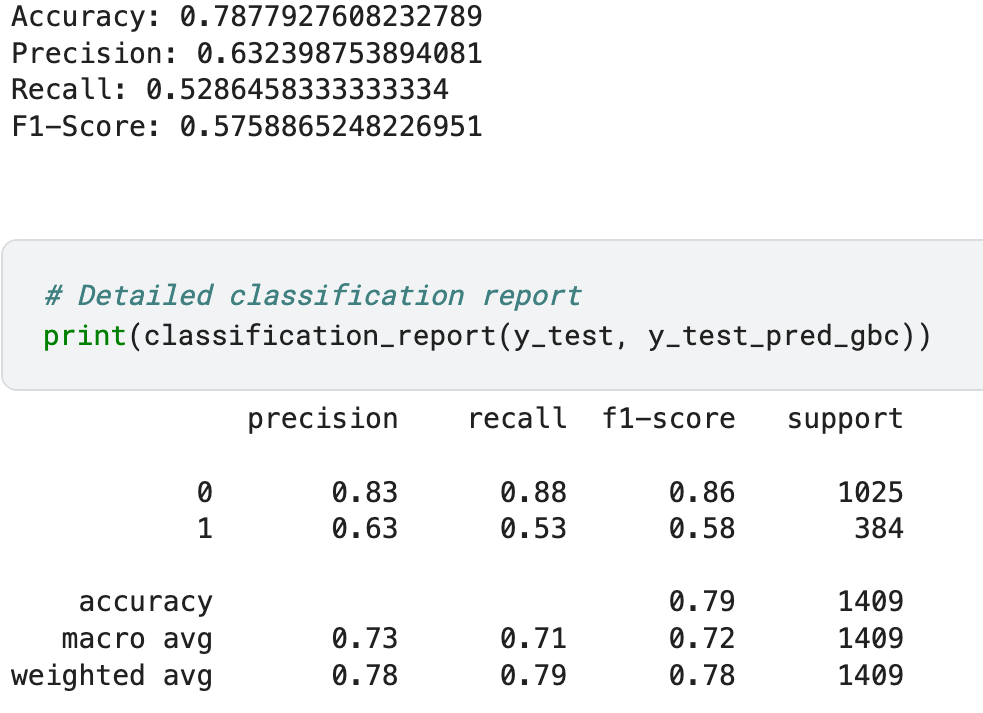
**7.3. Gradient Boosting Classifier**

In this model we initialized a Gradient Boosting Classifier (GBC) with specific hyperparameters and employs Recursive Feature Elimination (RFE) to select the top 13 features that contribute most to predicting the target variable. The GradientBoostingClassifier is configured with 100 trees (n\_estimators), a learning rate of 0.1, and a maximum depth of 3 for each tree, ensuring a balance between model complexity and generalization. The RFE wrapper uses the GBC model to iteratively train and evaluate the importance of each feature, eventually retaining the 13 most significant ones. These selected features are then used to fit the GBC model and make predictions on both the training and test datasets, aiming to enhance the model’s performance by focusing on the most relevant data.

The following data shows the model's Training Accuracy and Test Accuracy:

* Training Accuracy: 0.8272985445509408
* Test Accuracy: 0.7877927608232789

**7.3.1. Model Evaluation**

The following image shows the model's Accuracy, Precision, Recall, F1-Score

**8. Challenges Faced**

1. **Imbalanced Data**: The target variable (`Churn`) was imbalanced. Techniques like stratified sampling and evaluation metrics suitable for imbalanced data (precision, recall) were employed.

2. **Feature Selection**: Balancing the trade-off between feature relevance and redundancy required multiple iterations of feature selection methods.

3. **Model** **Overfitting**: Ensured model generalizability by tuning hyperparameters and using cross-validation techniques.

**8. Conclusion**

The project successfully developed a churn prediction model with significant accuracy and reliability. Key insights from EDA helped in feature engineering and improving model performance. The model can assist the telecommunications company in implementing targeted retention strategies, potentially reducing churn rates and improving customer satisfaction.