Analyzing the role of machine learning models in customer segmentation and churn prediction

Course No. : S2-24_MBAZG622T

Course Title:

Final Semester Project Report

Student Name: Manish Deore

BITS ID: 2023mb21412

Program: MBA in Business analytics

Work Integrated Learning Programmes Division (WILPD),

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI,

VIDYA VIHAR, PILANI, RAJASTHAN - 333031.

(July,2025)

Acknowledgment

It is with immense gratitude that I acknowledge the invaluable support and guidance received throughout the course of this project. The successful completion of this work would not have been possible without the encouragement, assistance, and contributions of several individuals and institutions.

I would like to express my sincere gratitude to all those who supported me throughout the course of this project. First and foremost, I am deeply thankful to my mentor/supervisor **Mrs. Manisha Thakur** and External examiner **Mr. Puneet Singh** for their valuable guidance, insightful feedback, and continuous encouragement during every phase of the project.

I also extend my appreciation to BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI for providing the resources and support necessary to carry out this work. Special thanks to my Faculty Mentor **Mr. Sarveshwar Kumar Inani** for their collaboration, ideas, and constructive discussions that greatly contributed to the improvement of this project.

On a personal note, I would like to thank my **family and friends** for their patience, encouragement, and emotional support during the highs and lows of this journey. Their belief in me kept me motivated and focused.

Lastly, I would like to acknowledge the use of publicly available datasets and open-source tools that made the development and experimentation of the machine learning models possible.

Abstract

Customer retention is a critical aspect of sustainable business growth, particularly in highly competitive industries such as telecommunications, banking, and e-commerce. This project leverages machine learning techniques to predict customer churn — the likelihood that a customer will discontinue using a company's products or services. Using historical customer data including demographics, service usage patterns, and account information, various classification algorithms such as Logistic Regression, Random Forest, and XGBoost were applied to identify at-risk customers. The models were evaluated using metrics like accuracy, precision, recall, and the F1-score, with XGBoost achieving the highest performance. The project demonstrates how predictive analytics can help businesses proactively engage at-risk customers, reduce churn rates, and enhance customer lifetime value.

A typical churn prediction project involves several steps, including data collection, pre-processing, feature engineering, model selection, hyper parameter tuning, model evaluation, and deployment. The data must be pre-processed and engineered to extract more relevant features, after which the best machine learning model is selected, tuned and evaluated on a holdout dataset. Finally, the model is deployed in a production environment to identify at-risk customers and take action to reduce churn.

Churn prediction can help businesses gain valuable insights into their customer base and take proactive steps to reduce churn and improve customer retention. By leveraging machine learning algorithms and advanced analytics techniques, businesses can identify patterns and correlations in customer behaviour, and take targeted actions to retain customers who are at risk of leaving. Overall, churn prediction is an essential tool for businesses looking to improve customer retention, reduce churn, and maximize revenue.

TABLE OF CONTENTS

1. Requirement Statement / Problem Statement

Large amount of data is generated in the Telecom Industry from large customer base. In this industry, acquiring new customer takes more time and money than keeping the existing customer. When a customer switches from one service provider to another service provider then it's called as customer churn.

The Goal is to use machine learning applications to predict the customer churn before it occurs. These models should be able to identify the patterns and trends that can identify customer churn factors and provide insights that can be used to mitigate the churning effect.

The problem statement involves selection of the appropriate machine learning techniques that can handle large amount of datasets to identify different types of customer behaviours. The challenge is to develop models that can effectively analyse and make sense of the vast amounts of customer data generated by telecom service providers.

So, the problem statement is to develop accurate and reliable models that can analyse large complex customer datasets to predict the customer churn risk and provide insights that to reduce customer churn rates and improve customer retention

2. Project Objectives

- To identify the key reasons why customers leave, such as poor customer service, pricing issues, lack of product features, or competitive options.
- To recognize customers exhibiting behaviours that indicate a high likelihood of churning, enabling targeted retention efforts.
- To analyse churn rates across different customer segments (demographics, usage patterns) to understand which groups are most prone to churn and tailor strategies accordingly.
- Using historical data to predict future churn and proactively engage with at-risk customers through personalized interventions.
- Utilizing insights from churn analysis to enhance customer service, product features, and overall customer journey to address pain points.
- Targeting retention efforts towards high-churn segments with personalized offers and communication strategies.
- To develop a predictive model that can predict whether the customer is likely to churn or

not based on its demographic and behavioural features.

3. Project Scope

Scope of customer churn project has been mentioned below:

- Project goals: To collect data, perform EDA and gain insights, apply machine learning models, suggest strategies for the management.
- Timeline:. In My project it is starting In January and will be finished by April/May end.
- Expected results: Extracting insights from the dataset which would help analyse the customer's behavioural patterns/requirement and provide services to reduce customer churn
- Deliverables: This includes the exact dashboards which represents the patterns and insights to support the suggested strategies.
- Predicting the likeliness of customer churn using predictive machine learning models based on their subscription package and demographic characteristics

4. Methodology

The proposed methodology for a churn prediction project involves different steps which includes data collection and pre-processing, exploratory data analysis (To gain insights), feature engineering, model selection and training and model evaluation.

The proposed methodology for churn prediction using SVM and Random Forest involves several steps.

The first step is data collection and pre-processing, where customer data related to their demographics, usage patterns, and past behaviour is collected and pre-processed. Pre-processing ensures that the data is of high quality and consistency, missing or garbage data is handled, and the data is transformed into a suitable format.

The second step feature engineering is performed to select or create derivative features that can add value while predicting customer churn. This involves using techniques such as correlation analysis or mutual information for feature selection and engineering new features based on domain expertise or other relevant factors.

The third step is model selection and training, where the selected machine learning algorithms (SVM and Random Forest) are evaluated and trained on pre-processed data using a suitable training technique such as cross-validation. Hyper parameter tuning is performed to optimize model performance.

In the fourth step, model evaluation is performed to assess the performance of the trained models on a holdout dataset. Various metrics such as accuracy, precision, recall, and F1 score are calculated to evaluate model performance. Techniques such as ROC curves or AUC are used to evaluate model performance.

Finally, the trained and evaluated models are deployed in the production environment. The models are integrated into the company's existing CRM system or other relevant tools and used to make churn predictions for new customers. Model performance is monitored, and the models are retrained as necessary to ensure continued accuracy and performance. Overall, this methodology enables companies to predict customer churn.

5. Plan vs Progress

We have provided information regarding the milestone of the project and the respective work completion.

Milestones	Percentage of work done	Progress bar
Data collection	100	🟩
Exploratory data analysis	100	🟩
Modelling	100	🟩
Creating dashboard	100	🟩
Suggesting marketing strategies	100	🟩

Table 5.1 – Planned Vs Progress table

<u>Data collection:</u>

We have collected data from KAGGLE data source for the project. We have performed Data cleaning with removing unwanted features, filling missing values and formatting feature values to get them compatible for model preparation.

Exploratory data analysis:

What has been done >> We performed exploratory data analysis in which we analysed relationship between features with each other and churn rate. We tried to find insights by analysing these relationships. We have found deeper insights that could help us in proposing marketing strategies.

Model preparation:

What has been done >> We have prepared Random Forest Classifier model and generated satisfactory results from the model. Models can be used to predict churn from the input features.

We also explored other models such as SVM (Support vector Machines), logistic regression to check whether we can obtain more accurate results from these models. We performed PCA analysis which helped to perform customer segmentation.

Creating Tableau dashboard:

What has been done >> We are working on getting basic understanding of the software so that we can apply its function to create dashboards for better understanding from visual presentation.

We Prepared Tableau dashboard that can show insights from the datasets visually. It helped in better understanding of customer dataset and behavioural activities.

Suggesting marketing strategies

What has been done >> currently we have found some insights from data exploration that has been mentioned in this report which could help in better understanding of customers.

We suggested marketing strategy that could provide better customer service and experience, ultimately leads to reduction in customer churn.

6. Detailed description of project

Literature review

Customer churn, the loss of customers over a period, is a significant concern for businesses. It's more expensive to acquire new customers than to retain existing ones, making churn prediction and management crucial for profitability and competitive advantage. The literature extensively explores factors influencing churn, the use of various machine learning techniques for churn prediction, and strategies for reducing churn rates.

S.No.	Base Papers	Summary
1	A Survey on Customer Churn Prediction using Machine Learning Techniques By *Saran Kumar* (2016)	literature on use of machine learning algorithms by researchers for churn prediction in banking sector and other sectors which highly depends on customer participation.
2	A comparative analysis of ML algorithms for customer churn prediction by Thanasis Vafeiadis (2015)	Literature studies on the use of machine learning methods and its application in the customer churn prediction in telecommunication industry
3	A Hybrid Machine Learning Model for Predicting Customer Churn in the Telecommunication Industry by Modupe Odusami (2021)	Study of hybrid model for the prediction of customer churn in the telecommunication industry that was developed using K Nearest Neighbour model, Logistic Regression model, Random Forest model and Decision Tree model. It has shown that our hybrid prediction model is superior to ordinary K nearest Neighbour, Logistic Regression model, Random Forest model and Decision Trees model.
4	Predicting customer churn in banking industry: A deep learning approach by Abisheak Jacob J & Sheetal Singh Parmar(*2023*)	Study of the critical application of predicting customer churn within the banking industry. This project uses Artificial Neural Networks (ANN), fine-tuned through sophisticated techniques such as K-fold cross-validation and Grid search cross-validation in Keras and Scikit-learn.

Working principles of an efficient framework for a churn prediction model involves following steps:

Dataset collection:

We have used a Kaggle dataset which contains various type of features about customers which is mentioned below:

Features	**Feature's meaning and significance**
Customers left in the last month	Churn
internet, device protection, streaming TV and movies, online backup, online security, tech support, multiple lines, phone	Services provided by company each customer has signed up for
Customer tenure, Payment method, Monthly charges, paperless billing, Total charges	Customer's behavioural information
if customer has partner and dependent, gender, senior citizen	Customer's Demographic information

Pre-processing of the dataset:

1.Data Cleaning

In this dataset, we have investigated for the missing values in the total charges features we dealt it with filling empty values with mean/median imputation methods.

All rows in these datasets contained unique values; each row contained a unique customer ID.

2.Data Transformation

For Random forest classifier method as a prediction model, there was no requirement of standardisation/normalization for feature scaling.

Since most of the feature in this dataset was categorical we had to convert these categorical features into numeric, so that the dataset would be compatible for the machine learning models. We have used one hot encoding and Label encoding methods to convert categorical data into numeric

3. Data Reduction

1. Dimensionality reduction: Reduce the number of features using techniques such as principal component analysis (PCA), t-SNE, or feature selection.

2. Data aggregation: Aggregate data by grouping it based on specific variables and applying aggregation functions such as sum, mean, or count.

4. Data Feature Engineering

In Feature engineering, we have converted tenure feature which was in month into years, we have used combination of 2 or more categorical feature using logic function to create new feature. We believe including these feature would help the model understand the patterns more clearly and provide more accurate results.

Exploratory data Analysis:

In exploratory data analysis, we have analysed churn rate according to the different features of the dataset

1. Churn rate of the customer is the highest within 1 year. Churn rate is decreasing with increase in tenure years.

Tenure in Years	Sum of Total_Counter	Average of MonthlyCharges	Churn Rate
0	11	41.42	0.00%
1	2175	56.17	47.68%
2	1024	61.36	28.71%
3	832	65.58	21.63%
4	762	66.32	19.03%
5	832	70.55	14.42%
6	1407	75.95	6.61%
Grand Total	7043	64.76	26.54%

Table 6.1 – Tenure in years Vs Churn rate

2. Below chart indicates the Tenure rate as per the customer's contract option.

We found that Most of the customer who opts for Month to Month contract option has the lowest tenure. In contrast to this Most of the Customers who opt for 2 year contract has the tenure of around 6 years.

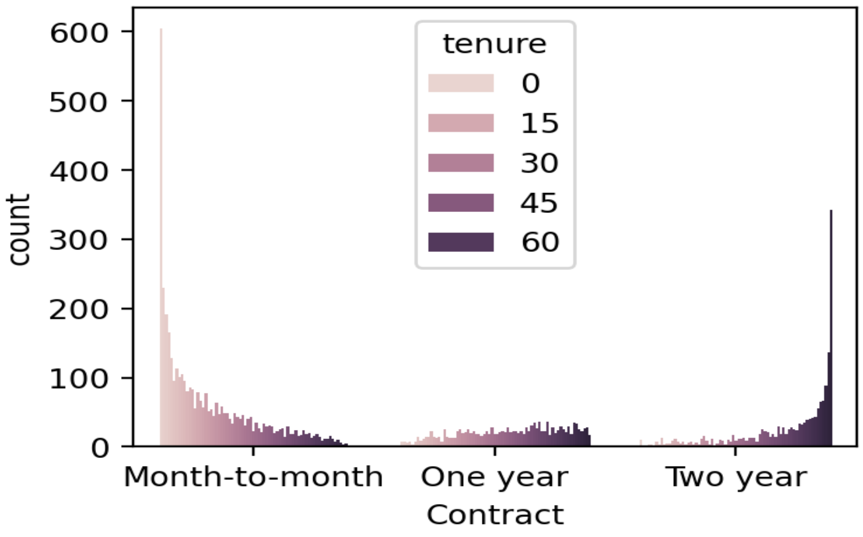
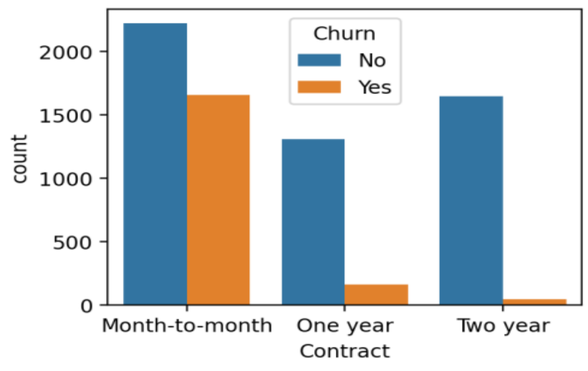


Fig 6.1 – Customer count Vs contract type (Hue by tenure)

<u>We have gathered some insights which are mentioned below:</u>

1. Churn rate of month to Month contract is the higher than among other contract options. While looking for insights we found that month to month contract customers are lacking Tech support. These customers are prone to dissatisfactory feedback and quite services due to lack tech support.



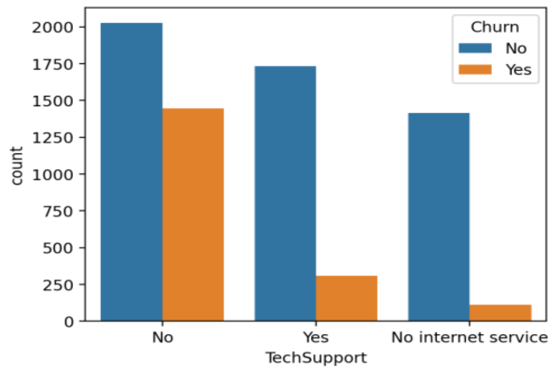


Fig 6.2 & Fig 6.3 – Contract type Vs Count & Tech support vs count

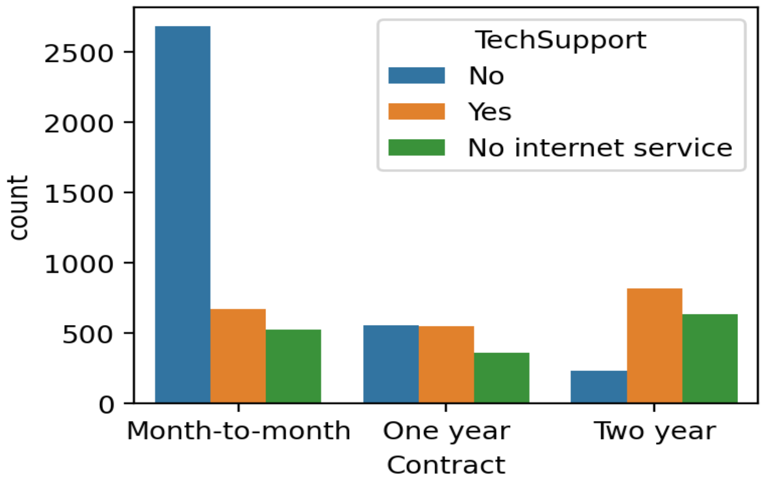
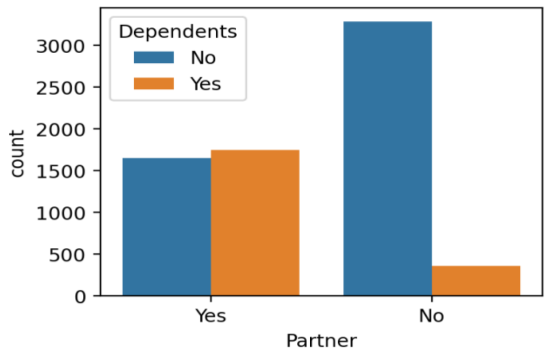


Fig 6.4 - Contract type Vs Count (Hue by Tech support)

2. Many Customers who have no partner has no dependents and they could become high value customer to the company, but currently these have opted for month to month contracts who actually has more churn rate than other contract options. Also, these customers are less attention in terms of tech support which could be one of the reasons for their higher churn rate



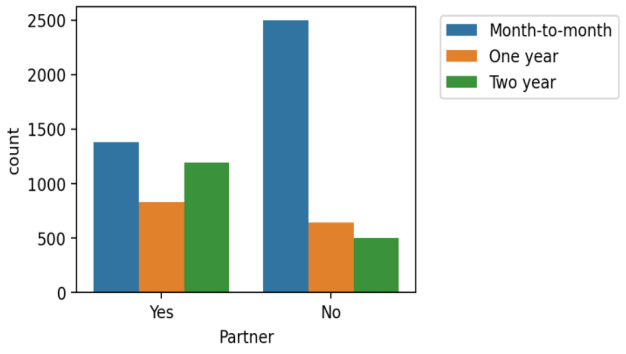


Fig 6.5 & Fig 6.6 – Partner Vs Count & Partner Vs Count

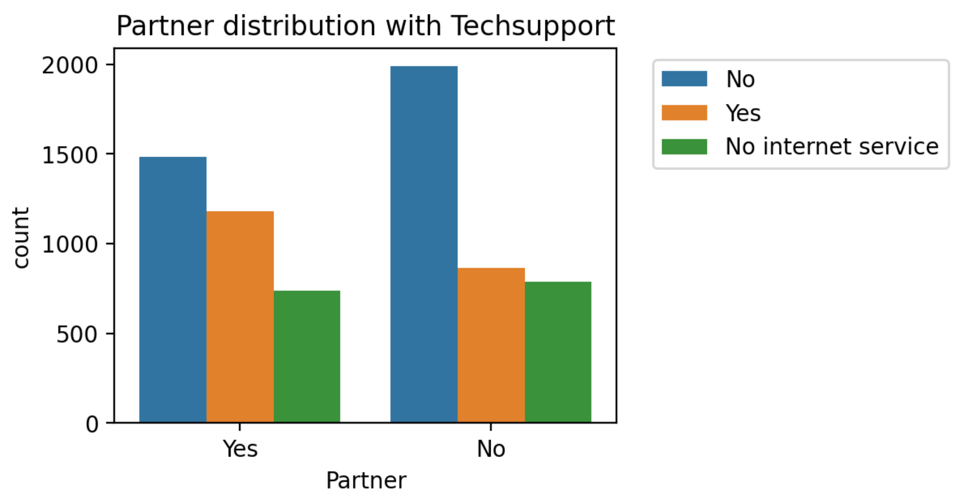


Fig 6.7 – Partner Distribution with tech support Vs count (Hue by Internet service)

3. Customers who pay their bill via Electronic check churn higher than other payment method. Non-fluency or technical issues in electronic check system that might have led to dissatisfactory experience for the customer encouraging the customer to churn

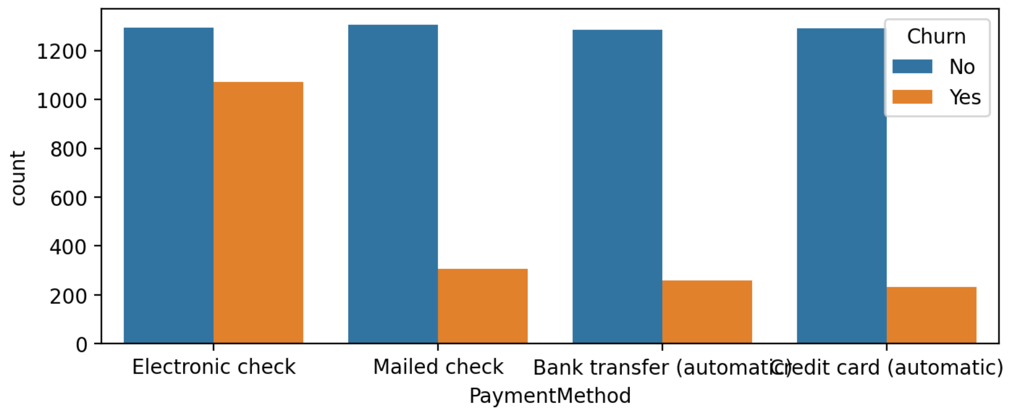


Fig 6.8 – Payment method Vs count

Model training Procedure:

(Decision Tree, Random Forest, XGBoost)

For model training we have used Jupyter Notebook platform. While preparing the model we have performed several steps. Below id the detailed information about the steps we had to go through while preparing the model, to make it more descriptive we have placed some parts of the code along with the output for better understanding.

1. First, we imported the required libraries such as Numpy, Pandas, matplotlib, seaborn, sklearn libraries, imblearn (For over sampling), xgboost, etc.

We loaded the customer churn dataset from Kaggle which represents the data of customers who are/were using the subscription plans provided by the company. We did some basic inspection of the dataset such as no. of rows and columns it contains, the features and their datatypes. We inspected if features contain null values.

In this dataset customerID feature was not useful since it contains random string values and does not provide any additional information about the customer. The TotalCharges feature was in object datatype, after converting it into float type we found some missing values in this feature. We have replaced the missing values with 0 because the missing values were of 0 tenure customers. We used unique function from pandas to get the unique values of categorical features.

Then, we checked the distribution of target column. We found that the target column distribution was imbalanced .Class_No values (5174 count) were higher than Class_Yes (1869 count). Due to this we would need to perform Undersampling/Oversampling to make Output column balance.

2. After this we performed some basic EDA to get some more information/Insights from the data we used df.describe() to get statistical information of the numerical features. Then we looked into the distribution of the numerical features by plotting the histogram of these features i.e. tenure, MonthlyCharges, TotalCharges. Then we plotted correlation matrix between the numerical features. From this we found that tenure and TotalCharges

are highly correlated. We can exclude one of the features to neglect Multicollinearity issue. We had already performed EDA in detailed finding some patterns between the categorical features and drawn out some insights from it. So we moved for the data pre-processing.

3. In the data pre-processing we had to make sure that the feature values are compatible for the model training. Since most of the features are categorical in this dataset we converted the categorical values into numeric using Label encoding method. We have used Label encoding instead of One-Hot-Encoding to avoid high dimensionality issue.

The next step is to handle the imbalance in the output column. For this we used SMOTE method (This is simply an oversampling method which duplicates the minority class data to balance the data). So we split the data into training and testing dataset first then we applied SMOTE to make it balance dataset. For now we are using decision tree based methods for modelling so there is no need to standardize the data. After applying SMOTE we verified that the both classes are now balanced. Now we stepped into the model training part.

4. We trained with the default hyperparameters of 3 models that are: Decision Tree Classifier, Random Forest Classifier, and XGBoost.

We kept the same random state for all 3 models so that we can compare the performance of these 3 models. We used cross validation technique to get a more reliable and unbiased estimate of a model's performance on unseen data and avoid the problems of overfitting and underfitting.

After running the model we got the cross validation accuracy score for all 5 iterations for each model. After this we calculated an average cross validation accuracy score.
The cross validation accuracy for each model is provided below:

Decision Tree Classifier:	0.78
Random Forest Classifier:	0.84
XGBoost :	0.83

Here the accuracy score for Random Forest is higher than the Decision Tree
and XGBoost accuracy score.
So to further fine tune the Random Forest cross-validation model we have use hyper

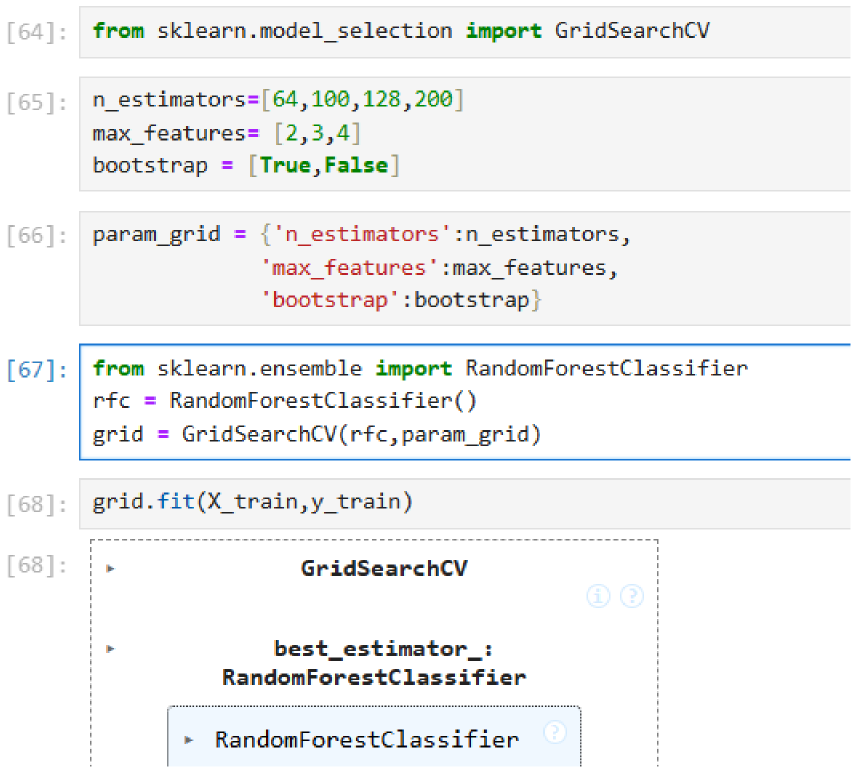
parameter tuning method using GridSearchCV to find the best combination which could produce highest accuracy.

For hyperparameter tuning of RandomForestClassifier, we have selected 3 parameters:

n_estimators: No of decision trees to produce in modelling

max_features : No. of features to use for each decision tree

bootstrap : Whether to use bootstrapping or not



```
[64]: from sklearn.model_selection import GridSearchCV
```

```
[65]: n_estimators=[64,100,128,200]
      max_features= [2,3,4]
      bootstrap = [True,False]
```

```
[66]: param_grid = {'n_estimators':n_estimators,
                    'max_features':max_features,
                    'bootstrap':bootstrap}
```

```
[67]: from sklearn.ensemble import RandomForestClassifier
      rfc = RandomForestClassifier()
      grid = GridSearchCV(rfc,param_grid)
```

```
[68]: grid.fit(X_train,y_train)
```

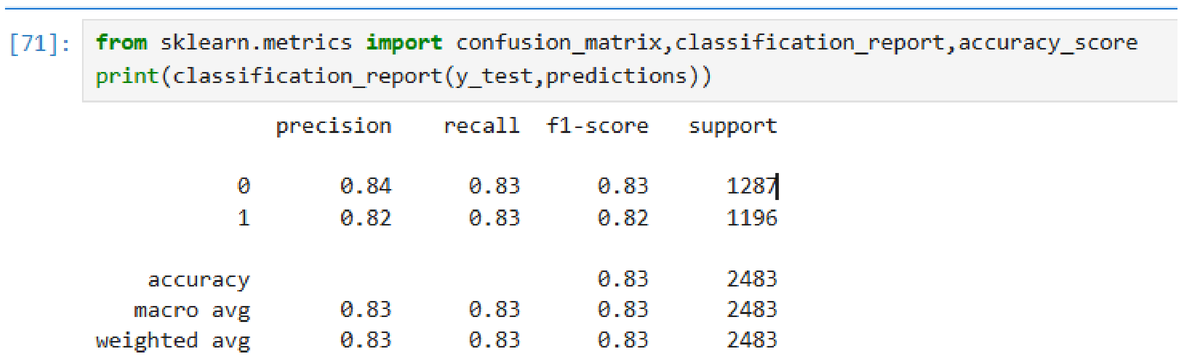
```
[68]: ▸ GridSearchCV
      ▸ best_estimator_:
        RandomForestClassifier
        ▸ RandomForestClassifier
```

Bootstrapping is a resampling technique where multiple datasets are created by randomly sampling with replacement from the original dataset, used to estimate model performance, variability, and stability.

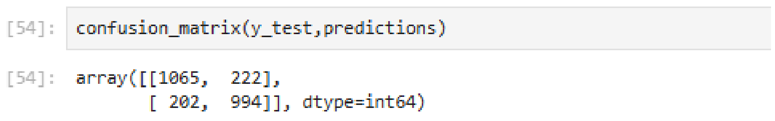
After running the model we got best parameters as

{'bootstrap': True, 'max_features': 4, 'n_estimators': 100}

Then we printed classification report and confusion matrix



```
[71]: from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
      print(classification_report(y_test,predictions))

                    precision    recall  f1-score   support

                 0       0.84      0.83      0.83      1287
                 1       0.82      0.83      0.82      1196

          accuracy                           0.83      2483
         macro avg       0.83      0.83      0.83      2483
      weighted avg       0.83      0.83      0.83      2483
```



```
[54]: confusion_matrix(y_test,predictions)

[54]: array([[1065,  222],
             [ 202,  994]], dtype=int64)
```

<u>Support Vector Machines:</u>

<u>Background:</u>

Support Vector Machines is one of the most recently developed machine learning algorithms Support Vector Machines involves hyper planes and margins. In an N-dimensional space, a hyper plane is a flat affine subspace of hyper plane dimension N - 1. For example, in one dimension, a hyper plane is really just a single point. In two dimensions, the hyper plane is a line. And in three dimensions, the hyper plane is a flat plane. The core idea behind support vector machines the use of hyper planes to create a separation between classes, then new points will fall on one side of the separating hyper plane, which we can then use to assign a class. We need to decide which of these points or hyper planes is going to be the best separator between the classes. We need some sort of quantitative methodology to choose "best separator". The best separator hyper plane is going to have the maximum margin from first instance of each class, and this is known as the **Maximum Margin Classifier.**

We need to introduce is a bias-variance trade-off which depends on where we actually place this separator because wherever we place this separator, it's essentially creating ranges for what is going to be classified as Churn versus what will be classified as Not-Churn. The distance between the threshold and the observations is a soft margin. A soft margin allows for misclassification inside of the margins, allowing us to introduce more bias to our model in order to hopefully greatly reduce variance. We need to figure out what level of misclassifications should be allowed. We can use cross validation to determine the optimal size of the margins, essentially testing out different levels of misclassifications allowed to be performed inside of the margins to figure out the best overall model

Firstly, we have imported all the required libraries for modelling into the jupyter notebook ex. Numpy, Pandas, Matplotlb, GridSearchCV, StandardScaler, etc

Machine learning steps:

We imported the required libraries for modelling. Then we have imported the dataset that we are going to use for machine learning modelling. We have dropped the "customerID" column since it not providing any value in the context of developing a model. Then we checked the different feature values of the dataset, we replaced the missing values of "TotalCharges" column with 0 since these missing values were coming from the newly joined customers within 1st month.

Then we checked the datatypes of the features. For converting string values into numerical values of the object features we selected group of features which belonged to object features and converted their values into numeric using “pd.get_dummies” function.

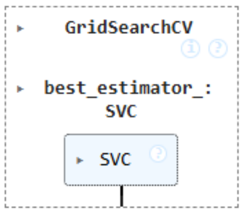
We created X dataset of input features and Y dataset of output feature. From sklearn.model we imported train_test_split. Using this we created “X_train, X_test, y_train, y_test” dataset for training and testing the models.

For Support vector machines, we scaled the values of the features using “StandardScaler” and created scaled_X_train and scaled_X_test dataset. Then we imported SVC model from sklearn.model_selection.

Then we created parameter Grid using GridSearchCV to check different combinations of hyperparameter to get the combination providing the best results.

```
[20]: from sklearn.svm import SVC

[21]: from sklearn.model_selection import GridSearchCV

[22]: svc = SVC(class_weight='balanced')

[23]: param_grid = {'C':[0.001,0.01,0.1,0.5,1],'gamma':['scale','auto']}
      grid = GridSearchCV(svc,param_grid)

[24]: grid.fit(scaled_X_train,y_train)
```

[24]:



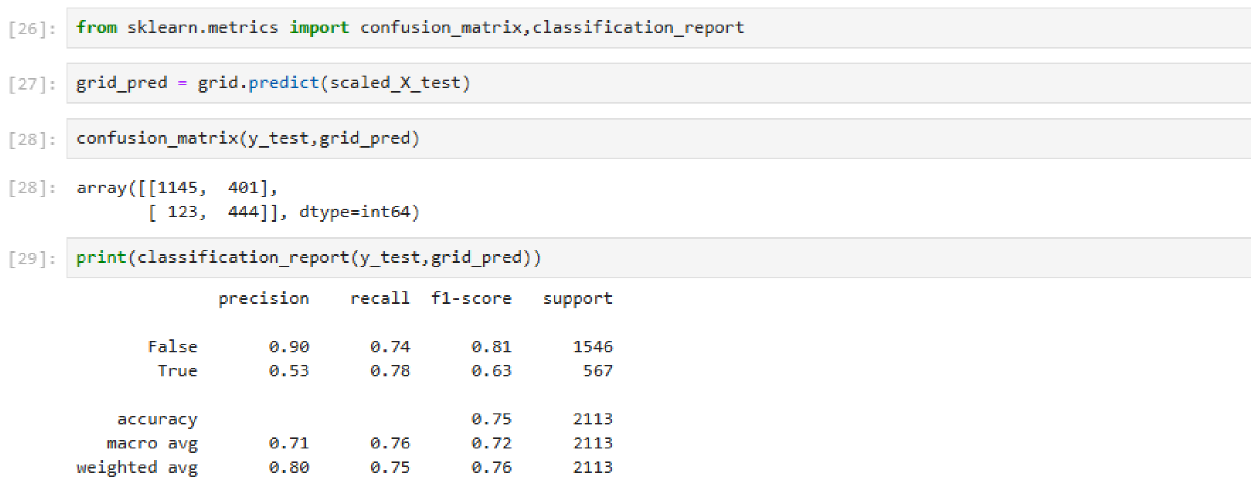
```
[25]: grid.best_params_

[25]: {'C': 0.5, 'gamma': 'scale'}
```

However, we balanced the model which uses the values of Y to automatically adjust weights. This is an important part in which we assign weightage to the class inversely proportional to class frequencies. We already know that there are very few instances of Churn_yes. So what we would like to be done is that the features for Churn_yes are given a little more weight during training. Essentially the weight is going to be determined inversely proportional to the frequency of that class. So balance is essentially going to try to auto-balance the classes for you by adding a weight to the instances of the classes that are imbalanced.

In the parameter grid we provided different ranges of C values from low to high and the gamma values we provided as scaled or auto. Then we fitted the parameter grid with scaled_X_train and y_train using "grid.fit(scaled_X_train,y_train)". Then we found best parameters as C=0.5 and gamma = scale.

From sklearn.metrics we imported confusion_matrix,classification_report to generate reports.

Then, we used best model to predict the vales for scaled_X_test dataset and compared these with the y_test vales to generate confusion_matrix and classification_report

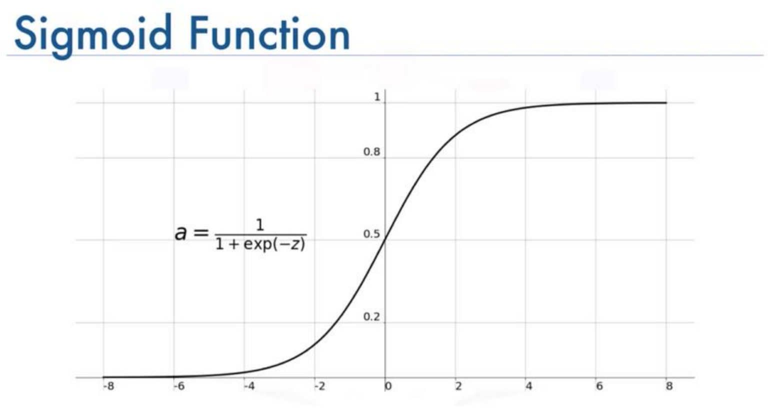


```
[26]: from sklearn.metrics import confusion_matrix,classification_report

[27]: grid_pred = grid.predict(scaled_X_test)

[28]: confusion_matrix(y_test,grid_pred)

[28]: array([[1145,  401],
             [ 123,  444]], dtype=int64)

[29]: print(classification_report(y_test,grid_pred))

                  precision    recall  f1-score   support

           False       0.90      0.74      0.81      1546
            True       0.53      0.78      0.63       567

        accuracy                           0.75      2113
       macro avg       0.71      0.76      0.72      2113
    weighted avg       0.80      0.75      0.76      2113
```

Logistic Regression:

Logistic regression is a classification algorithm designed to predict categorical target labels. In this model, we transform the linear regression algorithm to logistic regression using the sigmoid function, otherwise known as the logistic function.

Logistic regression in general works by transforming a linear regression into a classification model through the use of the logistic function shown here.



In this model, we used Ridged and Lasso regularisation techniquesfor the regularisation of the linear models to avoid over fitting and improve predictive performance.

- Ridge Regression (L2 regularization), adds the squared magnitude of the coefficients as a penalty.
- On the other hand, Lasso Regression (L1 regularization) introduces a penalty based on the absolute value of the coefficients.

<u>Machine learning steps:</u>

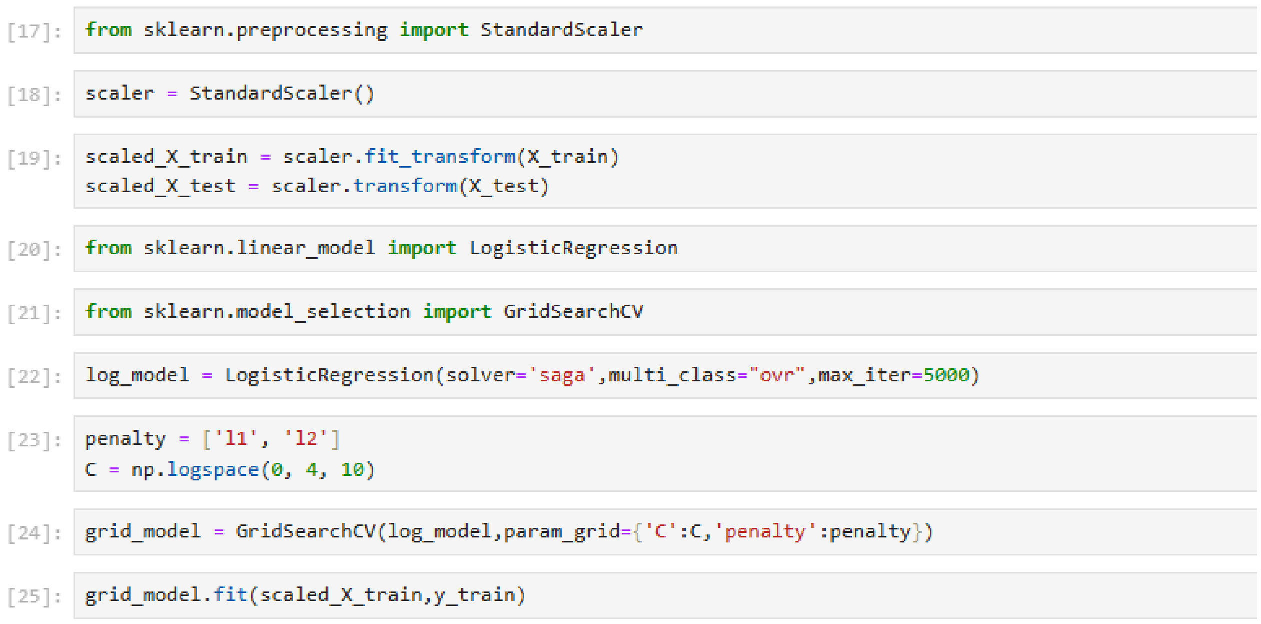
In logistic regression modelling, first we imported the libraries, then we read in the dataset using pd.read_csv function. Then we dropped the non-required features from the dataset. Then, we dealt with the missing values in the dataset.

We converted categorical values into numeric with one hot encoding method by using pd.get_dummies function. Then we imported train_test_split function to split the data into training dataset and testing dataset.

To prevent any single feature from dominating the learning process due to its larger magnitude we did scaling of the data using StandardScaler function

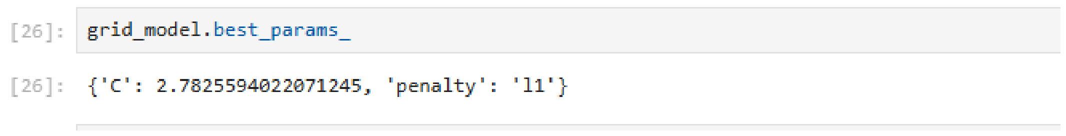
Then we transformed scaled_x_test dataset and *fit + transformed* the scaled_x_train dataset to prepare them for modelling computation. Then we imported LogisticRegression model, we also used GridSearchCV to compare different hyper parameters combination results and find the best parameter results.

For GridSearchCV, we used below grid of hyperparameter, penalty = ['l1', 'l2'] ; C = np.logspace(0, 4, 10)



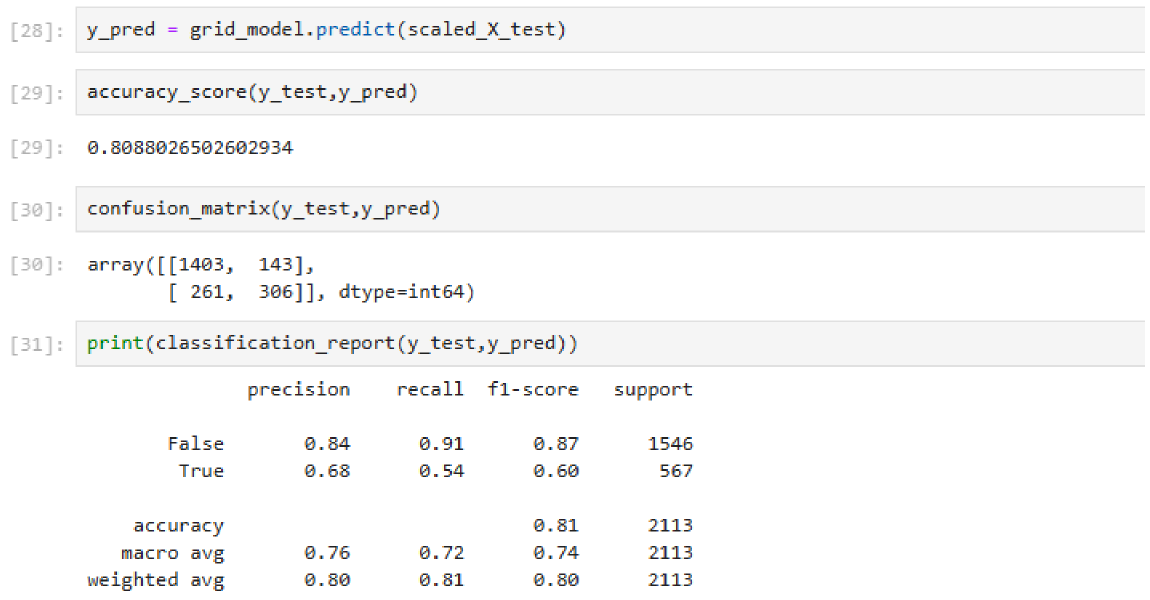
```
[17]: from sklearn.preprocessing import StandardScaler

[18]: scaler = StandardScaler()

[19]: scaled_X_train = scaler.fit_transform(X_train)
      scaled_X_test = scaler.transform(X_test)

[20]: from sklearn.linear_model import LogisticRegression

[21]: from sklearn.model_selection import GridSearchCV

[22]: log_model = LogisticRegression(solver='saga',multi_class="ovr",max_iter=5000)

[23]: penalty = ['l1', 'l2']
      C = np.logspace(0, 4, 10)

[24]: grid_model = GridSearchCV(log_model,param_grid={'C':C,'penalty':penalty})

[25]: grid_model.fit(scaled_X_train,y_train)
```

Then we fitted the model to (scaled_X_train,y_train) dataset for the computation of model prepara tion. Then we found the best parameter as (C': 2.7825594022071245, 'penalty': 'l1')



```
[26]: grid_model.best_params_

[26]: {'C': 2.7825594022071245, 'penalty': 'l1'}
```

Then, we used best model to predict the vales for scaled_X_test dataset and compared these with the y_test vales to generate confusion_matrix and classification_report



```
[28]: y_pred = grid_model.predict(scaled_X_test)

[29]: accuracy_score(y_test,y_pred)

[29]: 0.8088026502602934

[30]: confusion_matrix(y_test,y_pred)

[30]: array([[1403,  143],
             [ 261,  306]], dtype=int64)

[31]: print(classification_report(y_test,y_pred))

                    precision    recall  f1-score   support

           False       0.84      0.91      0.87      1546
            True       0.68      0.54      0.60       567

        accuracy                           0.81      2113
       macro avg       0.76      0.72      0.74      2113
    weighted avg       0.80      0.81      0.80      2113
```

<u>Model deployment:</u>

Background:

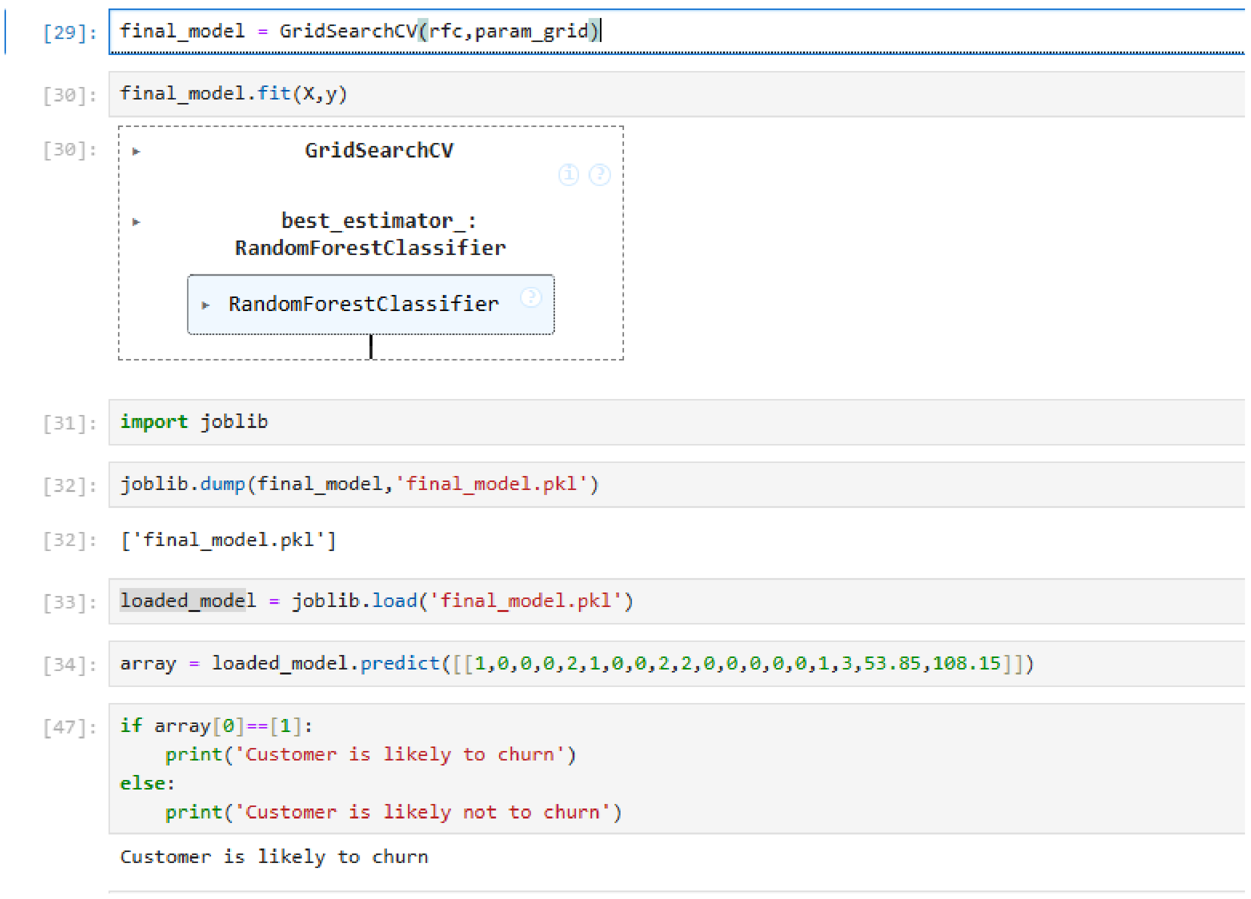
Model deployment in machine learning is the process in which we train the model and make the model available for use in a real-world application or system. This involves integrating the model into an environment where it can receive input data, process it, and generate predictions or classifications.

In this we learnt about the lifecycle of creating, training, saving and loading a machine learning model of Scikit-Learn. Then we would set up a saved model to be used to generate predictions or classifications.

As we have tested and compared the performance of the different models, we have selected the model with the best model in terms of Precision and Recall which is *RandomForestClassifier.*

Steps for the model deployment:

1) We have saved model as *final_model* by providing the parameters.
2) We fitted the model with the X and y parameters.
3) In this, we have imported *joblib* which will be used for dumping and saving the model.
4) Then we dumped and saved the model as *loaded_model*
5) Then we provided inputs into the model. After this model provided the predictions in terms of 0, 1 arrays which we converted into interpretable language For Ex. Customer is likely to churn or Customer is likely not to churn



```
[29]: final_model = GridSearchCV(rfc,param_grid)

[30]: final_model.fit(X,y)

[30]: ▸ GridSearchCV
      ▸ best_estimator_:
        RandomForestClassifier
        ▸ RandomForestClassifier

[31]: import joblib

[32]: joblib.dump(final_model,'final_model.pkl')

[32]: ['final_model.pkl']

[33]: loaded_model = joblib.load('final_model.pkl')

[34]: array = loaded_model.predict([[1,0,0,0,2,1,0,0,2,2,0,0,0,0,0,1,3,53.85,108.15]])

[47]: if array[0]==[1]:
          print('Customer is likely to churn')
      else:
          print('Customer is likely not to churn')

      Customer is likely to churn
```

PCA analysis of customer data:

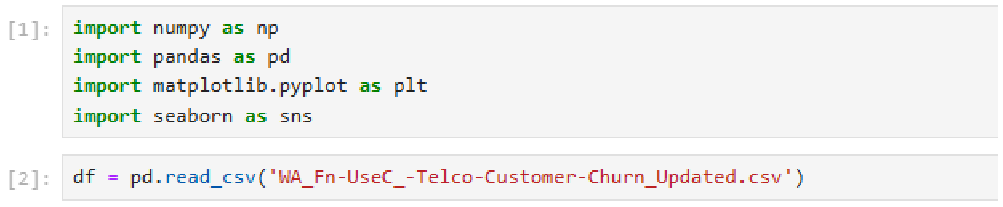
Too many features in the dataset can cause problems like over fitting (provides good results on training data but poor results on new data) with slower computation and lower accuracy. It is known as the curse of dimensionality, where data required for reliable results increases exponentially with the no. of features.

The more number of feature combinations makes sampling harder in the high-dimensional data and this makes tasks like clustering and classification more complex and slow

PCA works on the principle of transforming high-dimensional data into a lower-dimensional while maximizing the variance of the data in the new space. This helps in preserving the most important patterns and relationships in the data.

Steps for PCA analysis:

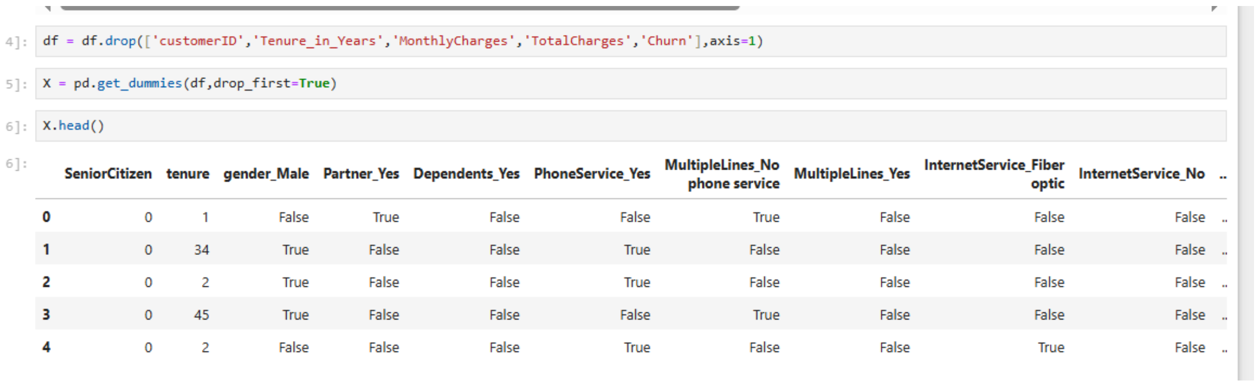
1. Importing the necessary libraries and the dataset.



```
[1]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns

[2]: df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn_Updated.csv')
```

2. Exclusion of some unnecessary features for PCA analysis.

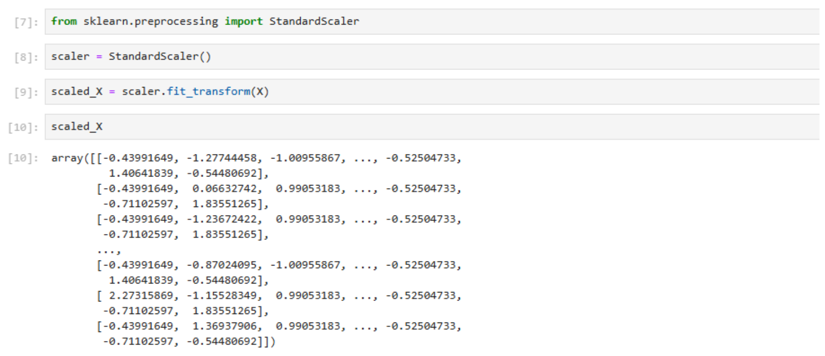
3. Conversion of categorical values into numerical for the features.



```
4]: df = df.drop(['customerID','Tenure_in_Years','MonthlyCharges','TotalCharges','Churn'],axis=1)

5]: X = pd.get_dummies(df,drop_first=True)

6]: X.head()

6]:
```

	SeniorCitizen	tenure	gender_Male	Partner_Yes	Dependents_Yes	PhoneService_Yes	MultipleLines_No phone service	MultipleLines_Yes	InternetService_Fiber optic	InternetService_No	..
0	0	1	False	True	False	False	True	False	False	False	..
1	0	34	True	False	False	True	False	False	False	False	..
2	0	2	True	False	False	True	False	False	False	False	..
3	0	45	True	False	False	False	True	False	False	False	..
4	0	2	False	False	False	True	False	False	True	False	..

4. Scaling of the data and transforming it to make it compatible for PCA analysis.

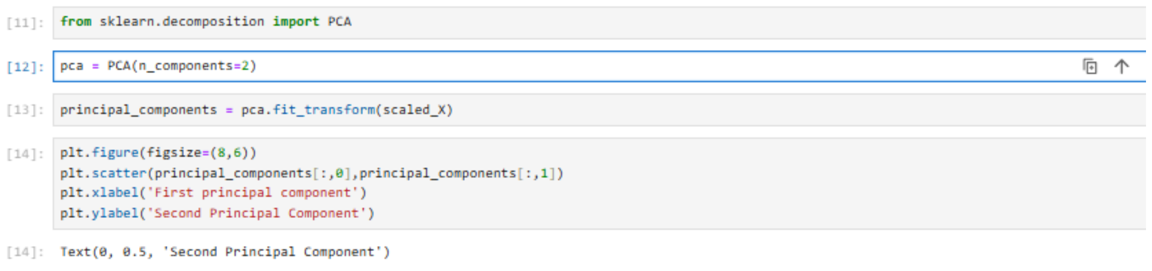


```
[7]:  from sklearn.preprocessing import StandardScaler

[8]:  scaler = StandardScaler()

[9]:  scaled_X = scaler.fit_transform(X)

[10]: scaled_X

[10]: array([[-0.43991649, -1.27744458, -1.00955867, ..., -0.52504733,
                1.40641839, -0.54480692],
              [-0.43991649,  0.06632742,  0.99053183, ..., -0.52504733,
               -0.71102597,  1.83551265],
              [-0.43991649, -1.23672422,  0.99053183, ..., -0.52504733,
               -0.71102597,  1.83551265],
              ...,
              [-0.43991649, -0.87024095, -1.00955867, ..., -0.52504733,
                1.40641839, -0.54480692],
              [ 2.27315869, -1.15528349,  0.99053183, ..., -0.52504733,
               -0.71102597,  1.83551265],
              [-0.43991649,  1.36937906,  0.99053183, ..., -0.52504733,
               -0.71102597, -0.54480692]])
```

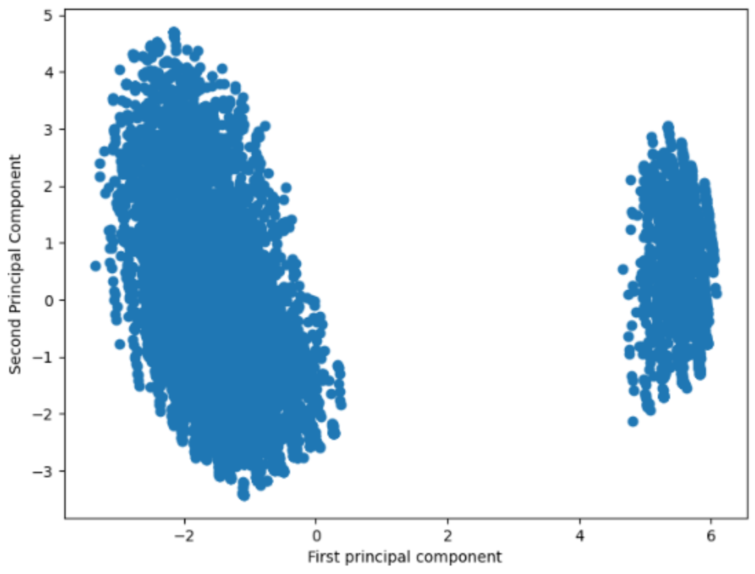
5. Importing PCA libraries , inputs given such as no. of PCA components, fitting and transforming PCA components into scales X dataset

6. Plotting of PCA components.

Here PCA components cam be recognized as completely separate from each other.



```
[11]: from sklearn.decomposition import PCA

[12]: pca = PCA(n_components=2)

[13]: principal_components = pca.fit_transform(scaled_X)

[14]: plt.figure(figsize=(8,6))
      plt.scatter(principal_components[:,0],principal_components[:,1])
      plt.xlabel('First principal component')
      plt.ylabel('Second Principal Component')

[14]: Text(0, 0.5, 'Second Principal Component')
```

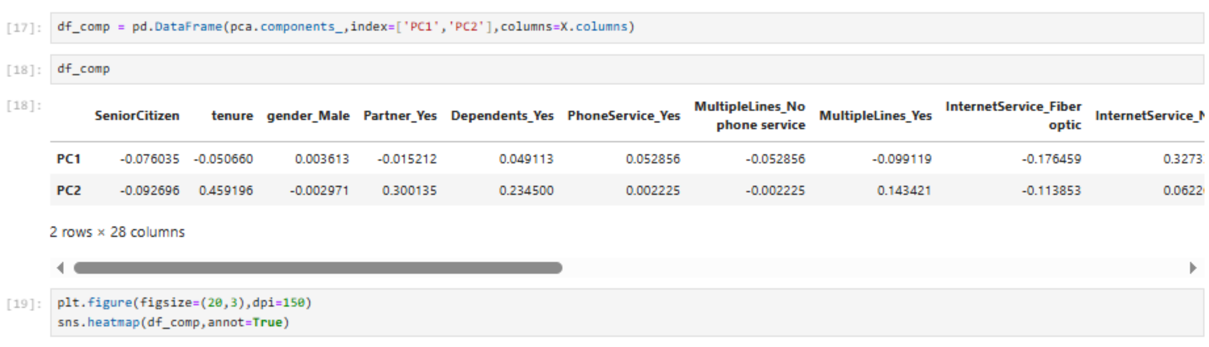


7. Computing PCA arrays and plotting heat map.

From the PCA heatmap we could classify 2 categories of the customers.

Category 1: customers with having partner/dependents , high tenure having 2 year contract subscribed to tech support , device protection, online security

Category 2: Customers have not subscribed to internet service, online security, online backup, tech support, streaming TV or movies



```
[17]: df_comp = pd.DataFrame(pca.components_,index=['PC1','PC2'],columns=X.columns)
```

```
[18]: df_comp
```

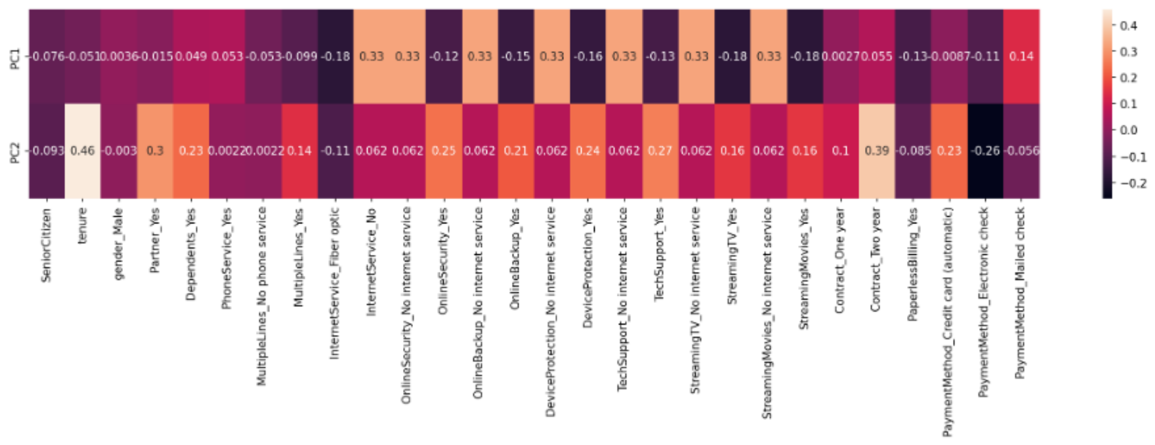
[18]:

	SeniorCitizen	tenure	gender_Male	Partner_Yes	Dependents_Yes	PhoneService_Yes	MultipleLines_No phone service	MultipleLines_Yes	InternetService_Fiber optic	InternetService_N
PC1	-0.076035	-0.050660	0.003613	-0.015212	0.049113	0.052856	-0.052856	-0.099119	-0.176459	0.3273
PC2	-0.092696	0.459196	-0.002971	0.300135	0.234500	0.002225	-0.002225	0.143421	-0.113853	0.0622

2 rows × 28 columns

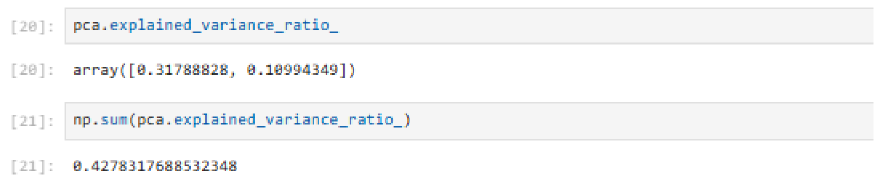
```
[19]: plt.figure(figsize=(20,3),dpi=150)
      sns.heatmap(df_comp,annot=True)
```

```
[19]: <Axes: >
```



8. Calculating total explained variance by PCA components.

Here we can observe that only 2 PCA components have explained 42.8 % of variance.



```
[20]: pca.explained_variance_ratio_

[20]: array([0.31788828, 0.10994349])

[21]: np.sum(pca.explained_variance_ratio_)

[21]: 0.42783177688532348
```

9. Plotting the no. of PCA components vs variance explained graph.

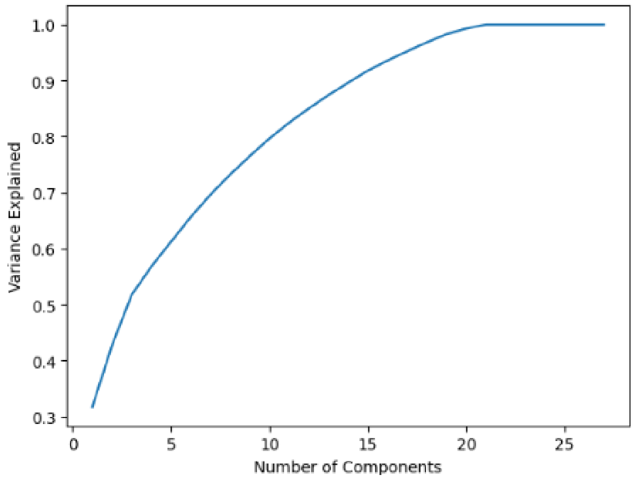
Here we have plotted the graph of no. of PCA components vs variance explained.

We can observe that 10-12 PCA components can explain 80-90 % of variance.

Note: that total no of components are 28, So with 10-12 no. of PCA components we can explain almost 80-90 % of variance of the dataset.

```
[25]: explained_variance = []

      for n in range(1,28):
          pca = PCA(n_components=n)
          pca.fit(scaled_X)

          explained_variance.append(np.sum(pca.explained_variance_ratio_))
```

```
[26]: plt.plot(range(1,28),explained_variance)
      plt.xlabel("Number of Components")
      plt.ylabel("Variance Explained");
```



10 .3 D plotting of 3 PCA components.

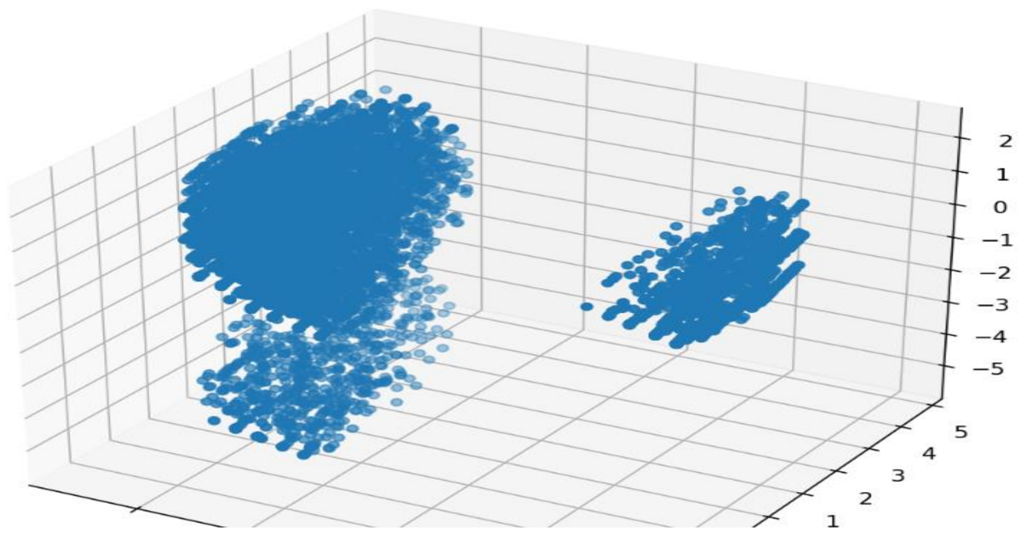
Just like 2 D plot, we have plotted 3-D plot of 3 principle components

```
[27]: from sklearn.decomposition import PCA
      pca_model = PCA(n_components=3)
```

```
[28]: pca_model = pca_model.fit_transform(scaled_X)
```

```
[29]: from mpl_toolkits import mplot3d
```

```
[30]: plt.figure(figsize=(8,8),dpi=150)
      ax = plt.axes(projection='3d')
      ax.scatter3D(pca_model[:,0],pca_model[:,1],pca_model[:,2]);
```



Creating a dashboard:

We have created a Dashboard using Tableau to provide a visual and concise overview of key data and metrics which can facilitate quick decision-making and informed actions. Dashboards present a comprehensive view of business performance which allows users to see the big picture and identify areas needing attention. Dashboards can be customized to reflect individual user needs and preferences, ensuring relevant information are readily available.

In this dashboard, we have created 3 graphs

1. No. of customers vs Tenure (Churn as Hue)

2. No. of customers vs Tenure (Churn as monthly charges)

3. No. of customers vs Monthly charges (Bin)

Also, we have created Partner, Dependents, Churn and Senior Citizen Venn diagram. Also, we can filter the data using action filters of paperless billing, Contract, Device protection and multiple lines.

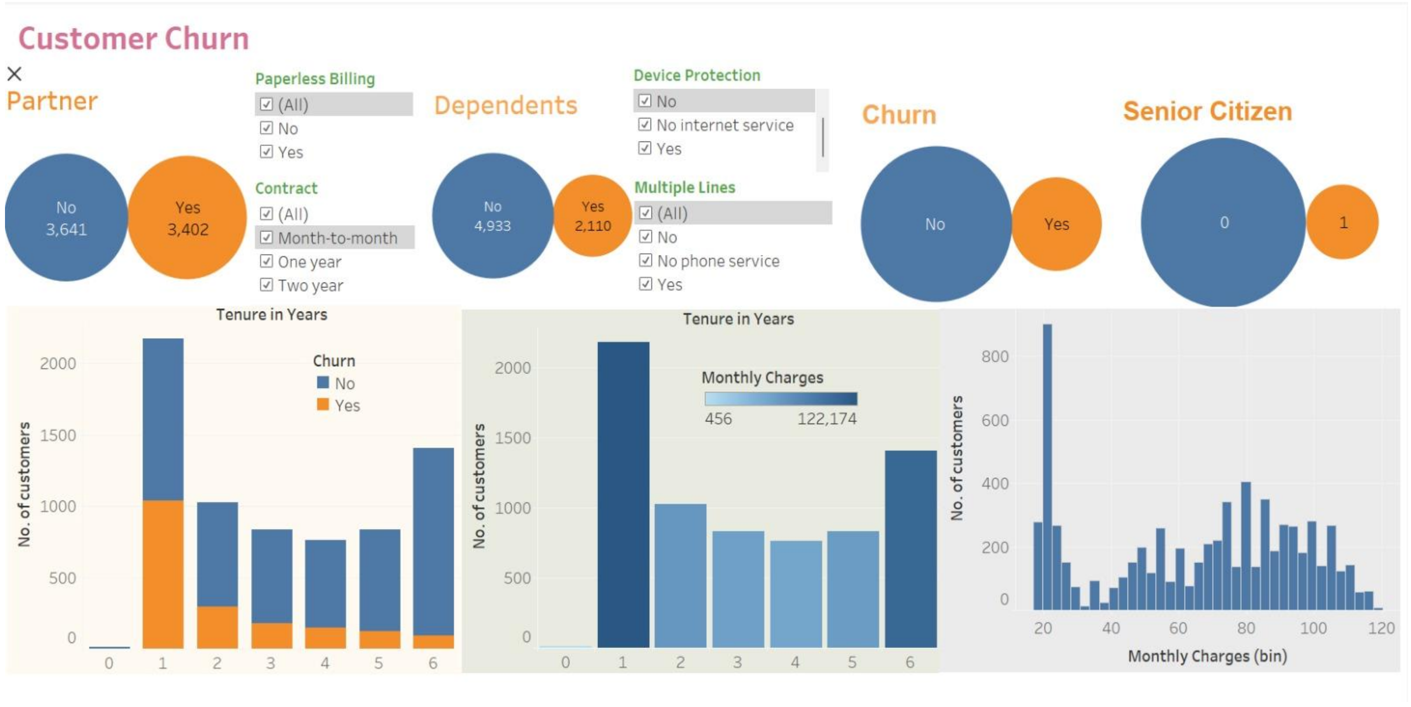


Fig 6.10 – Customer Churn dashboard

7. Resource Requirements and their availability

Software Requirements

Software - Jupyter Notebook

Language - Python(libraries:pandas,matplotlib,sklearn, imblearn)

Python is a popular language for machine learning known for its simplicity and readability. It is also equipped with extensive libraries such as scikit-learn, Pandas, Matplotlib, Tensor Flow, and PyTorch, which streamline complex ML tasks.

Libraries:

Pandas : Pandas libraries facilitates importing and exporting datasets from various file formats, such as CSV, SQL, and spread sheets. Pandas libraries are known for its cleaning, reshaping and analysing tabular and statistical datasets capabilities

Scikit-learn, is a popular machine learning library for Python. Scikit-learn is equipped with various libraries that are used for data mining and data analysis. It comes with inbuilt machine learning Classification, Regression models and dimensionality reduction via consistent and easy to use interface.

Matplotlib is equipped with data visualization libraries for the Python programming language that provides tools for creating different types of graphs, charts and customisable plots. It is a popular library and used for data science and scientific computing because of its versatility and flexibility.

Imbalanced-learn (imblearn) is a Python library is used to balance datasets that highly skewed to one category of a feature. We can use oversampling/Under sampling to prepare balance dataset.

Operating System - Any Operation System (using windows 11)

3.6.6 Hardware Requirements

Processor - intel i3 (using intel i5)

Hard disk - 10GB (using 256 GB) ; RAM - 4GB (using 8GB)

8. Risks and Mitigations

Predicting customer churn in the telecom industry using machine learning (ML) techniques can be a challenging task due to several reasons:

1. Imbalanced datasets: Customer churn datasets in the telecom industry are often imbalanced, with a much smaller number of churn instances compared to non-churn instances. This can lead to biased models that perform poorly on predicting the minority class. This misbalancing problem can be mitigated by using oversampling and under sampling techniques

Imbalanced-learn (imblearn) is a Python library for dealing with imbalanced datasets in machine learning. An imbalanced dataset is one in which the distribution of the classes is highly skewed, with one class having significantly fewer instances than the others.

Imbalanced datasets can lead to biased or inaccurate machine learning models, as the classifier may be biased towards the majority class. Imblearn provides a range of techniques for addressing this problem, such as oversampling, undersampling, and generating synthetic data.

2. Large and complex datasets: Telecom datasets can be large and complex, with a vast amount of customer data such as demographics, usage patterns, and call records. pre-processing and cleaning this data can be a time-consuming and challenging task.

3. High dimensionality: Telecom datasets can have a high number of features, making it difficult to identify the most important factors that contribute to customer churn. Feature selection and feature extraction techniques can be used to avoid high dimensionality issue. Dimensionality reduction techniques such as principal component analysis (PCA) can also help address this issue.

4. Variability in customer behaviour: Customer behaviour in the telecom industry can be highly variable and subject to change. This can make it challenging to identify stable patterns and trends that accurately predict churn. Visualisation techniques can be used to identify patterns between different features of the customers.

9. Issues and Resolutions

1. Data Quality:

Poor quality data (noise, missing values, and inconsistencies) can lead to inaccurate predictions and unreliable models.

To maintain data quality we had to perform data pre-processing which involves handling missing data using mean/median imputation, forward/backward fill, or interpolation methods. We can use missing value as feature by finding the reason of missing values.

2. Data Representation:

Choosing the right features and representing data effectively is crucial for model accuracy.

Performing Exploratory data analysis, finding relation between an individual feature with output features, finding P1 score can help to find important features for model training. Pandas come with libraries that shows feature importance after training the model that can help to find the importance.

3. Data Bias:

Biases in the training data can result in models that unfairly discriminate against certain groups.

Cross-validation: In cross validation, training and testing datasets are splitted multiple times each time in different order and model performance is evaluated for each iteration. It reduces the chances of over fitting or under fitting.

Feature selection: Selecting only important features respective to model preparation can help in reducing the variance error.

Ensemble methods: Ensemble methods use multiple models to improve generalization performance. Bagging and Boosting are the type of ensemble techniques that are used for reducing variance and improve generalisation performance.

10. Conclusions and Recommendations

The churn prediction is a critical task for businesses looking to improve customer retention and reduce the impact of churn on their bottom line. Churn prediction enables businesses to proactively identify and retain customers at risk of leaving, leading to improved customer retention, reduced acquisition costs, and sustained business growth. Machine learning algorithms such as SVM and Random Forest can be used to build predictive models that analyse customer behaviour and identify those who are at risk of churning.

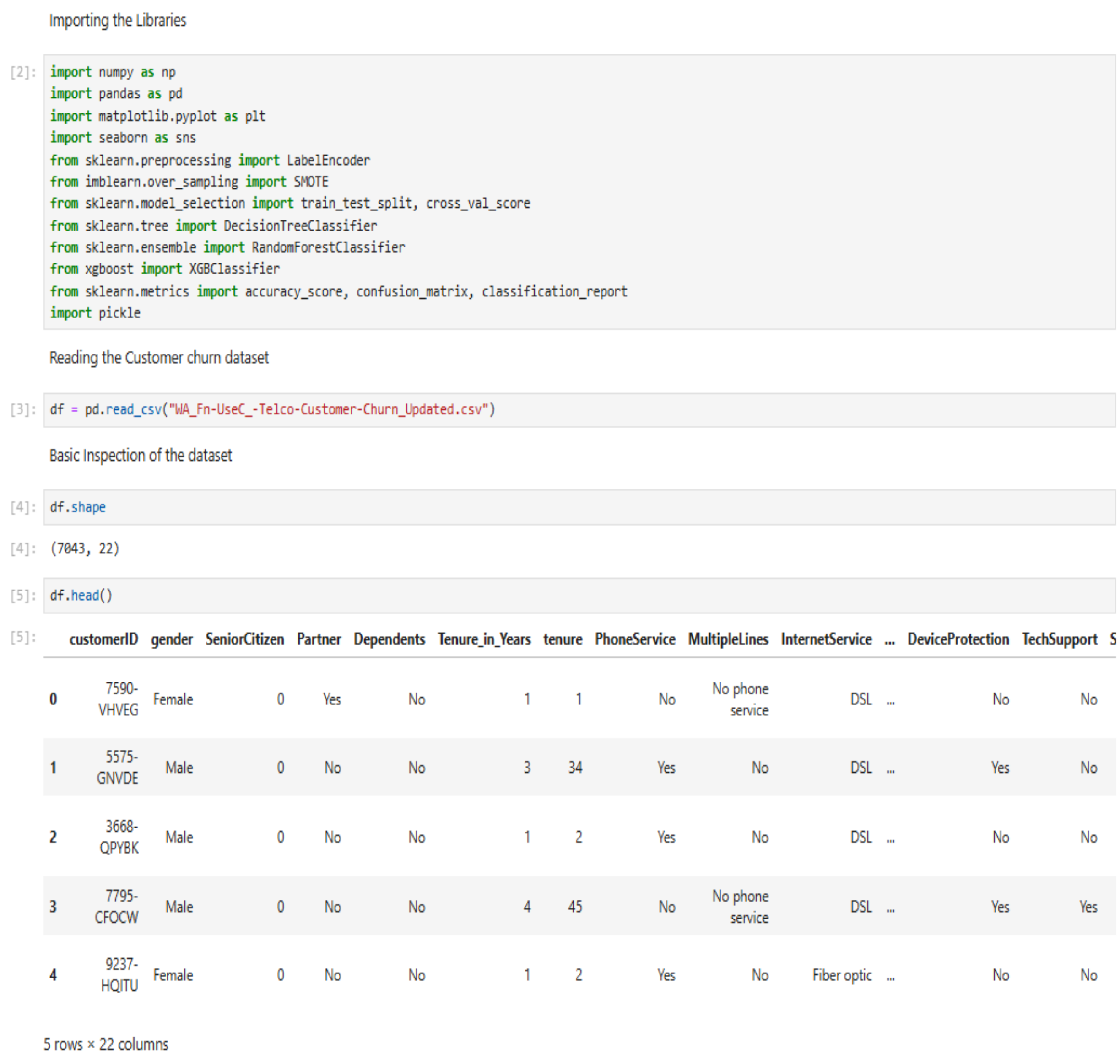
A successful churn prediction project requires careful data collection, preprocessing, feature engineering, model selection, hyperparameter tuning, model evaluation, and deployment. By following these steps, businesses can build accurate and effective churn prediction models that can be used to identify at-risk customers and take targeted actions to reduce churn.

Churn prediction provides insights into customer behaviour and preferences, allowing businesses to make more informed decisions about product development, marketing, and customer service. Overall, churn prediction is an essential tool for businesses looking to improve customer retention, reduce churn, and maximize revenue. By leveraging advanced analytics techniques and machine learning algorithms, businesses can stay ahead of the competition and achieve long-term success.

Annexures:

(Decision Tree, Random Forest, XGBoost)

Here we have provided the Sample code



Importing the Libraries

```
[2]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.preprocessing import LabelEncoder
     from imblearn.over_sampling import SMOTE
     from sklearn.model_selection import train_test_split, cross_val_score
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.ensemble import RandomForestClassifier
     from xgboost import XGBClassifier
     from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
     import pickle
```

Reading the Customer churn dataset

```
[3]: df = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn_Updated.csv")
```

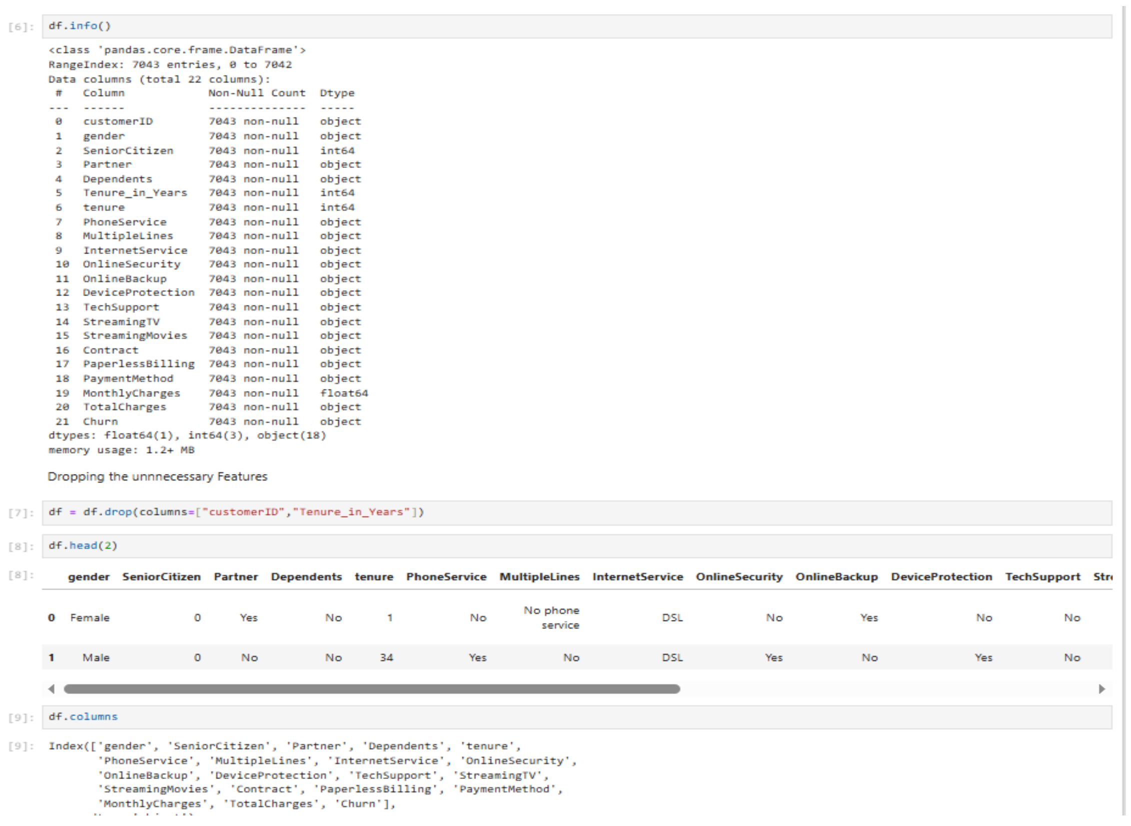
Basic Inspection of the dataset

```
[4]: df.shape

[4]: (7043, 22)

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

[5]:

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

5 rows × 22 columns



```
[6]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 22 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_in_Years   7043 non-null   int64
 6   tenure            7043 non-null   int64
 7   PhoneService      7043 non-null   object
 8   MultipleLines     7043 non-null   object
 9   InternetService   7043 non-null   object
 10  OnlineSecurity    7043 non-null   object
 11  OnlineBackup      7043 non-null   object
 12  DeviceProtection  7043 non-null   object
 13  TechSupport       7043 non-null   object
 14  StreamingTV       7043 non-null   object
 15  StreamingMovies   7043 non-null   object
 16  Contract          7043 non-null   object
 17  PaperlessBilling  7043 non-null   object
 18  PaymentMethod     7043 non-null   object
 19  MonthlyCharges    7043 non-null   float64
 20  TotalCharges      7043 non-null   object
 21  Churn             7043 non-null   object
dtypes: float64(1), int64(3), object(18)
memory usage: 1.2+ MB
```

Dropping the unnnecessary Features

```
[7]: df = df.drop(columns=["customerID","Tenure_in_Years"])
```

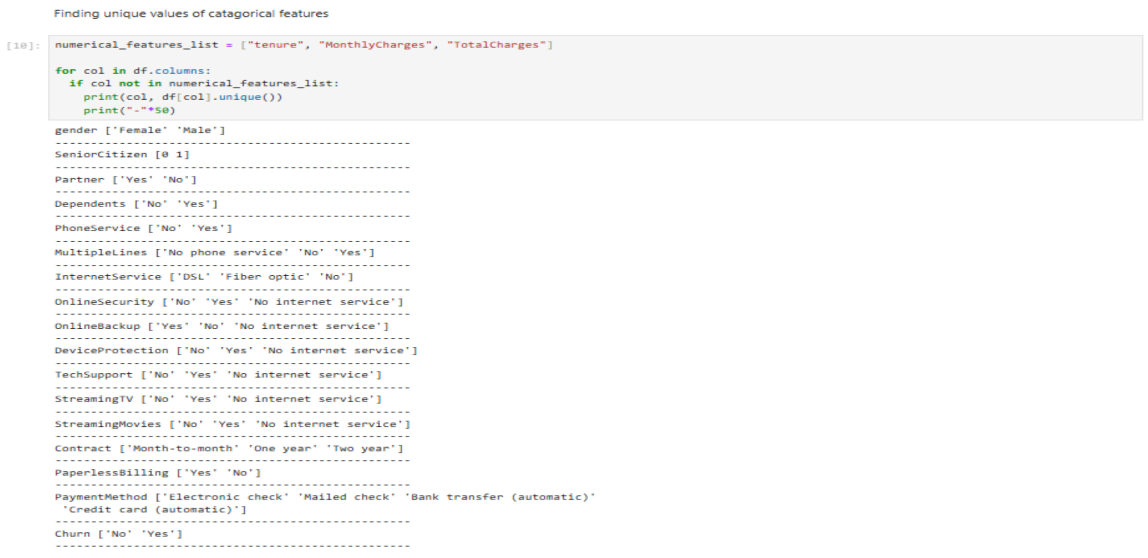
```
[8]: df.head(2)
```

[8]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	Str
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No	No	
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	No	

```
[9]: df.columns
```

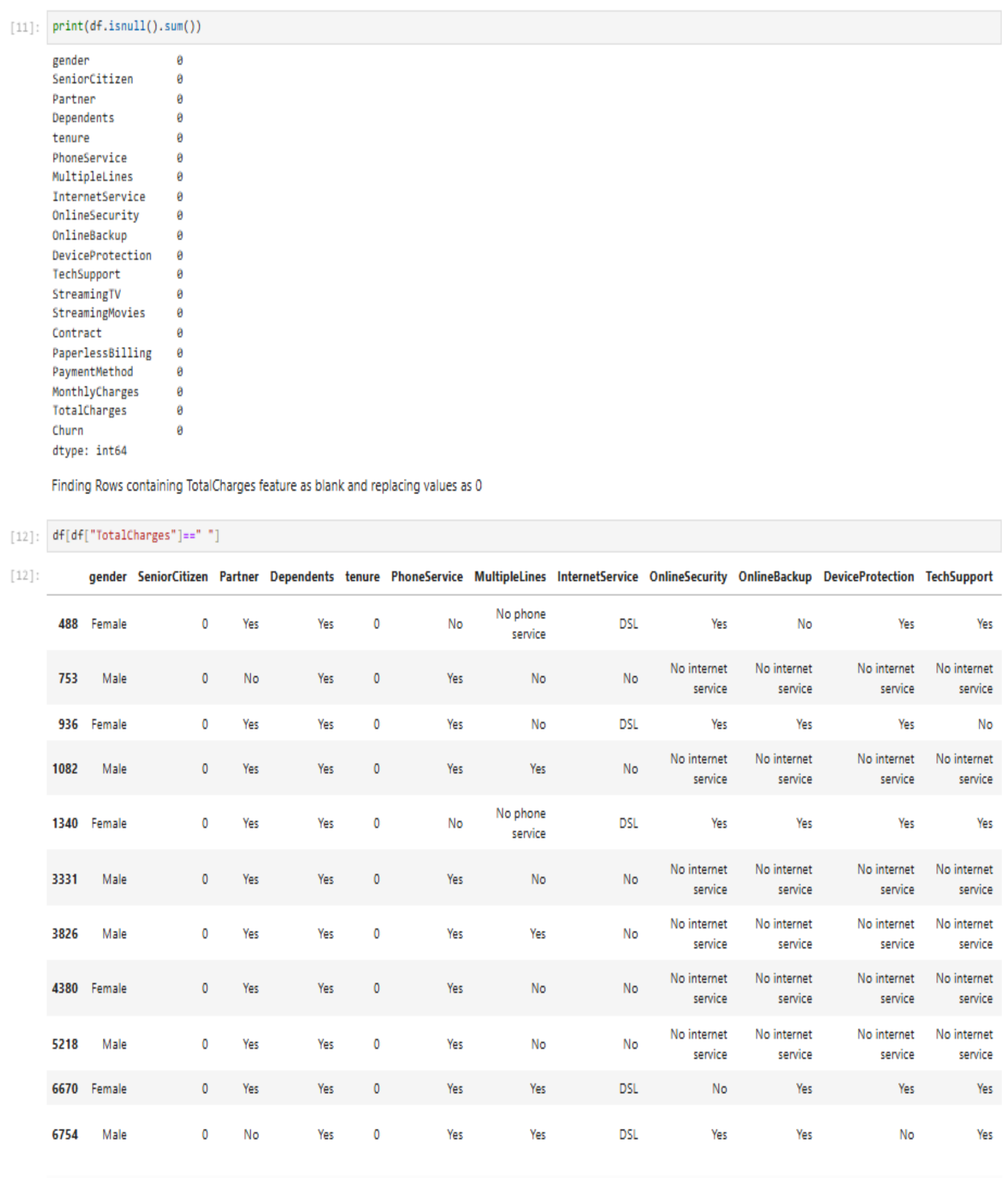
```
[9]: Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod',
       'MonthlyCharges', 'TotalCharges', 'Churn'],
```



Finding unique values of catagorical features

```
[10]: numerical_features_list = ["tenure", "MonthlyCharges", "TotalCharges"]

      for col in df.columns:
        if col not in numerical_features_list:
          print(col, df[col].unique())
          print("-"*50)
```

```
gender ['Female' 'Male']
--------------------------------------------------
SeniorCitizen [0 1]
--------------------------------------------------
Partner ['Yes' 'No']
--------------------------------------------------
Dependents ['No' 'Yes']
--------------------------------------------------
PhoneService ['No' 'Yes']
--------------------------------------------------
MultipleLines ['No phone service' 'No' 'Yes']
--------------------------------------------------
InternetService ['DSL' 'Fiber optic' 'No']
--------------------------------------------------
OnlineSecurity ['No' 'Yes' 'No internet service']
--------------------------------------------------
OnlineBackup ['Yes' 'No' 'No internet service']
--------------------------------------------------
DeviceProtection ['No' 'Yes' 'No internet service']
--------------------------------------------------
TechSupport ['No' 'Yes' 'No internet service']
--------------------------------------------------
StreamingTV ['No' 'Yes' 'No internet service']
--------------------------------------------------
StreamingMovies ['No' 'Yes' 'No internet service']
--------------------------------------------------
Contract ['Month-to-month' 'One year' 'Two year']
--------------------------------------------------
PaperlessBilling ['Yes' 'No']
--------------------------------------------------
PaymentMethod ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
--------------------------------------------------
Churn ['No' 'Yes']
--------------------------------------------------
```



```
[11]: print(df.isnull().sum())
```

```
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

Finding Rows containing TotalCharges feature as blank and replacing values as 0

```
[12]: df[df["TotalCharges"]==" "]
```

[12]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport
488	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	No	Yes	Yes
753	Male	0	No	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service
936	Female	0	Yes	Yes	0	Yes	No	DSL	Yes	Yes	Yes	No
1082	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No internet service	No internet service
1340	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	Yes	Yes	Yes
3331	Male	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service
3826	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No internet service	No internet service
4380	Female	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service
5218	Male	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service
6670	Female	0	Yes	Yes	0	Yes	Yes	DSL	No	Yes	Yes	Yes
6754	Male	0	No	Yes	0	Yes	Yes	DSL	Yes	Yes	No	Yes

```
[13]: df["TotalCharges"] = df["TotalCharges"].replace({" ": "0.0"})
      df["TotalCharges"] = df["TotalCharges"].astype(float)
```

Here we can noticed that target column distribution is imbalanced.

```
[14]: print(df["Churn"].value_counts())
```

```
Churn
No      5174
Yes     1869
Name: count, dtype: int64
```

```
[15]: df.shape
```

```
[15]: (7043, 20)
```

```
[16]: df.describe()
```

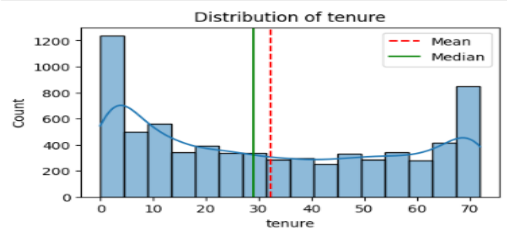
[16]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692	2279.734304
std	0.368612	24.559481	30.090047	2266.794470
min	0.000000	0.000000	18.250000	0.000000
25%	0.000000	9.000000	35.500000	398.550000
50%	0.000000	29.000000	70.350000	1394.550000
75%	0.000000	55.000000	89.850000	3786.600000
max	1.000000	72.000000	118.750000	8684.800000

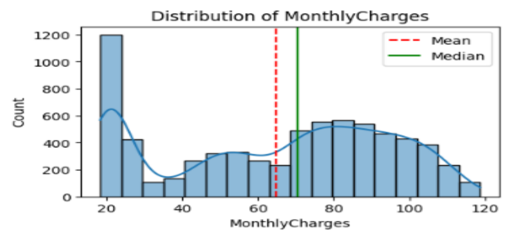
Performing basic EDA to get the understnading of the dataset.

```
[18]: def plot_histogram(df, column_name):

        plt.figure(figsize=(5, 3))
        sns.histplot(df[column_name], kde=True)
        plt.title(f"Distribution of {column_name}")

        # calculate the mean and median values for the columns
        col_mean = df[column_name].mean()
        col_median = df[column_name].median()

        # add vertical lines for mean and median
        plt.axvline(col_mean, color="red", linestyle="--", label="Mean")
        plt.axvline(col_median, color="green", linestyle="-", label="Median")

        plt.legend()

        plt.show()
```

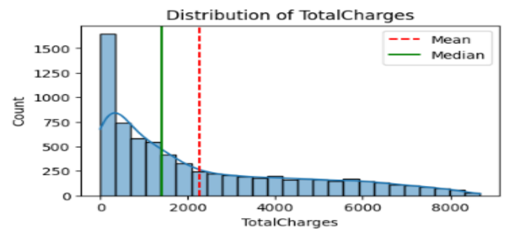
```
[19]: plot_histogram(df, "tenure")
```



```
[20]: plot_histogram(df, "MonthlyCharges")
```

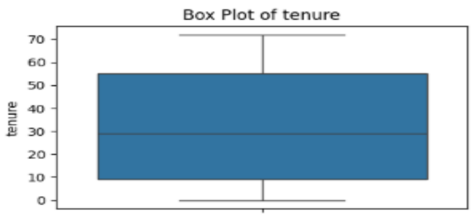


```
[21]: plot_histogram(df, "TotalCharges")
```

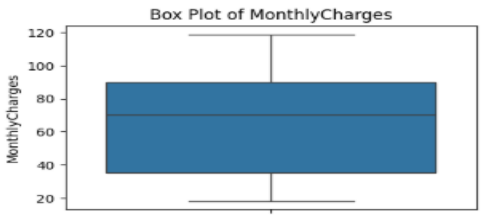


```
[22]: def plot_boxplot(df, column_name):

        plt.figure(figsize=(5, 3))
        sns.boxplot(y=df[column_name])
        plt.title(f"Box Plot of {column_name}")
        plt.ylabel(column_name)
        plt.show
```

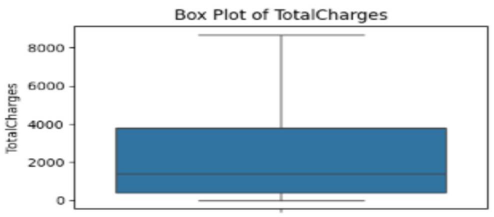
```
[23]: plot_boxplot(df, "tenure")
```



```
[24]: plot_boxplot(df, "MonthlyCharges")
```

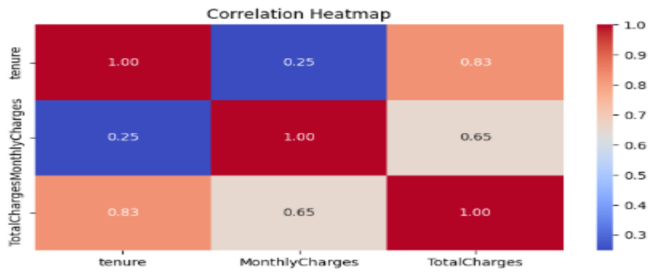


```
[25]: plot_boxplot(df, "TotalCharges")
```



Plotting correlation matrix between numerical features

```
[26]: plt.figure(figsize=(8, 4))
      sns.heatmap(df[["tenure", "MonthlyCharges", "TotalCharges"]].corr(), annot=True, cmap="coolwarm", fmt=".2f")
      plt.title("Correlation Heatmap")
      plt.show()
```



Converting catagorical values into numeric for churn feature.

```
[27]: df["Churn"] = df["Churn"].replace({"Yes": 1, "No": 0})
```

```
C:\Users\Sakshi\AppData\Local\Temp\ipykernel_13612\2364848822.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed i
n a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option
('future.no_silent_downcasting', True)`
  df["Churn"] = df["Churn"].replace({"Yes": 1, "No": 0})
```

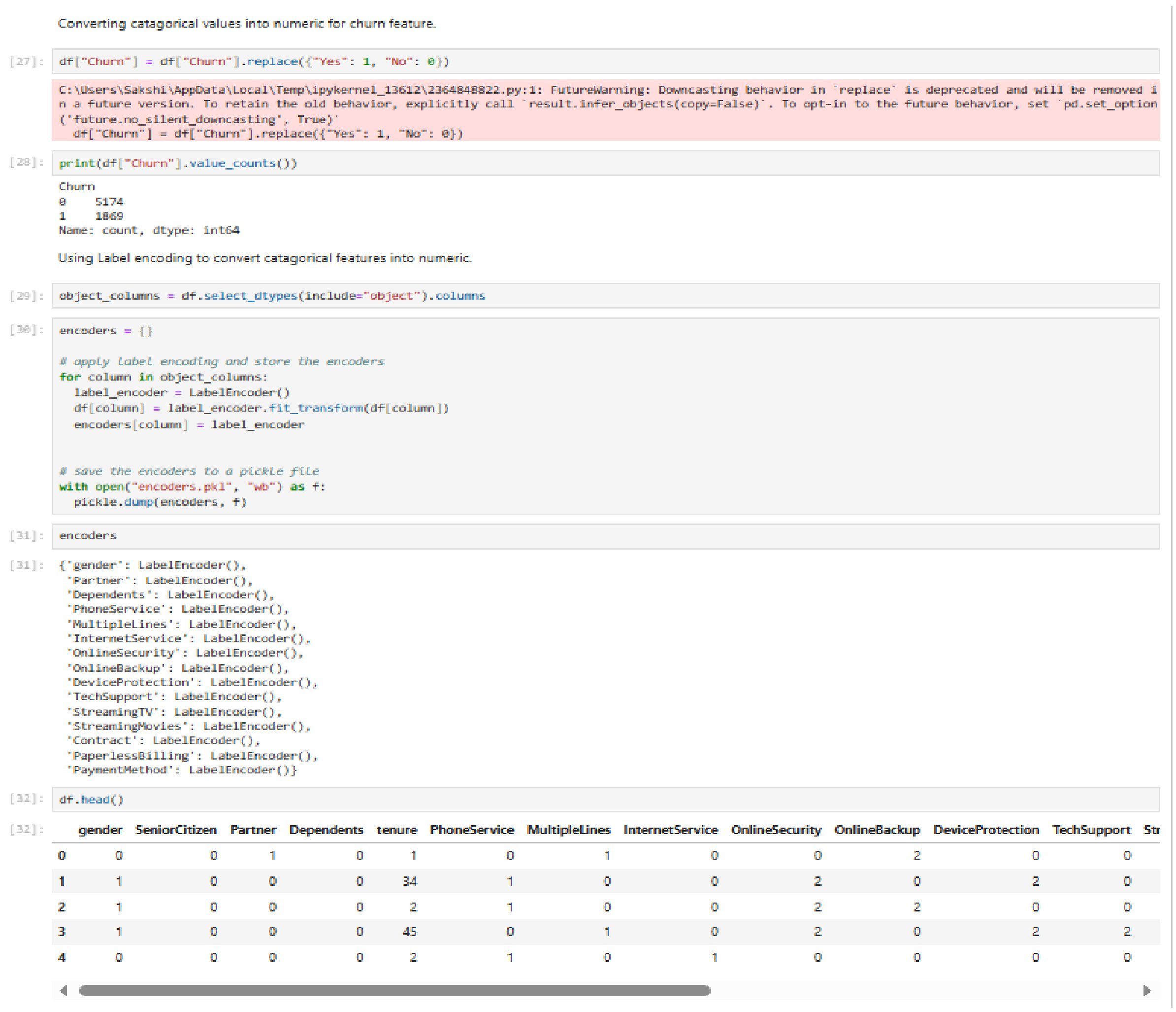
```
[28]: print(df["Churn"].value_counts())
```

```
Churn
0    5174
1    1869
Name: count, dtype: int64
```

Using Label encoding to convert catagorical features into numeric.

```
[29]: object_columns = df.select_dtypes(include="object").columns
```

```
[30]: encoders = {}

      # apply label encoding and store the encoders
      for column in object_columns:
        label_encoder = LabelEncoder()
        df[column] = label_encoder.fit_transform(df[column])
        encoders[column] = label_encoder


      # save the encoders to a pickle file
      with open("encoders.pkl", "wb") as f:
        pickle.dump(encoders, f)
```



Converting catagorical values into numeric for churn feature.

```
[27]: df["Churn"] = df["Churn"].replace({"Yes": 1, "No": 0})
```

```
C:\Users\Sakshi\AppData\Local\Temp\ipykernel_13612\2364848822.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed i
n a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option
('future.no_silent_downcasting', True)`
  df["Churn"] = df["Churn"].replace({"Yes": 1, "No": 0})
```

```
[28]: print(df["Churn"].value_counts())
```

```
Churn
0    5174
1    1869
Name: count, dtype: int64
```

Using Label encoding to convert catagorical features into numeric.

```
[29]: object_columns = df.select_dtypes(include="object").columns
```

```
[30]: encoders = {}

      # apply label encoding and store the encoders
      for column in object_columns:
        label_encoder = LabelEncoder()
        df[column] = label_encoder.fit_transform(df[column])
        encoders[column] = label_encoder


      # save the encoders to a pickle file
      with open("encoders.pkl", "wb") as f:
        pickle.dump(encoders, f)
```

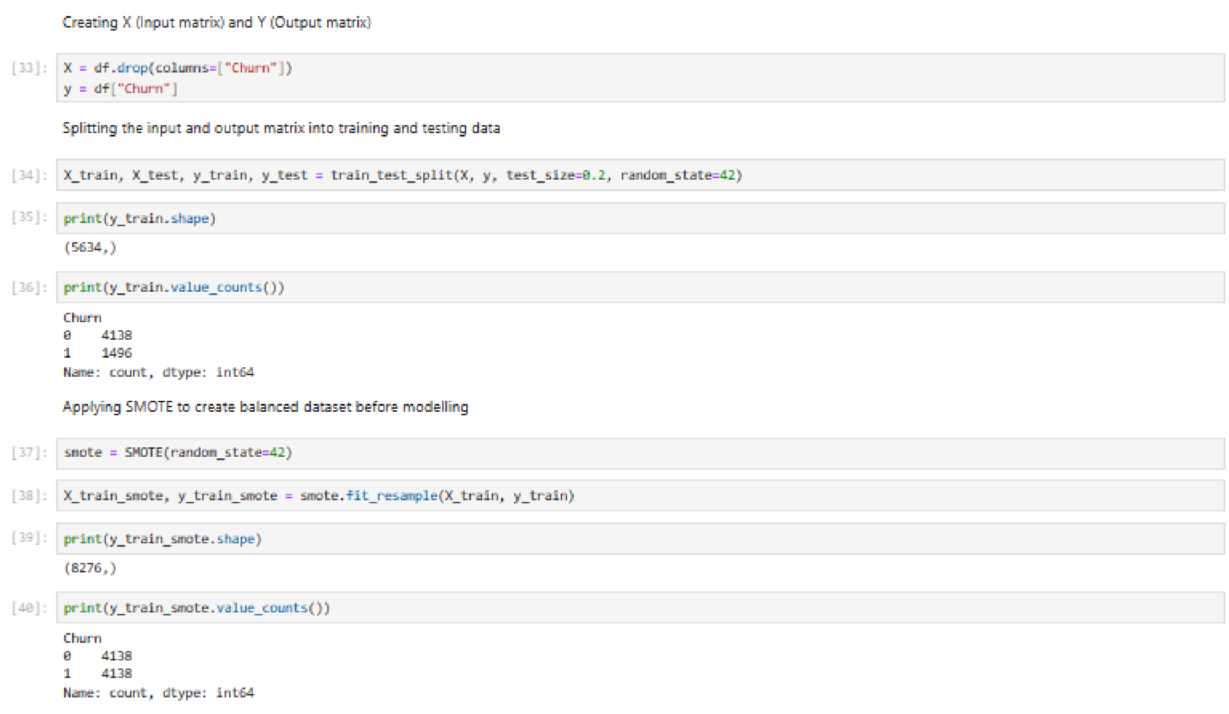
```
[31]: encoders
```

```
[31]: {'gender': LabelEncoder(),
       'Partner': LabelEncoder(),
       'Dependents': LabelEncoder(),
       'PhoneService': LabelEncoder(),
       'MultipleLines': LabelEncoder(),
       'InternetService': LabelEncoder(),
       'OnlineSecurity': LabelEncoder(),
       'OnlineBackup': LabelEncoder(),
       'DeviceProtection': LabelEncoder(),
       'TechSupport': LabelEncoder(),
       'StreamingTV': LabelEncoder(),
       'StreamingMovies': LabelEncoder(),
       'Contract': LabelEncoder(),
       'PaperlessBilling': LabelEncoder(),
       'PaymentMethod': LabelEncoder()}
```

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

[32]:

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



Creating X (Input matrix) and Y (Output matrix)

```
[33]: X = df.drop(columns=["Churn"])
      y = df["Churn"]
```

Splitting the input and output matrix into training and testing data

```
[34]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[35]: print(y_train.shape)
```

```
(5634,)
```

```
[36]: print(y_train.value_counts())
```

```
Churn
0    4138
1    1496
Name: count, dtype: int64
```

Applying SMOTE to create balanced dataset before modelling

```
[37]: smote = SMOTE(random_state=42)
```

```
[38]: X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

```
[39]: print(y_train_smote.shape)
```

```
(8276,)
```

```
[40]: print(y_train_smote.value_counts())
```

```
Churn
0    4138
1    4138
Name: count, dtype: int64
```

Training 3 different models to compare the model performance

```
[41]: models = {
          "Decision Tree": DecisionTreeClassifier(random_state=42),
          "Random Forest": RandomForestClassifier(random_state=42),
          "XGBoost": XGBClassifier(random_state=42)
      }
```

Using Cross validation technique while creating a model

Here we got average cross validation score for each type of model

```
[42]: # dictionary to store the cross validation results
      cv_scores = {}

      # perform 5-fold cross validation for each model
      for model_name, model in models.items():
        print(f"Training {model_name} with default parameters")
        scores = cross_val_score(model, X_train_smote, y_train_smote, cv=5, scoring="accuracy")
        cv_scores[model_name] = scores
        print(f"{model_name} cross-validation accuracy: {np.mean(scores):.2f}")
        print("-"*70)
```

```
Training Decision Tree with default parameters
Decision Tree cross-validation accuracy: 0.78
----------------------------------------------------------------------
Training Random Forest with default parameters
Random Forest cross-validation accuracy: 0.84
----------------------------------------------------------------------
Training XGBoost with default parameters
XGBoost cross-validation accuracy: 0.83
----------------------------------------------------------------------
```

```
[43]: cv_scores
```

```
[43]: {'Decision Tree': array([0.68297101, 0.71601208, 0.81993958, 0.83564955, 0.83746224]),
       'Random Forest': array([0.72826087, 0.7734139 , 0.90332326, 0.89969789, 0.8978852 ]),
       'XGBoost': array([0.71135266, 0.74864048, 0.91178248, 0.88640483, 0.91117825])}
```

Using hyper parameter tuning method using GridSearchCV to find the best combination for Random forest Classifier model

```
[44]: from sklearn.model_selection import train_test_split
```

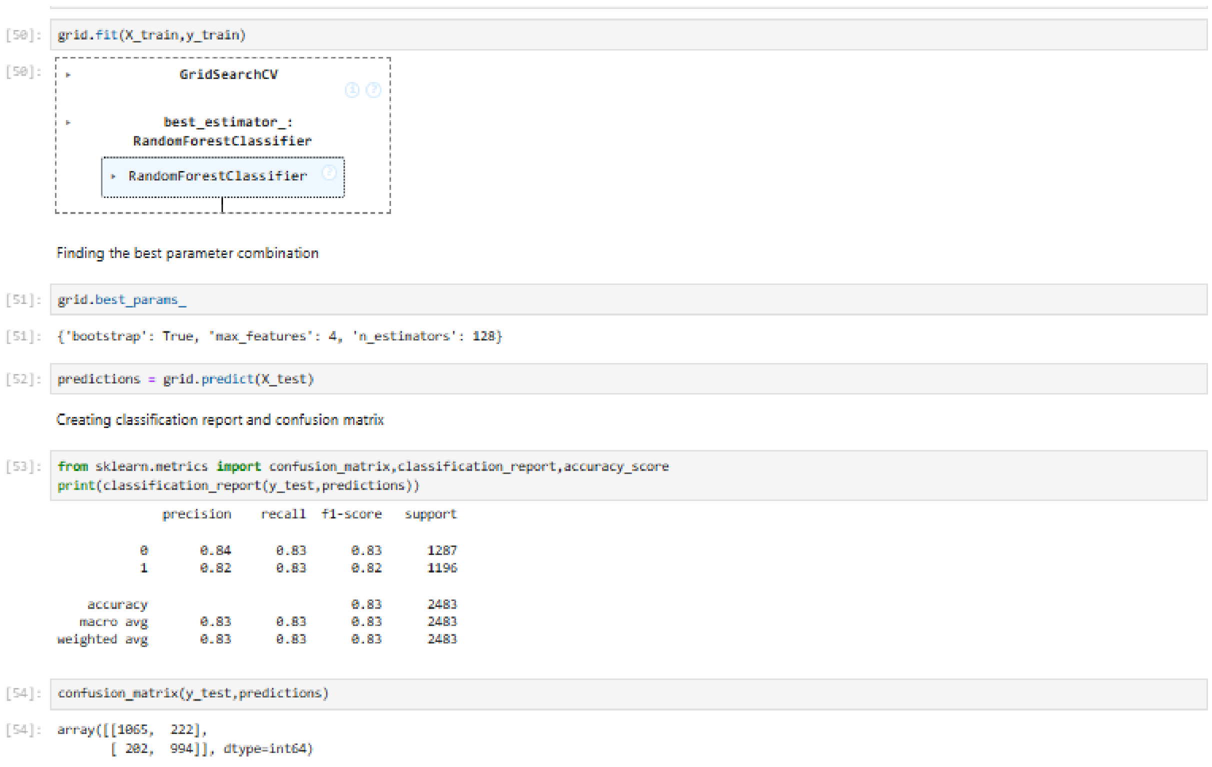
```
[45]: X_train, X_test, y_train, y_test = train_test_split(X_train_smote,y_train_smote, test_size=0.3, random_state=101)
```

```
[46]: from sklearn.model_selection import GridSearchCV
```

```
[47]: n_estimators=[64,100,128,200]
      max_features= [2,3,4]
      bootstrap = [True,False]
```

```
[48]: param_grid = {'n_estimators':n_estimators,
                   'max_features':max_features,
                   'bootstrap':bootstrap}
```

```
[49]: from sklearn.ensemble import RandomForestClassifier
      rfc = RandomForestClassifier()
      grid = GridSearchCV(rfc,param_grid)
```



```
[50]: grid.fit(X_train,y_train)
```

```
[50]: GridSearchCV
      best_estimator_:
      RandomForestClassifier
      RandomForestClassifier
```

Finding the best parameter combination

```
[51]: grid.best_params_
```

```
[51]: {'bootstrap': True, 'max_features': 4, 'n_estimators': 128}
```

```
[52]: predictions = grid.predict(X_test)
```

Creating classification report and confusion matrix

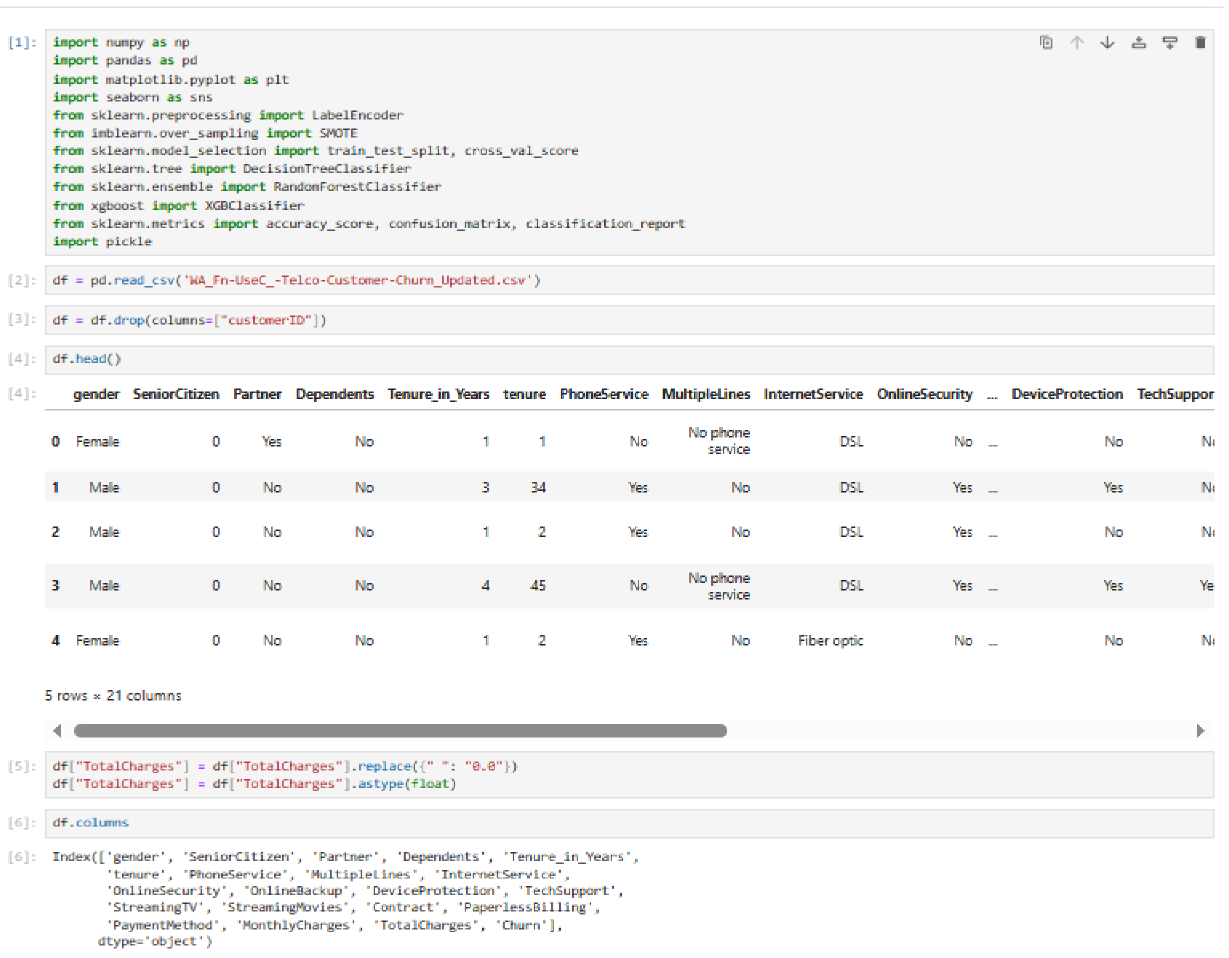
```
[53]: from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
      print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.84      0.83      0.83      1287
           1       0.82      0.83      0.82      1196

    accuracy                           0.83      2483
   macro avg       0.83      0.83      0.83      2483
weighted avg       0.83      0.83      0.83      2483
```

```
[54]: confusion_matrix(y_test,predictions)
```

```
[54]: array([[1065,  222],
             [ 202,  994]], dtype=int64)
```

SVM(Support Vector Machines): Here we have provided the Sample code



```
[1]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.preprocessing import LabelEncoder
     from imblearn.over_sampling import SMOTE
     from sklearn.model_selection import train_test_split, cross_val_score
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.ensemble import RandomForestClassifier
     from xgboost import XGBClassifier
     from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
     import pickle
```

```
[2]: df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn_Updated.csv')
```

```
[3]: df = df.drop(columns=["customerID"])
```

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

[4]:

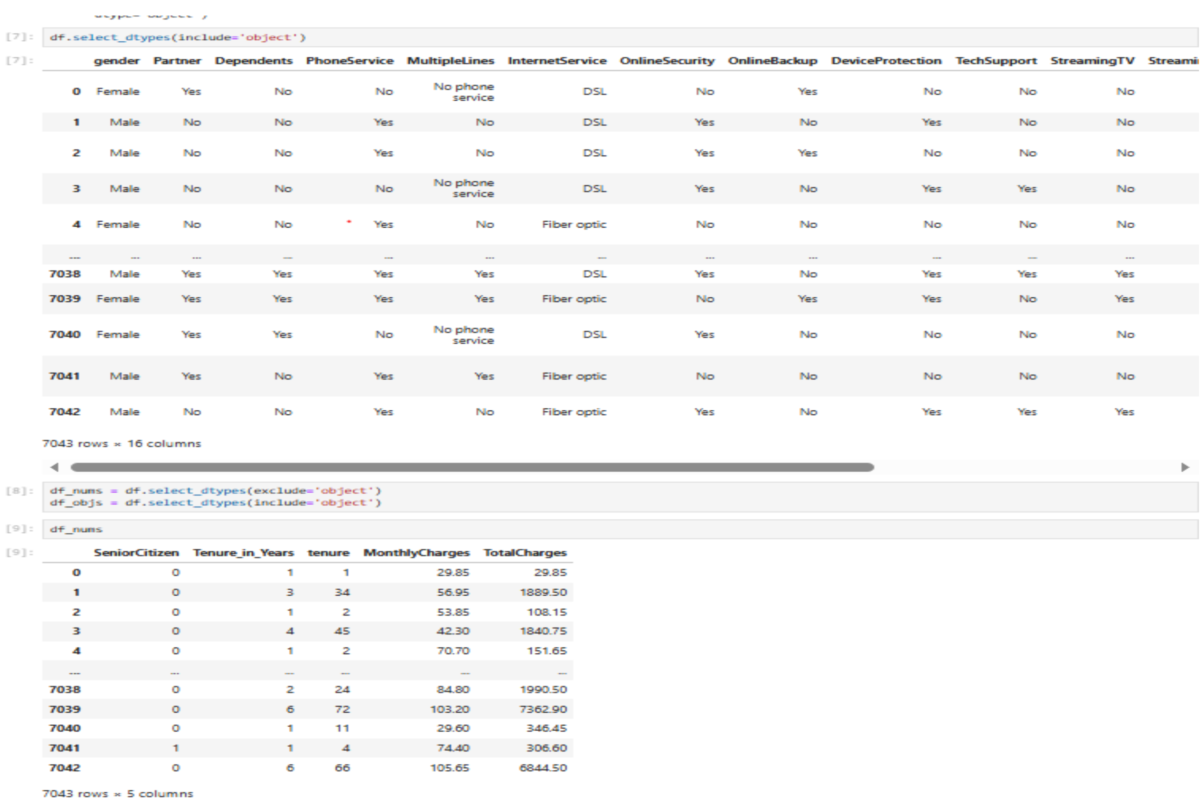
	gender	SeniorCitizen	Partner	Dependents	Tenure_in_Years	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSuppor
0	Female	0	Yes	No	1	1	No	No phone service	DSL	No	...	No	N
1	Male	0	No	No	3	34	Yes	No	DSL	Yes	...	Yes	N
2	Male	0	No	No	1	2	Yes	No	DSL	Yes	...	No	N
3	Male	0	No	No	4	45	No	No phone service	DSL	Yes	...	Yes	Ye
4	Female	0	No	No	1	2	Yes	No	Fiber optic	No	...	No	N

5 rows × 21 columns

```
[5]: df["TotalCharges"] = df["TotalCharges"].replace({" ": "0.0"})
     df["TotalCharges"] = df["TotalCharges"].astype(float)
```

```
[6]: df.columns
```

```
[6]: Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'Tenure_in_Years',
            'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
            'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
            'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
            'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
           dtype='object')
```



```
      dtype='object')
[7]:  df.select_dtypes(include='object')
[7]:
```

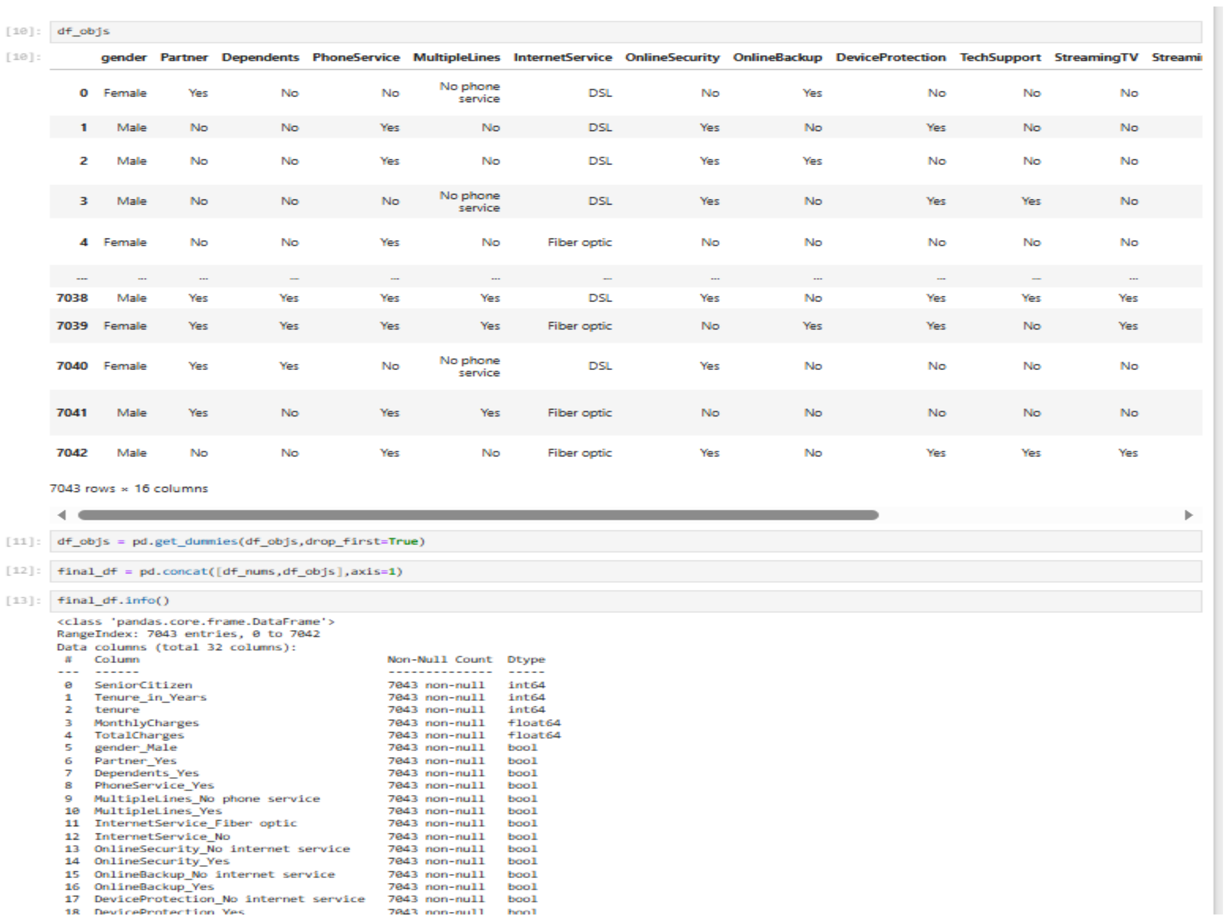
	gender	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	Streami
0	Female	Yes	No	No	No phone service	DSL	No	Yes	No	No	No	
1	Male	No	No	Yes	No	DSL	Yes	No	Yes	No	No	
2	Male	No	No	Yes	No	DSL	Yes	Yes	No	No	No	
3	Male	No	No	No	No phone service	DSL	Yes	No	Yes	Yes	No	
4	Female	No	No	Yes	No	Fiber optic	No	No	No	No	No	
...	...	...	...	...	...	...	...	...	...	...	...	
7038	Male	Yes	Yes	Yes	Yes	DSL	Yes	No	Yes	Yes	Yes	
7039	Female	Yes	Yes	Yes	Yes	Fiber optic	No	Yes	Yes	No	Yes	
7040	Female	Yes	Yes	No	No phone service	DSL	Yes	No	No	No	No	
7041	Male	Yes	No	Yes	Yes	Fiber optic	No	No	No	No	No	
7042	Male	No	No	Yes	No	Fiber optic	Yes	No	Yes	Yes	Yes	

7043 rows × 16 columns

```
[8]:  df_nums = df.select_dtypes(exclude='object')
      df_objs = df.select_dtypes(include='object')

[9]:  df_nums
[9]:
```

	SeniorCitizen	Tenure_in_Years	tenure	MonthlyCharges	TotalCharges
0	0	1	1	29.85	29.85
1	0	3	34	56.95	1889.50
2	0	1	2	53.85	108.15
3	0	4	45	42.30	1840.75
4	0	1	2	70.70	151.65
...	...	...	...	...	...
7038	0	2	24	84.80	1990.50
7039	0	6	72	103.20	7362.90
7040	0	1	11	29.60	346.45
7041	1	1	4	74.40	306.60
7042	0	6	66	105.65	6844.50

7043 rows × 5 columns



```
[10]:  df_objs
[10]:
```

	gender	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	Streami
0	Female	Yes	No	No	No phone service	DSL	No	Yes	No	No	No	
1	Male	No	No	Yes	No	DSL	Yes	No	Yes	No	No	
2	Male	No	No	Yes	No	DSL	Yes	Yes	No	No	No	
3	Male	No	No	No	No phone service	DSL	Yes	No	Yes	Yes	No	
4	Female	No	No	Yes	No	Fiber optic	No	No	No	No	No	
...	...	...	...	...	...	...	...	...	...	...	...	
7038	Male	Yes	Yes	Yes	Yes	DSL	Yes	No	Yes	Yes	Yes	
7039	Female	Yes	Yes	Yes	Yes	Fiber optic	No	Yes	Yes	No	Yes	
7040	Female	Yes	Yes	No	No phone service	DSL	Yes	No	No	No	No	
7041	Male	Yes	No	Yes	Yes	Fiber optic	No	No	No	No	No	
7042	Male	No	No	Yes	No	Fiber optic	Yes	No	Yes	Yes	Yes	

7043 rows × 16 columns

```
[11]:  df_objs = pd.get_dummies(df_objs,drop_first=True)

[12]:  final_df = pd.concat([df_nums,df_objs],axis=1)

[13]:  final_df.info()

       <class 'pandas.core.frame.DataFrame'>
       RangeIndex: 7043 entries, 0 to 7042
       Data columns (total 32 columns):
        #   Column                                  Non-Null Count  Dtype  
       ---  ------                                  --------------  -----  
        0   SeniorCitizen                           7043 non-null   int64  
        1   Tenure_in_Years                         7043 non-null   int64  
        2   tenure                                  7043 non-null   int64  
        3   MonthlyCharges                          7043 non-null   float64
        4   TotalCharges                            7043 non-null   float64
        5   gender_Male                             7043 non-null   bool   
        6   Partner_Yes                             7043 non-null   bool   
        7   Dependents_Yes                          7043 non-null   bool   
        8   PhoneService_Yes                        7043 non-null   bool   
        9   MultipleLines_No phone service          7043 non-null   bool   
        10  MultipleLines_Yes                       7043 non-null   bool   
        11  InternetService_Fiber optic             7043 non-null   bool   
        12  InternetService_No                      7043 non-null   bool   
        13  OnlineSecurity_No internet service      7043 non-null   bool   
        14  OnlineSecurity_Yes                      7043 non-null   bool   
        15  OnlineBackup_No internet service        7043 non-null   bool   
        16  OnlineBackup_Yes                        7043 non-null   bool   
        17  DeviceProtection_No internet service    7043 non-null   bool   
        18  DeviceProtection_Yes                    7043 non-null   bool
```

```
[14]: X = final_df.drop('Churn_Yes',axis=1)
      y = final_df['Churn_Yes']
```

```
[15]: from sklearn.model_selection import train_test_split
```

```
[16]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

```
[17]: from sklearn.preprocessing import StandardScaler
```

```
[18]: scaler = StandardScaler()
```

```
[19]: scaled_X_train = scaler.fit_transform(X_train)
      scaled_X_test = scaler.transform(X_test)
```

```
[20]: from sklearn.svm import SVC
```

```
[21]: from sklearn.model_selection import GridSearchCV
```

```
[22]: svc = SVC(class_weight='balanced')
```

```
[23]: param_grid = {'C':[0.001,0.01,0.1,0.5,1],'gamma':['scale','auto']}
      grid = GridSearchCV(svc,param_grid)
```

```
[24]: grid.fit(scaled_X_train,y_train)
```

```
[24]:  ▸ GridSearchCV

       ▸ best_estimator_:
             SVC

          ▸ SVC
```

```
[25]: grid.best_params_
```

```
[25]: {'C': 0.5, 'gamma': 'scale'}
```

```
[26]: from sklearn.metrics import confusion_matrix,classification_report
```

```
[27]: grid_pred = grid.predict(scaled_X_test)
```

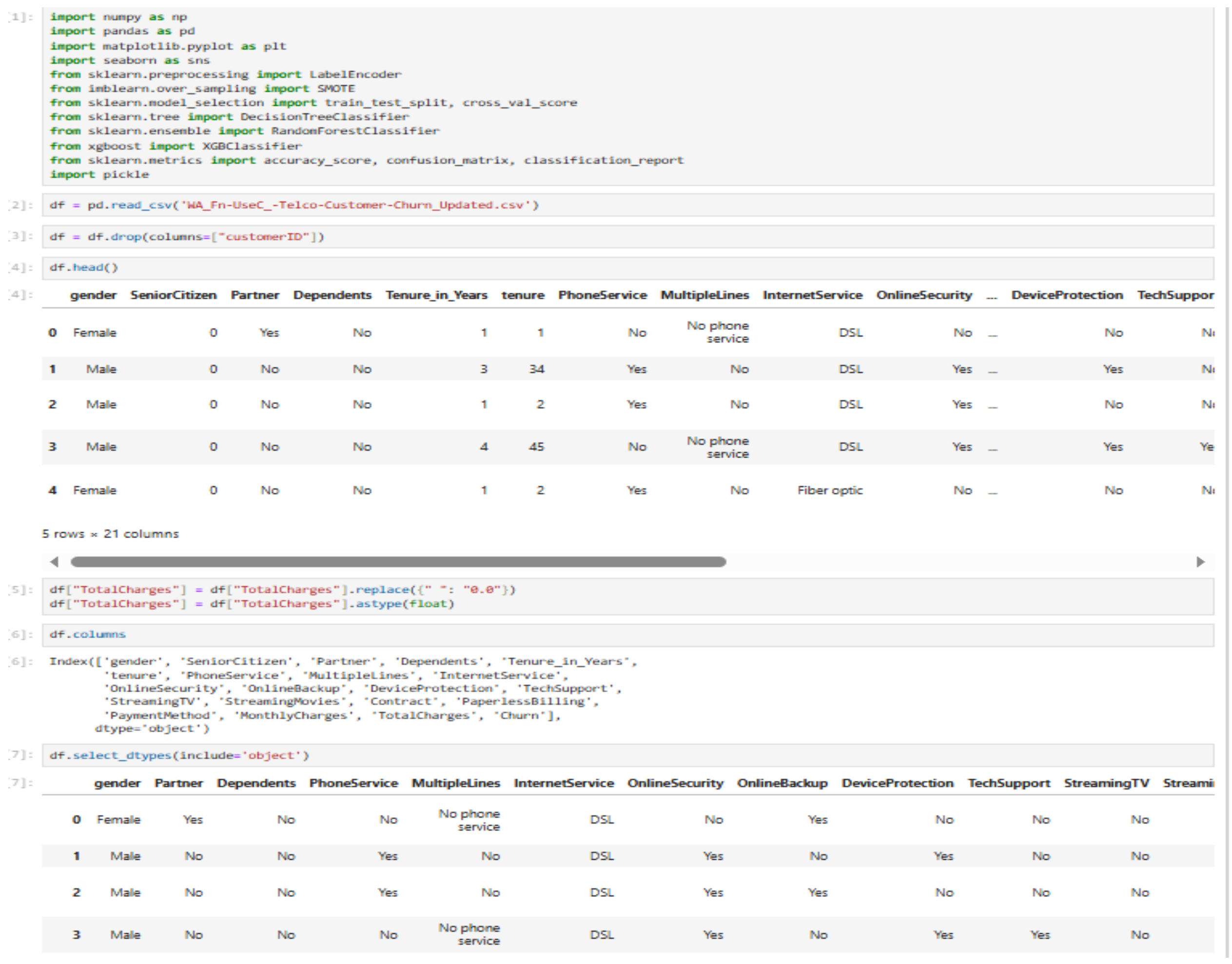
```
[28]: confusion_matrix(y_test,grid_pred)
```

```
[28]: array([[1145,  401],
             [ 123,  444]], dtype=int64)
```

```
[29]: print(classification_report(y_test,grid_pred))
```

```
              precision    recall  f1-score   support

       False       0.90      0.74      0.81      1546
        True       0.53      0.78      0.63       567

    accuracy                           0.75      2113
   macro avg       0.71      0.76      0.72      2113
weighted avg       0.80      0.75      0.76      2113
```

<u>Logistic Classification:</u> Here we have provided the Sample code



```
[1]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.preprocessing import LabelEncoder
     from imblearn.over_sampling import SMOTE
     from sklearn.model_selection import train_test_split, cross_val_score
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.ensemble import RandomForestClassifier
     from xgboost import XGBClassifier
     from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
     import pickle
```

```
[2]: df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn_Updated.csv')
```

```
[3]: df = df.drop(columns=["customerID"])
```

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

[4]:

	gender	SeniorCitizen	Partner	Dependents	Tenure_in_Years	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSuppor
0	Female	0	Yes	No	1	1	No	No phone service	DSL	No	...	No	N
1	Male	0	No	No	3	34	Yes	No	DSL	Yes	...	Yes	N
2	Male	0	No	No	1	2	Yes	No	DSL	Yes	...	No	N
3	Male	0	No	No	4	45	No	No phone service	DSL	Yes	...	Yes	Ye
4	Female	0	No	No	1	2	Yes	No	Fiber optic	No	...	No	N

5 rows × 21 columns

```
[5]: df["TotalCharges"] = df["TotalCharges"].replace({" ": "0.0"})
     df["TotalCharges"] = df["TotalCharges"].astype(float)
```

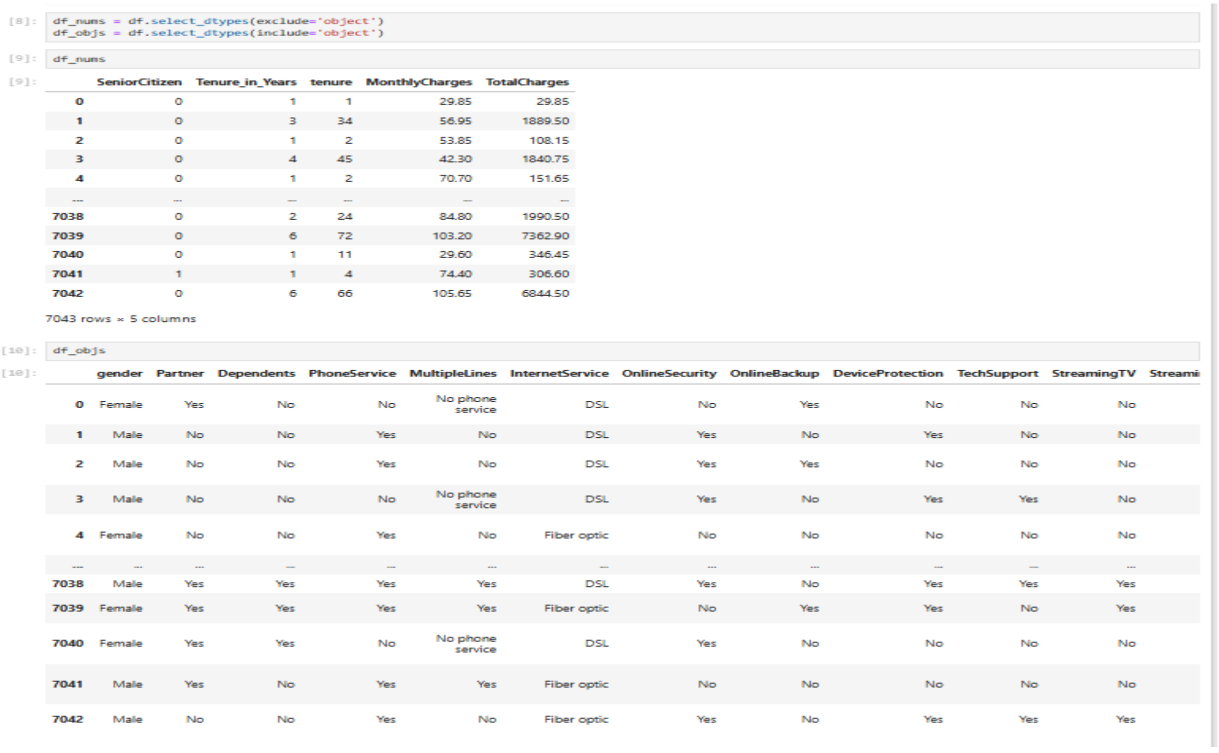
```
[6]: df.columns
```

```
[6]: Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'Tenure_in_Years',
            'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
            'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
            'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
            'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
           dtype='object')
```

```
[7]: df.select_dtypes(include='object')
```

[7]:

	gender	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	Streami
0	Female	Yes	No	No	No phone service	DSL	No	Yes	No	No	No	
1	Male	No	No	Yes	No	DSL	Yes	No	Yes	No	No	
2	Male	No	No	Yes	No	DSL	Yes	Yes	No	No	No	
3	Male	No	No	No	No phone service	DSL	Yes	No	Yes	Yes	No	



```
[8]: df_nums = df.select_dtypes(exclude='object')
     df_objs = df.select_dtypes(include='object')
```

```
[9]: df_nums
```

[9]:

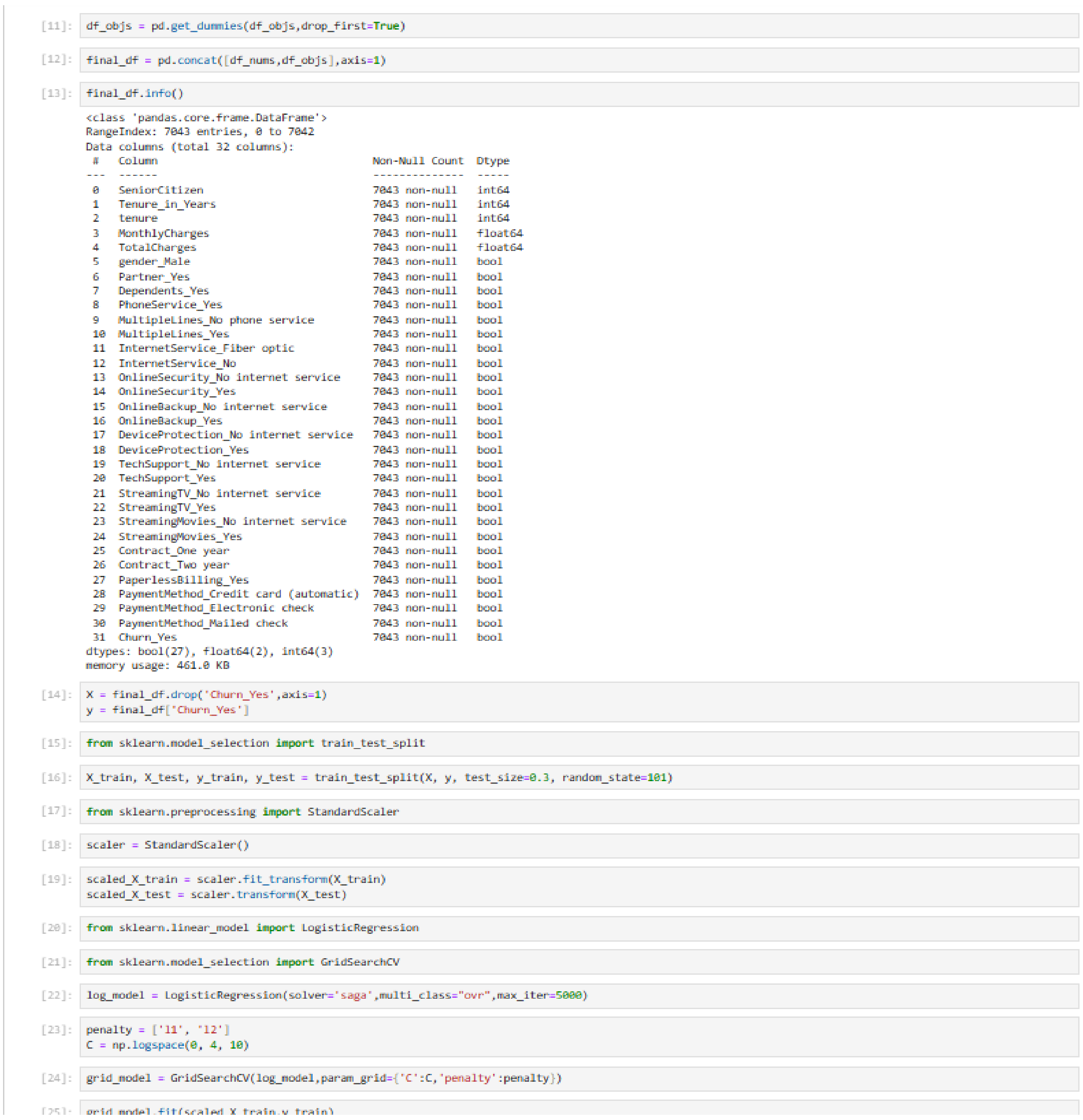
	SeniorCitizen	Tenure_in_Years	tenure	MonthlyCharges	TotalCharges
0	0	1	1	29.85	29.85
1	0	3	34	56.95	1889.50
2	0	1	2	53.85	108.15
3	0	4	45	42.30	1840.75
4	0	1	2	70.70	151.65
...	...	...	...	...	...
7038	0	2	24	84.80	1990.50
7039	0	6	72	103.20	7362.90
7040	0	1	11	29.60	346.45
7041	1	1	4	74.40	306.60
7042	0	6	66	105.65	6844.50

7043 rows × 5 columns

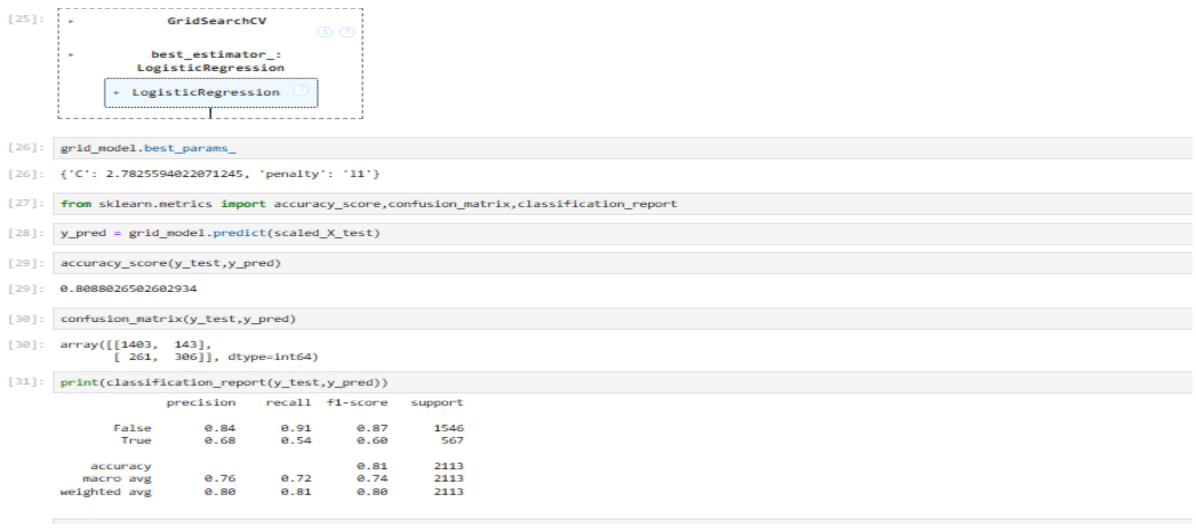
```
[10]: df_objs
```

[10]:

	gender	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	Streami
0	Female	Yes	No	No	No phone service	DSL	No	Yes	No	No	No	
1	Male	No	No	Yes	No	DSL	Yes	No	Yes	No	No	
2	Male	No	No	Yes	No	DSL	Yes	Yes	No	No	No	
3	Male	No	No	No	No phone service	DSL	Yes	No	Yes	Yes	No	
4	Female	No	No	Yes	No	Fiber optic	No	No	No	No	No	
...	...	...	...	...	...	...	...	...	...	...	...	
7038	Male	Yes	Yes	Yes	Yes	DSL	Yes	No	Yes	Yes	Yes	
7039	Female	Yes	Yes	Yes	Yes	Fiber optic	No	Yes	Yes	No	Yes	
7040	Female	Yes	Yes	No	No phone service	DSL	Yes	No	No	No	No	
7041	Male	Yes	No	Yes	Yes	Fiber optic	No	No	No	No	No	
7042	Male	No	No	Yes	No	Fiber optic	Yes	No	Yes	Yes	Yes	



```
[11]: df_objs = pd.get_dummies(df_objs,drop_first=True)

[12]: final_df = pd.concat([df_nums,df_objs],axis=1)

[13]: final_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 32 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   SeniorCitizen                          7043 non-null   int64
 1   Tenure_in_Years                        7043 non-null   int64
 2   tenure                                 7043 non-null   int64
 3   MonthlyCharges                         7043 non-null   float64
 4   TotalCharges                           7043 non-null   float64
 5   gender_Male                            7043 non-null   bool
 6   Partner_Yes                            7043 non-null   bool
 7   Dependents_Yes                         7043 non-null   bool
 8   PhoneService_Yes                       7043 non-null   bool
 9   MultipleLines_No phone service         7043 non-null   bool
 10  MultipleLines_Yes                      7043 non-null   bool
 11  InternetService_Fiber optic            7043 non-null   bool
 12  InternetService_No                     7043 non-null   bool
 13  OnlineSecurity_No internet service     7043 non-null   bool
 14  OnlineSecurity_Yes                     7043 non-null   bool
 15  OnlineBackup_No internet service       7043 non-null   bool
 16  OnlineBackup_Yes                       7043 non-null   bool
 17  DeviceProtection_No internet service   7043 non-null   bool
 18  DeviceProtection_Yes                   7043 non-null   bool
 19  TechSupport_No internet service        7043 non-null   bool
 20  TechSupport_Yes                        7043 non-null   bool
 21  StreamingTV_No internet service        7043 non-null   bool
 22  StreamingTV_Yes                        7043 non-null   bool
 23  StreamingMovies_No internet service    7043 non-null   bool
 24  StreamingMovies_Yes                    7043 non-null   bool
 25  Contract_One year                      7043 non-null   bool
 26  Contract_Two year                      7043 non-null   bool
 27  PaperlessBilling_Yes                   7043 non-null   bool
 28  PaymentMethod_Credit card (automatic)  7043 non-null   bool
 29  PaymentMethod_Electronic check         7043 non-null   bool
 30  PaymentMethod_Mailed check             7043 non-null   bool
 31  Churn_Yes                              7043 non-null   bool
dtypes: bool(27), float64(2), int64(3)
memory usage: 461.0 KB

[14]: X = final_df.drop('Churn_Yes',axis=1)
      y = final_df['Churn_Yes']

[15]: from sklearn.model_selection import train_test_split

[16]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)

[17]: from sklearn.preprocessing import StandardScaler

[18]: scaler = StandardScaler()

[19]: scaled_X_train = scaler.fit_transform(X_train)
      scaled_X_test = scaler.transform(X_test)

[20]: from sklearn.linear_model import LogisticRegression

[21]: from sklearn.model_selection import GridSearchCV

[22]: log_model = LogisticRegression(solver='saga',multi_class="ovr",max_iter=5000)

[23]: penalty = ['l1', 'l2']
      C = np.logspace(0, 4, 10)

[24]: grid_model = GridSearchCV(log_model,param_grid={'C':C,'penalty':penalty})

[25]: grid_model.fit(scaled_X_train,y_train)
```



```
[25]: ▸ GridSearchCV
      ▸ best_estimator_:
        LogisticRegression
        ▸ LogisticRegression

[26]: grid_model.best_params_

[26]: {'C': 2.7825594022071245, 'penalty': 'l1'}

[27]: from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

[28]: y_pred = grid_model.predict(scaled_X_test)

[29]: accuracy_score(y_test,y_pred)

[29]: 0.8088026502602934

[30]: confusion_matrix(y_test,y_pred)

[30]: array([[1403,  143],
             [ 261,  306]], dtype=int64)

[31]: print(classification_report(y_test,y_pred))
              precision    recall  f1-score   support

       False       0.84      0.91      0.87      1546
        True       0.68      0.54      0.60       567

    accuracy                           0.81      2113
   macro avg       0.76      0.72      0.74      2113
weighted avg       0.80      0.81      0.80      2113
```

References:

1. Adnan Idris, Muhammad Rizwan, Asifullah Khan,"Customer Churn Prediction for Telecommunication: Employing various various features selection techniques and tree based ensemble classifiers"

2. Sumana Sharma Poudel, Suresh Pokharel, Mohan Timilsina,"Explaining customer churn prediction in telecom industry using tabular machine learning models"

3. Sharmila K. Wagh, Aishwarya A. Andhale, Kishor S. Wagh, Jayshree R. Pansare, Sarita P. Ambadekar, S.H. Gawande,"Customer churn prediction in telecom sector using machine learning techniques"

4. "An Introduction to Statistical Learning with Applications in R" Book By Gareth James Daniela Witten Trevor Hastie Robert Tibshirani with Applications in R

5. A Survey on Customer Churn Prediction using Machine Learning Techniques by Saran Kumar A. & Chandrakala D., PhD

6. A Comparison of Machine Learning Techniques for Customer Churn Prediction by Thanasis Vafeiadis , Kostas Diamantaras, Konstantinos Ch. Chatzisavvas, G. Sarigiannidis

7. A Hybrid Machine Learning Model for Predicting Customer Churn in the Telecommunication Industry by Modupe Odusami, Olusola Oluwakemi Abayomi-Alli, Sanjay Misra, Abayomi-Alli Adebayo

https://scikit-learn.org/stable/modules/tree.html#classification

https://www.kaggle.com/datasets/blastchar/telco-customer-churn

Glossary:

LIST OF ABBREVATIONS

SVM >>	Support Vector Machine
ML >>	Machine Learning
SMOTE>>	Synthetic Minority Over Sampling Technique
PCA>>	Principle Component Analysis