Predicting Telco Customer Churn Using Machine Learning Models

Introduction: -

Customers were the heart of the telecommunication industries. Their profit, loss, stock market, and company value is fully dependent on the customers they have. Many companies were trying to make new customers, so that the assets of the company will increase. And also the major factor is that the obtained customers should be satisfied or continue with company services. Retaining the existing customers is always a most cost-effective than getting new customers. For that, company needs to perform various analysis of their services which makes the customer ease at it and which services not provided any profit, what are the changes need to keep the customers. By performing various strategies and understanding the factors which contribute to customer churn and design a model that accurately predict which customers are likely to leave can significantly enhance a company's ability to implement effective retention strategies. The machine learning models like Logistic Regression, Random forest, Support Vector Machines and K-nearest neighbour algorithms were used. For predicting the accuracy of the model the metrics like F1-score, Precision, Recall, accuracy score and confusion matrix were determined.

Description of dataset: -

For the prediction of customer churn in telecommunication Kaggle's "Telco Customer Churn" dataset is used, which is the dataset of IBM Sample dataset. The main context of this dataset is to analyse all relevant customer data and develop focused customer retention program.

Analysis of dataset: -

The dataset contains information of 7043 customer data with the 21 features in it. The target of the dataset is the churn feature. The dataset is about the demographic information of the customers, customer account information and the services for which the customers choose. The dataset has the customer id of 7043 unique ids' which represents the count of the customers. And the attributes in the dataset are,

- **Gender:** Indicates the gender of the customer (Male/Female).
- **SeniorCitizen:** Indicates if the customer is a senior citizen (0: No, 1: Yes).
- **Partner:** Indicates if the customer has a partner (Yes/No).
- **Dependents:** Indicates if the customer has dependents (Yes/No).
- **PhoneService:** Indicates if the customer has phone service (Yes/No).
- **MultipleLines:** Indicates if the customer has multiple lines (No phone service, No, Yes).
- **InternetService:** Indicates the type of internet service (DSL, Fiber optic, No).
- **OnlineSecurity:** Indicates if the customer has online security (No internet service, No, Yes).
- **OnlineBackup:** Indicates if the customer has online backup (No internet service, No, Yes).

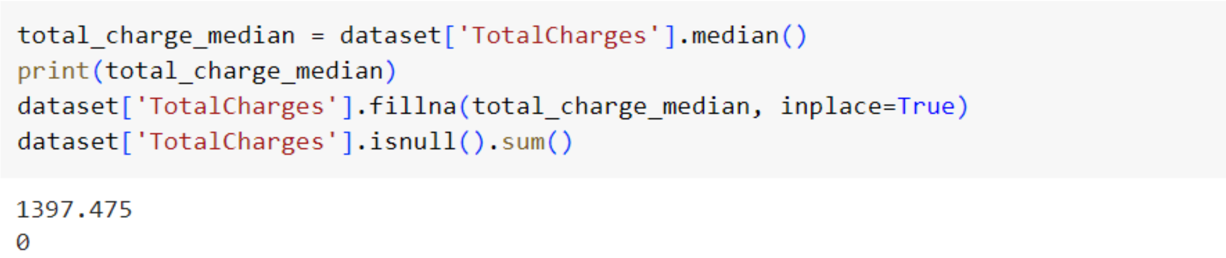
- **DeviceProtection:** Indicates if the customer has device protection (No internet service, No, Yes).
- **TechSupport:** Indicates if the customer has tech support (No internet service, No, Yes).
- **StreamingTV:** Indicates if the customer has streaming TV (No internet service, No, Yes).
- **StreamingMovies:** Indicates if the customer has streaming movies (No internet service, No, Yes).
- **CustomerID:** A unique identifier for each customer.
- **Contract:** Indicates the contract type (Month-to-month, One year, Two year).
- **PaperlessBilling:** Indicates if the customer uses paperless billing (Yes/No).
- **PaymentMethod:** Indicates the payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic)).
- **Tenure:** Indicates the number of months the customer has stayed with the company.
- **MonthlyCharges:** The amount charged to the customer monthly.
- **TotalCharges:** The total amount charged to the customer.

Among the dataset, there are 17 objective features, 2 integer features and one float feature.

The TotalCharges feature is the total amount charged to customers for their services. Which is the numeric data which is provided as the objective data. And the feature has 11 empty rows. First we need to replace the empty strings to null values using the numpy nan function. After changing to null values we need to convert the objective datatype to numeric datatype. And for the missing values we need to calculate the median of the values in the dataset and then replace the missing values to the median values.

Why median?

The median is the measure of central tendency that represents the middle value of a dataset when it is ordered from smallest to largest. If the values in the dataset is in the range of small difference, then we need to use the mean value which is effective. But here in the dataset has bigger difference between the data's. The mean value will provide higher value. So we are using the median value here. Which is very effective when the difference between the variables is higher.



```
total_charge_median = dataset['TotalCharges'].median()
print(total_charge_median)
dataset['TotalCharges'].fillna(total_charge_median, inplace=True)
dataset['TotalCharges'].isnull().sum()
```

```
1397.475
0
```

Figure 1. Filling the missing values.

Data Analysis of Objective dataset: -

As seen above the dataset has 17 objective features in it. We need to separate the features for better analysis of the dataset. As the analysis of the data, the first step is to check how much different values or the distribution of the values in the column. For this the unique values of the column were extracted and the count of the value is plotted in the pie chart for the better understanding of the dataset which contributes the target.

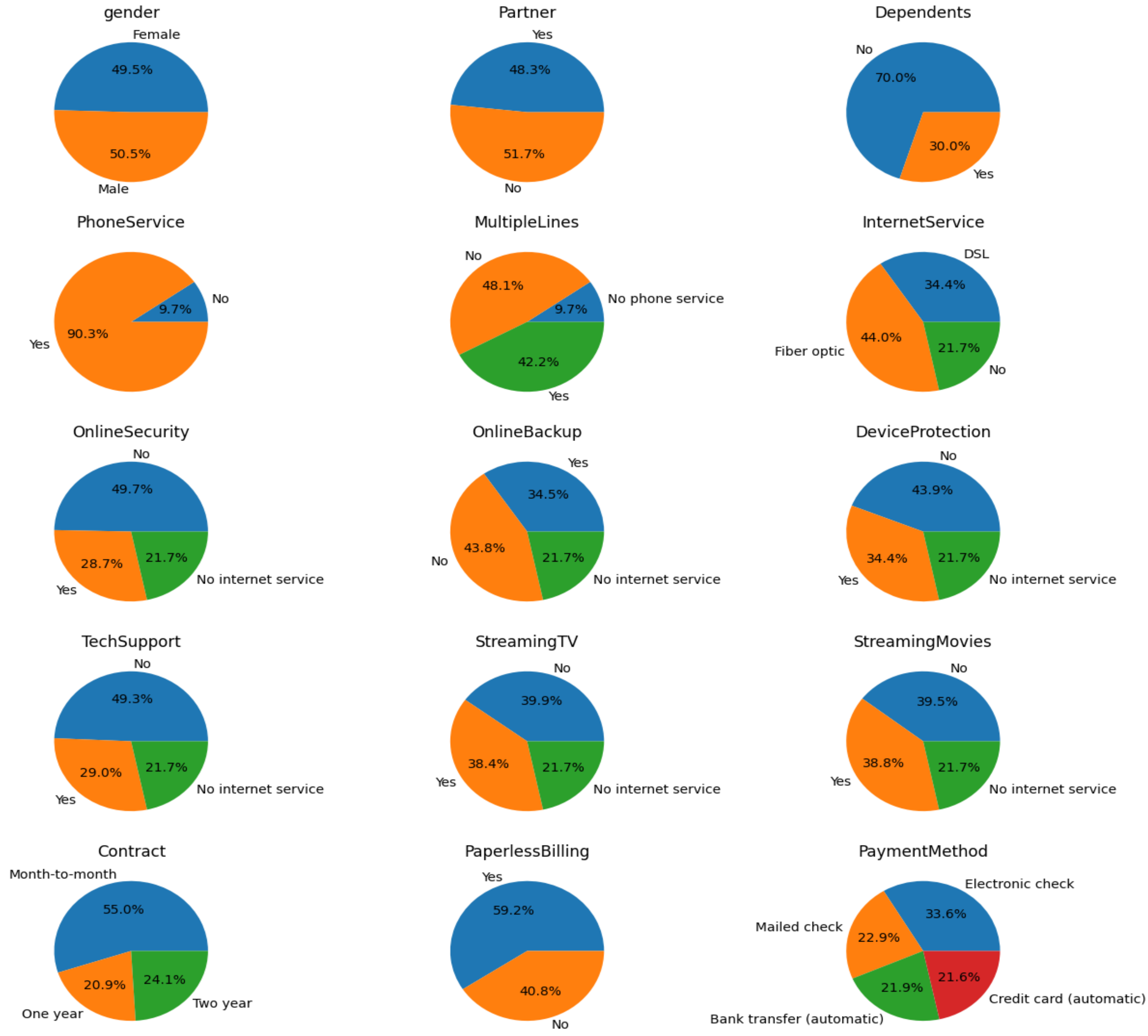


Figure 2. Distribution of the Objective dataset.

For example, the column PaymentMethod has four different features namely Electronic check, Mailed check, Bank transfer, and Credit Card. By analysing it we came to know that the distribution of each variable contributes 33.6%, 22.9%, 21.9% and 21.6%.

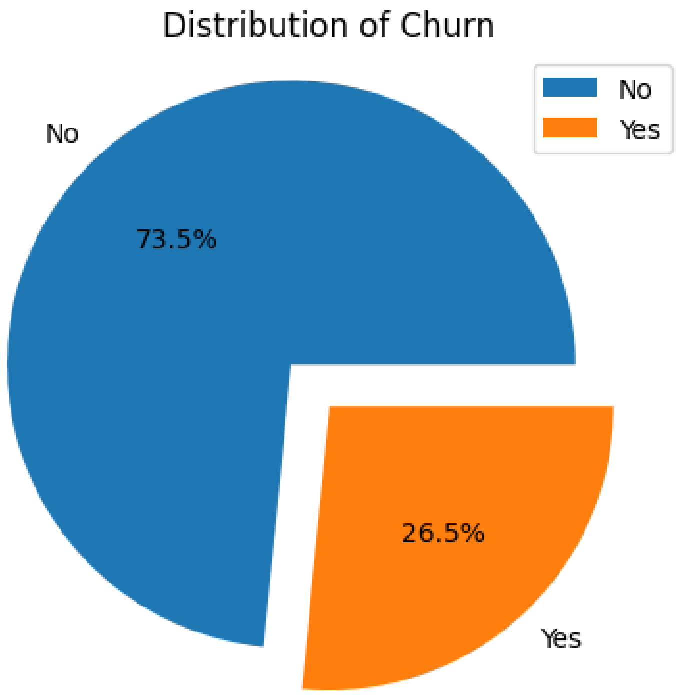


Figure 3. Target distribution.

The above diagram explains the distribution of target for 2 classes. The **“Yes” class** being the negative class with the percentage of **26.5%** in total datasets. The **Positive class “NO”** is on the other hand has the distribution of **73.5% classes**. By the above chart we can clearly see that the nearly 75% of the dataset lies on the class “NO” it **may resultant to the model to bias on one sided**. This may be the major drawback which may occur.

In the below figures 4 & 5, the distribution of the data over the target yes and no is visualized separately. For this the data was separated into two by filtering churn as yes and churn as no. This helps to analyse the data based on the classes and contribution of the dataset.

For example, from the figures 4 and 5, we can observe that, in “Yes” target class the OnlineSecurity feature says that, the customer has **not onlinescurity the churn rate is higher.** From this we can came to know that there are some drawbacks in the OnlineSecurity feature. And second example, if the **customer has the dependent** then the percentage of them to churn is **higher** than the independent. Likewise, we can easily analyse each class data’s and came to know the fault in the service structure.

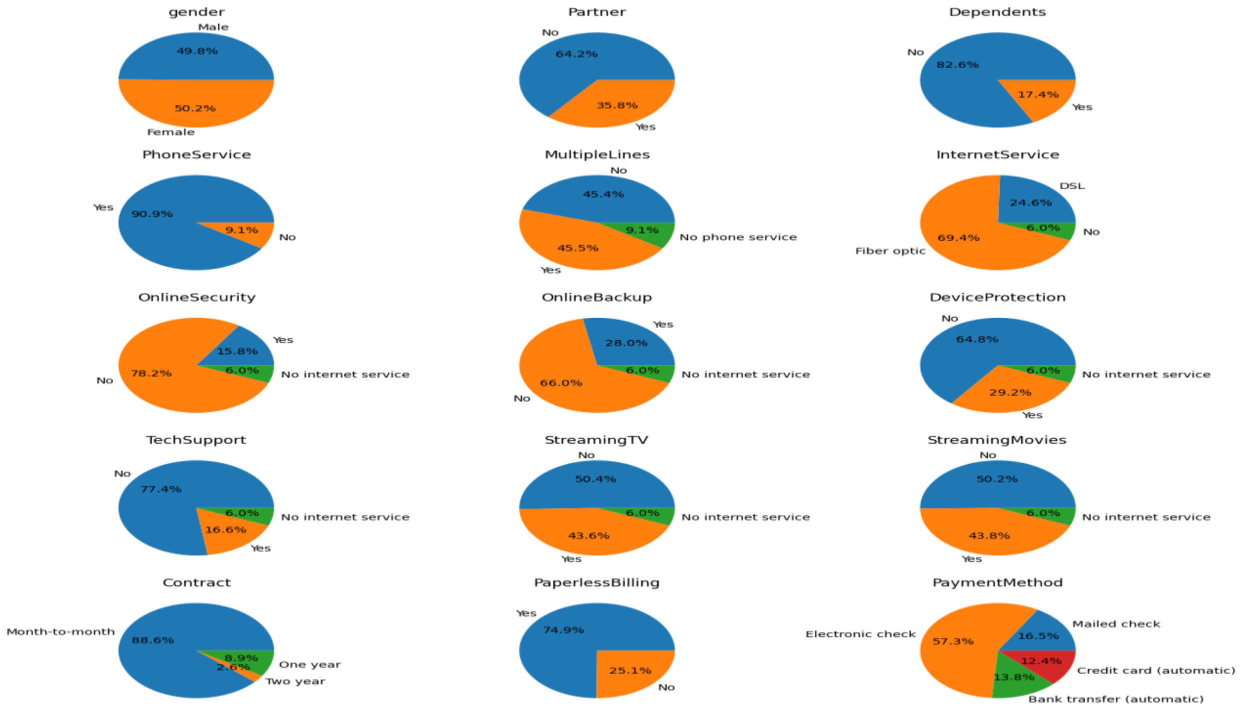


Figure 4. Distribution of the dataset over the class “Yes”.

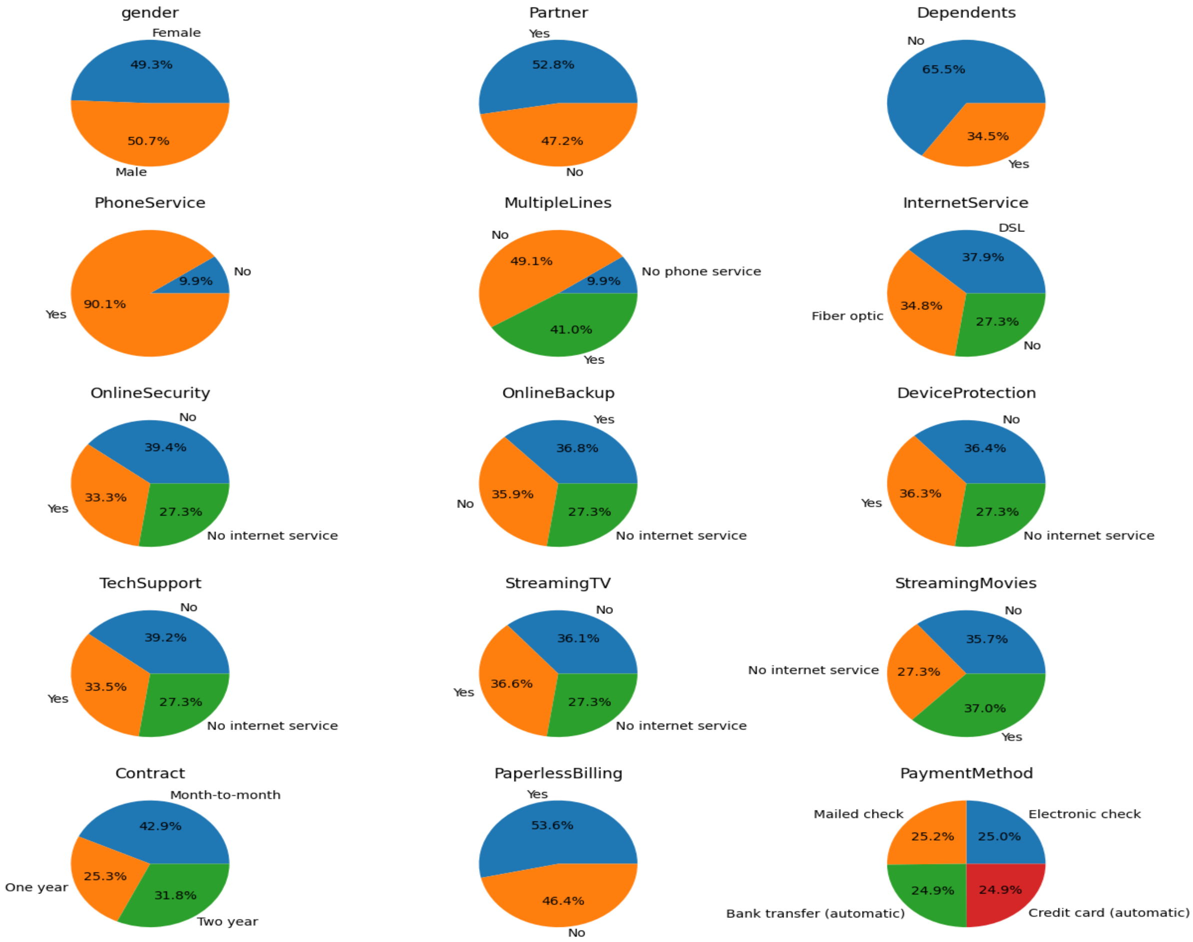


Figure 5. Distribution of dataset with “No” class.

Data Analysis of Numeric dataset: -

As stated above that there are four different numeric datasets which is SeniorCitizen, tenure, MonthlyCharges, TotalCharges. By separating it, we can more deeply analyse the structure and overview of the model.

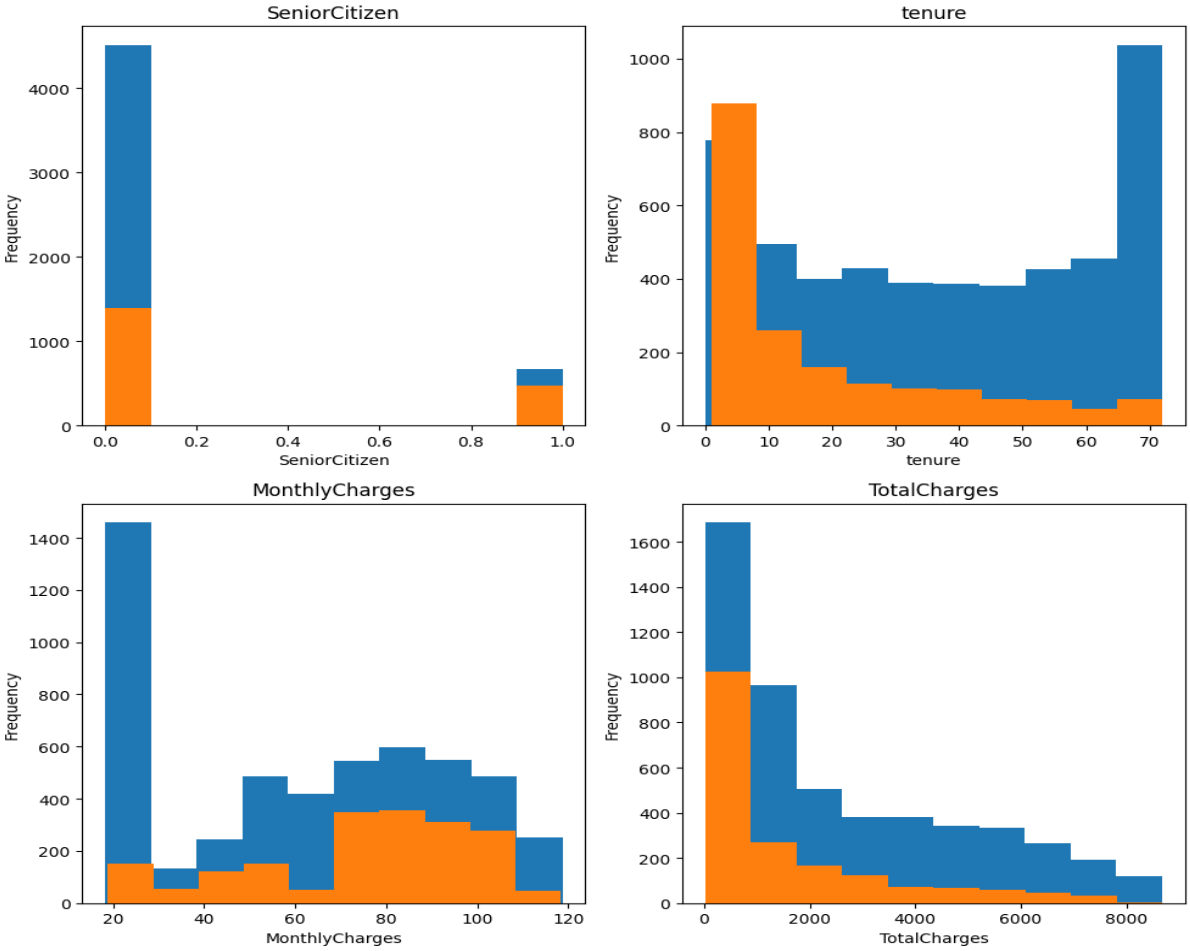


Figure 6. Distribution of numeric datatype dataset.

From the figure 6, we came to know that the distribution of these dataset. The use of visualizing is that, the SeniorCitizen feature or column is only having two classes 0 and 1. Which is representing as yes and no separately. Which is a objective type, but it was in numeric form. Here we need to convert the columns to numeric anyway so we are ignoring the conversion now.

Encoders: -

The neural network model can’t work on the categorical models. It can only handle the numeric values. Here the dataset contains 17 columns of non-numeric data. We need to convert everything to the numeric data. so that the model will run without any interruption.
Here I have used two encoders namely LabelEncoder and One-Hot Encoder methods.

Label Encoder: -

LabelEncoder is a method which will converts the classes into the numeric values. The label encoders transform data that represents the categorical values into integers. For example, the gender has two values Male and Female. It convert these values into 0 and 1. 0 represents the Male and 1 as Female.

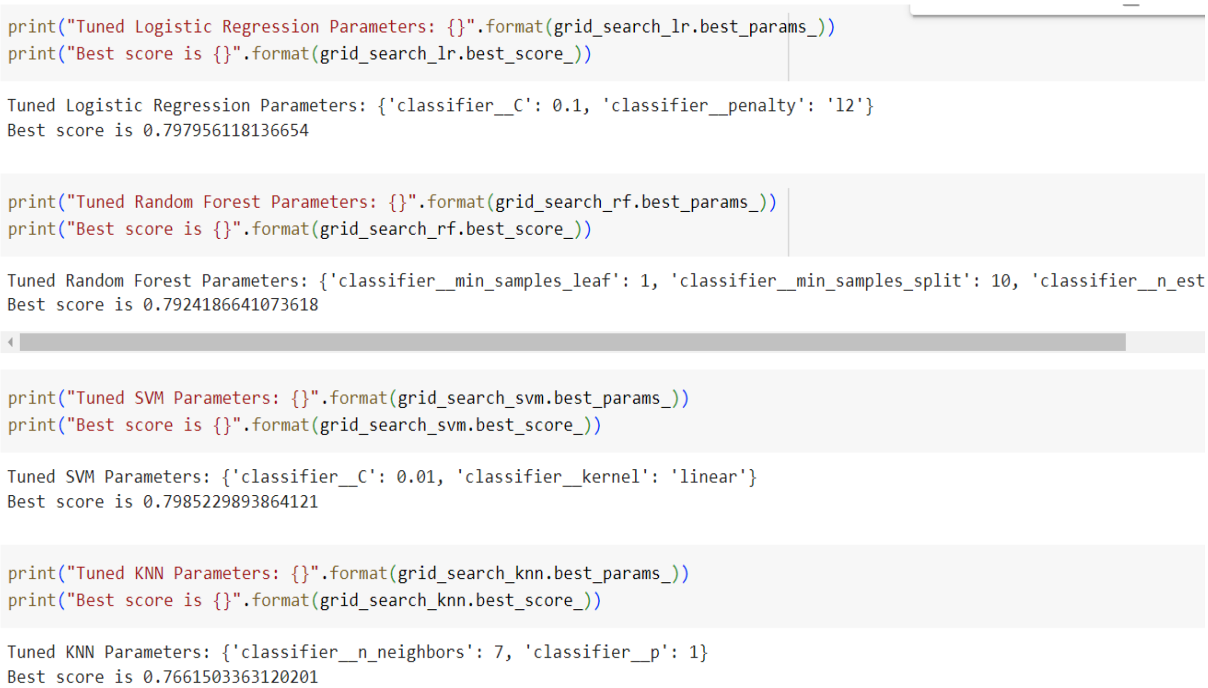
One-hot encoder: -

Like the Label Encoder the one-hot encoder is also doing same tasks. One difference is that unlike the label encoder it converts the categorical dataset into vectors of numbers. For example, the gender has two values Male and Female. It converts these values into [1,0] for male and [0,1] for female.

Models Used: -

GridSearchCV: -

Hyperparameter tuning is the process of selecting the optimal values for a machine learning model's hyperparameters. Hyperparameters are the one which controls the learning process of the models, with changing the parameters while learning. GridSearchCv is one of the hyperparameter tuning methods which is working like a brute force method. It will search for all of the possible combinations with the passed parameters. For example, the SVM model's parameters the kernel has four ['linear', 'poly', 'rbf', 'sigmoid'] and C parameter as [0.001, 0.01, 0.1, 1]. As there are 4 different parameters in kernel and C, the combination of models built is 16. Among the 16 models it will fetch the best performing model and produce the output.



```
print("Tuned Logistic Regression Parameters: {}".format(grid_search_lr.best_params_))
print("Best score is {}".format(grid_search_lr.best_score_))
```

```
Tuned Logistic Regression Parameters: {'classifier__C': 0.1, 'classifier__penalty': 'l2'}
Best score is 0.797956118136654
```

```
print("Tuned Random Forest Parameters: {}".format(grid_search_rf.best_params_))
print("Best score is {}".format(grid_search_rf.best_score_))
```

```
Tuned Random Forest Parameters: {'classifier__min_samples_leaf': 1, 'classifier__min_samples_split': 10, 'classifier__n_est
Best score is 0.7924186641073618
```

```
print("Tuned SVM Parameters: {}".format(grid_search_svm.best_params_))
print("Best score is {}".format(grid_search_svm.best_score_))
```

```
Tuned SVM Parameters: {'classifier__C': 0.01, 'classifier__kernel': 'linear'}
Best score is 0.7985229893864121
```

```
print("Tuned KNN Parameters: {}".format(grid_search_knn.best_params_))
print("Best score is {}".format(grid_search_knn.best_score_))
```

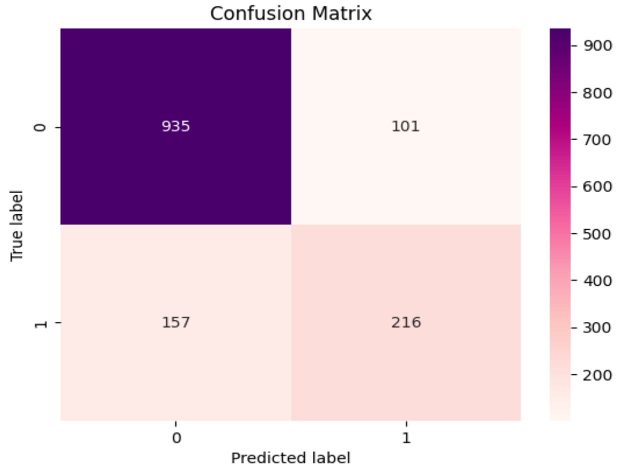
```
Tuned KNN Parameters: {'classifier__n_neighbors': 7, 'classifier__p': 1}
Best score is 0.7661503363120201
```

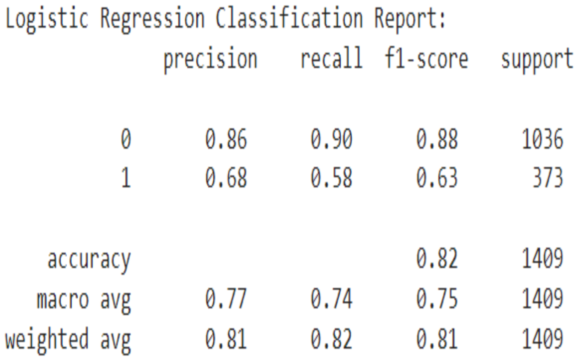
Figure 7. Best parameters of the models that are producing highest accuracy.

Logistic Regression: -

Logistic Regression is the simpler and more effective method for binary and linear classification tasks. It is a classification model, which is very easy to realize and achieves very good performance with linearly separable classes. the logistic regression maps a weighted linear combination of features into real values between 0 and 1. These real values can be interpreted as probabilities. rather than predicting a class, predicting a probability of belonging to a given class of data.

By training the model with the GridSearchCv hyperparameter tuning method, we came to know that the parameters C = 0.1 and penalty = l2 has produced best results.





```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.90      0.88      1036
           1       0.68      0.58      0.63       373

    accuracy                           0.82      1409
   macro avg       0.77      0.74      0.75      1409
weighted avg       0.81      0.82      0.81      1409
```

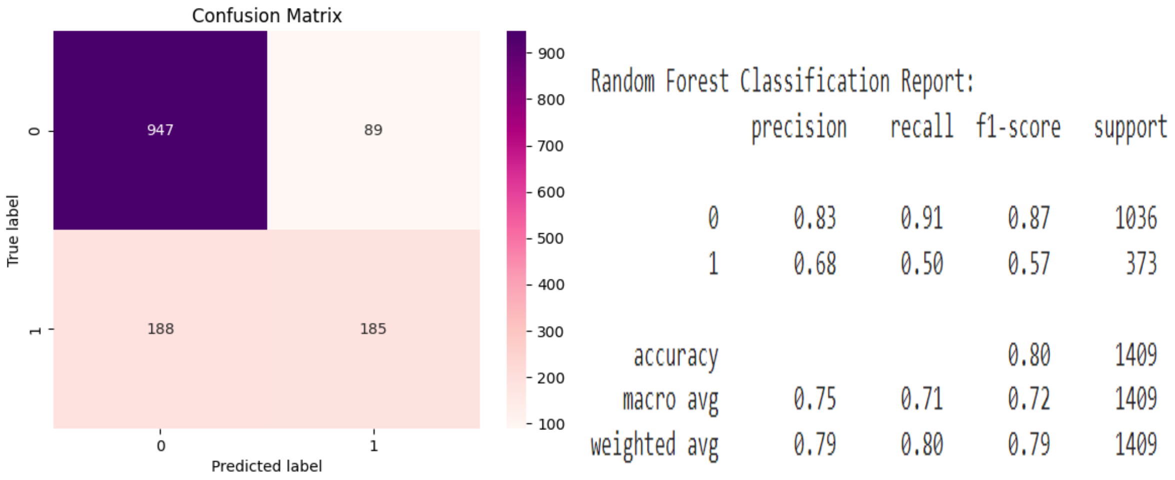
Figure 8. Confusion matrix and classification report of Logistic Regression

Random Forest Algorithm: -

A prominent tree approach to learning for machine learning applications is the Random Forest algorithm. During the training phase, it generates a predetermined number of Decision Tree algorithms. To measure a random subset of features in each division, a random subset of the dataset is used in the construction of each tree. Each decision tree have different datasets by using the bagging methods.

Bagging is a technique it will select the different training sets randomly from the training data with the with-replacement technique. So that at every decision tree in the architecture will get the different datasets. It will help the model to increase the accuracy and also avoids the overfitting the model. In bagging concept, the trees were connected in parallel and each trees will perform its task simultaneously. And at the final stage all output of the trees were grouped together. Before producing the output, it will analyse the grouped output and decides the output by certain tasks like majority voting, Averaging, Stacking, Feature level Fusion, Decision level fusion. Here the random forest is using majority voting. The majority voting means higher occurrence of the output will be stated out as the output.

From the GridSearchCV, we can came to know the beast parameters are min sample leaf = 1, min samples split = 10, n_estimators = 300.



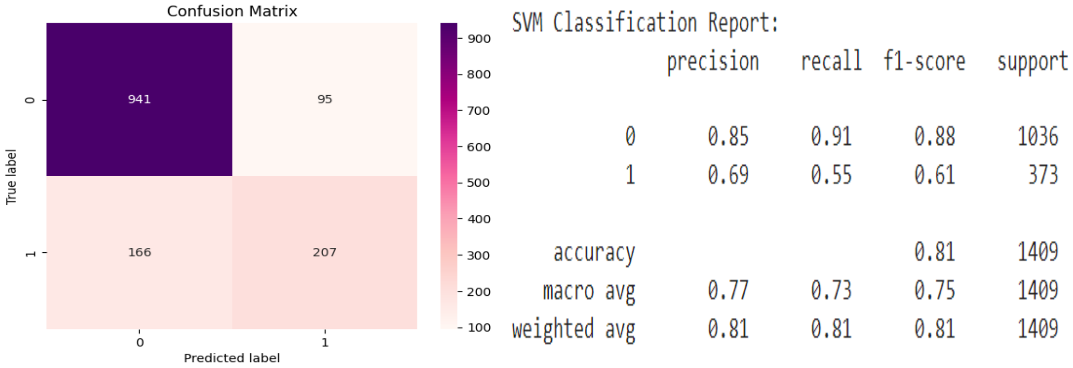
```
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      1036
           1       0.68      0.50      0.57       373

    accuracy                           0.80      1409
   macro avg       0.75      0.71      0.72      1409
weighted avg       0.79      0.80      0.79      1409
```

Figure 9. Confusion matrix and classification report of Random Forest.

Support Vector Machines: -

A support vector machine (SVM) is a supervised machine learning algorithm that classifies data by finding an optimal line or hyperplane that maximizes the distance between each class in an N-dimensional space. The main objective of the SVM is find the optimal hyperplane that can separate the dataset classes.

From the GridSearchCV, we can came to know the beast parameters are C=0.01, kernel = linear.



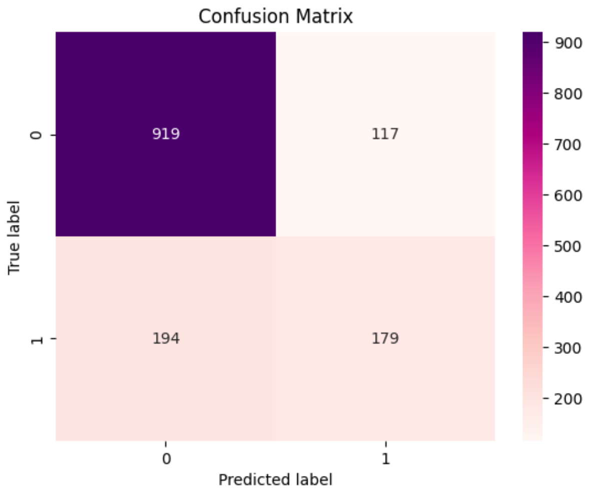
```
SVM Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.91      0.88      1036
           1       0.69      0.55      0.61       373

    accuracy                           0.81      1409
   macro avg       0.77      0.73      0.75      1409
weighted avg       0.81      0.81      0.81      1409
```

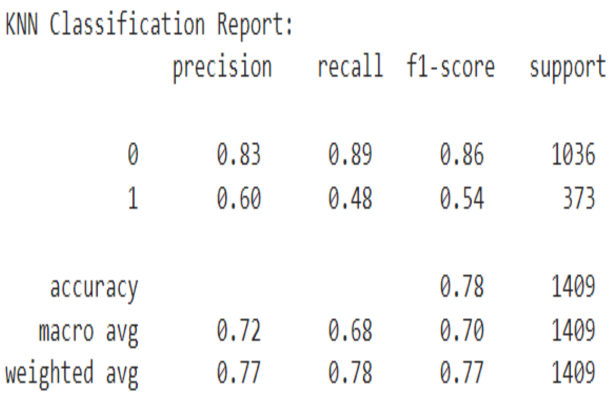
Figure 10. confusion matrix and classification report of SVM

KNN: -

Because of its simplicity and ease of usage, the K-NN) algorithm is a popular and adaptable machine learning technique. No presumptions on the distribution of the underlying data are necessary. It is a versatile option for a range of dataset types in classification and regression applications because it can handle both numerical and categorical data. This non-parametric technique bases its predictions on how similar the data points in a particular dataset are to one another. By comparison, K-NN is less susceptible to outliers than other algorithms.

The best parameters that are determined from the hyper parameter tuning is, n_neighbours = 7, p = 1.





```
KNN Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.89      0.86      1036
           1       0.60      0.48      0.54       373

    accuracy                           0.78      1409
   macro avg       0.72      0.68      0.70      1409
weighted avg       0.77      0.78      0.77      1409
```

Figure 11. Confusion matrix and classification matrix of KNN.

Result: -

In this project, the churn prediction of Telco customer churn dataset was used after the pre-processing. The pre-processed dataset was then passed to the machine learning models like Logistic Regression, Random Forest, Support Vector Machines, K-nearest neighbour. The accuracy of the model is determined with the metrics of F1-score, precision, recall, accuracy score, confusion matrix, and classification report. The models produced the accuracy of 81.6%, 81.4%, 77.9% and 80.3% respectively. From these we came to know that the logistic regression produces the highest accuracy of 81.6%. In future work, as stated above the distribution of dataset over yes is very less. We can increase the distribution and can increase the models accuracy.