Telco Customer Churn Project

- **Introduction**

This documentation provides an overview and explanation of the code used for customer churn prediction. Customer churn refers to the phenomenon where customers stop doing business with a company or cancel their subscriptions. The code demonstrates various steps involved in the data analysis, preprocessing, feature engineering, and building predictive models using machine learning algorithms.

- **Step 1: Importing Required Libraries**

This report begins with importing necessary libraries and modules. These libraries include:

- ✓ pandas: For data manipulation and analysis
- ✓ numpy: For mathematical operations on arrays
- ✓ seaborn and matplotlib.pyplot: For data visualization
- ✓ sklearn: For machine learning algorithms and evaluation metrics
- ✓ imblearn: For handling imbalanced datasets (SMOTE)

```
[315] from sklearn.model_selection import train_test_split
      from sklearn.model_selection import cross_val_score
      from sklearn.linear_model import LogisticRegression
      from sklearn.tree import DecisionTreeClassifier
      from sklearn.ensemble import RandomForestClassifier
      from sklearn.model_selection import GridSearchCV
      from sklearn.svm import SVC
      from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix, classification_report
      import pandas as pd
      import numpy as np
      import seaborn as sns
      import matplotlib.pyplot as plt
      from imblearn.over_sampling import SMOTE
```

- **Step 2: Load the Dataset**

We load the customer churn dataset from a CSV file using the pandas library. The dataset contains information about customers, their attributes, and churn status.

```
df = pd.read_csv('/content/WA_Fn-UseC_-Telco-Customer-Churn.csv')
df.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No	No	One year	No	Mailed check	56.95	1889.5	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes

5 rows × 21 columns

3.1. Dimension and Information:

We examine the dimension and structure of the dataset using the `shape` and `info` functions.

```
3]  df.shape

    (7043, 21)
```

nfo

```
4]  df.info()

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

3.2. Descriptive Statistics

We calculate descriptive statistics of the dataset using the `describe` function, providing insights into the distribution and summary of numerical features.

```
df.describe()
```

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

3.3. Checking for Null Values

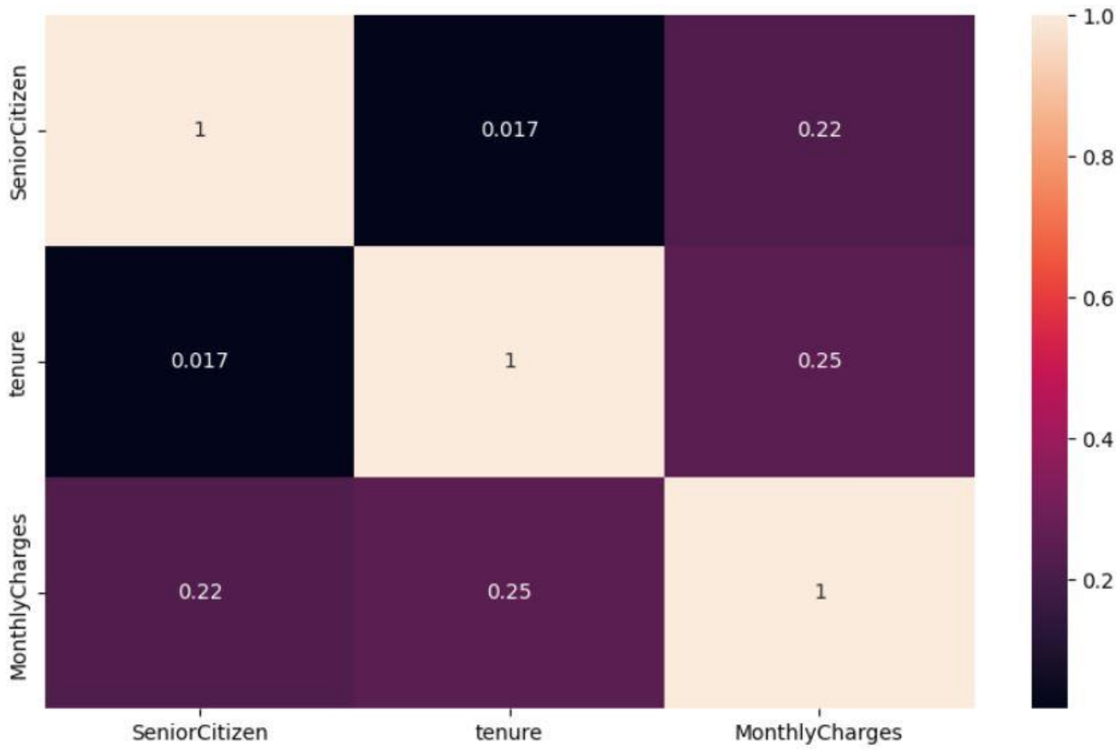
We check for missing values in the dataset using the `isnull().sum()` function, which returns the count of null values in each column.

```
df.isnull().sum()
```

```
customerID          0
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
```

3.4. Heatmap

A heatmap is created using seaborn to visualize the correlation between different features in the dataset. This helps identify relationships and potential multicollinearity.



3.5. Checking Dataset Balancing

We can check the class distribution of the target variable ('Churn') to assess if the dataset is imbalanced or not. It prints the counts of each class which shows dataset is imbalanced so we will use SMOTE onwards.

```
[276] print(df['Churn'].value_counts())

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

- ## Step 4: Data Pre-processing:

4.1. Dropping Irrelevant Columns

We remove the 'customerID' column from the dataset as it is not relevant for churn prediction.

4.2. Checking Feature Uniqueness

We print the count of unique values for each feature in the dataset to determine if any feature has constant values.

```
gender               2
SeniorCitizen        2
Partner              2
Dependents           2
tenure              73
PhoneService         2
MultipleLines        3
InternetService      3
OnlineSecurity       3
OnlineBackup         3
DeviceProtection     3
TechSupport          3
StreamingTV          3
StreamingMovies      3
Contract             3
PaperlessBilling     2
PaymentMethod        4
MonthlyCharges    1585
TotalCharges      6531
Churn                2
dtype: int64
```

- ## Step 5: Exploratory Data Analysis (EDA)

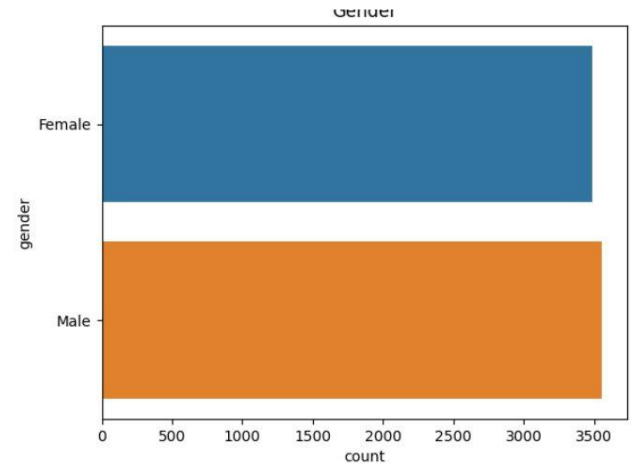
We perform univariate and bivariate analysis to explore relationships between variables and their impact on churn.

5.1. Univariate Analysis

We do univariate analysis which means we visualize every single feature using count plots and pie charts to visualize the distribution of categorical features such as 'gender', 'PhoneService', 'MultipleLines', 'Contract', 'PaymentMethod', and the target variable 'Churn'.

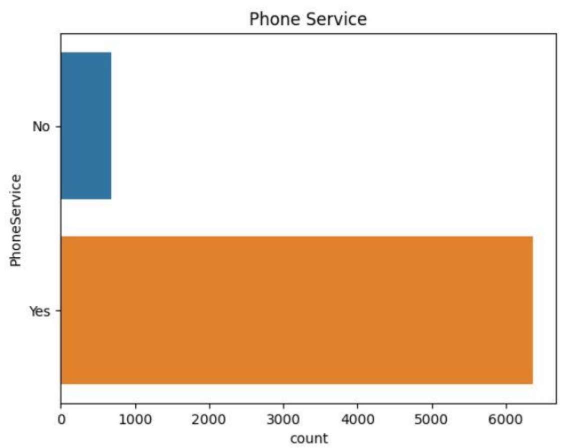
✓ **Gender:**

This shows whether the customer is a male or a female.



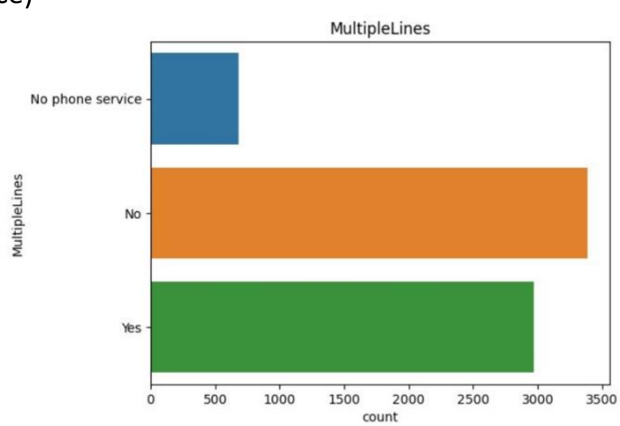
- ✓ **Phone Service:**

  This feature shows whether the customer has a phone service or not (Yes, No)



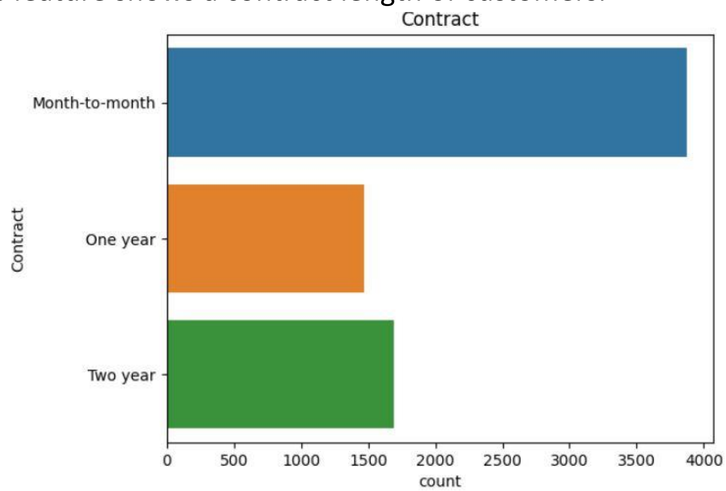
- ✓ **Multiple Lines:**

  This feature shows Whether the customer has multiple lines or not (Yes, No, No phone service)



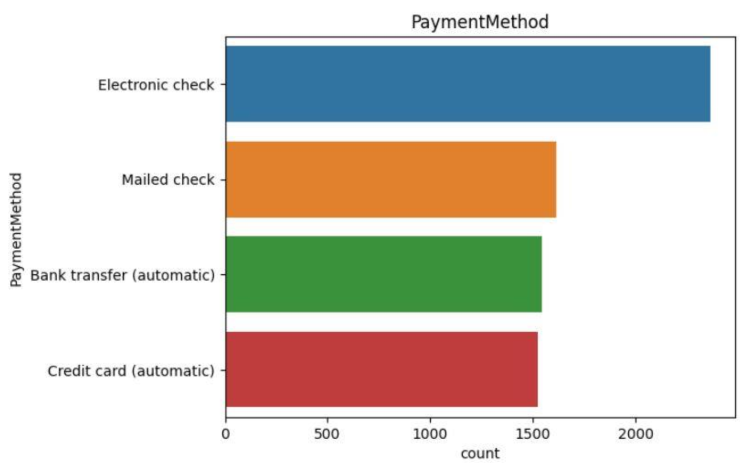
- **Contract:**

This feature shows a contract length of customers.



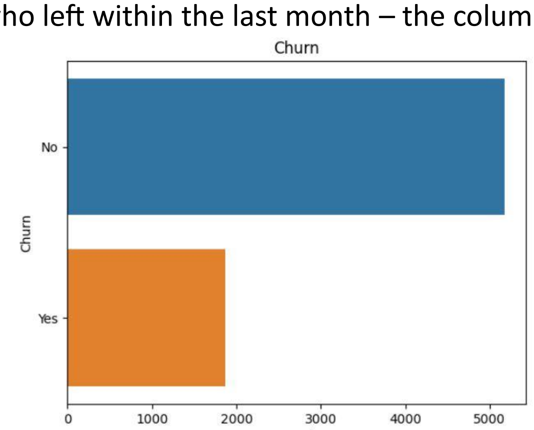
- **Payment method:**

This feature shows multiple payment methods



- **Churn:**

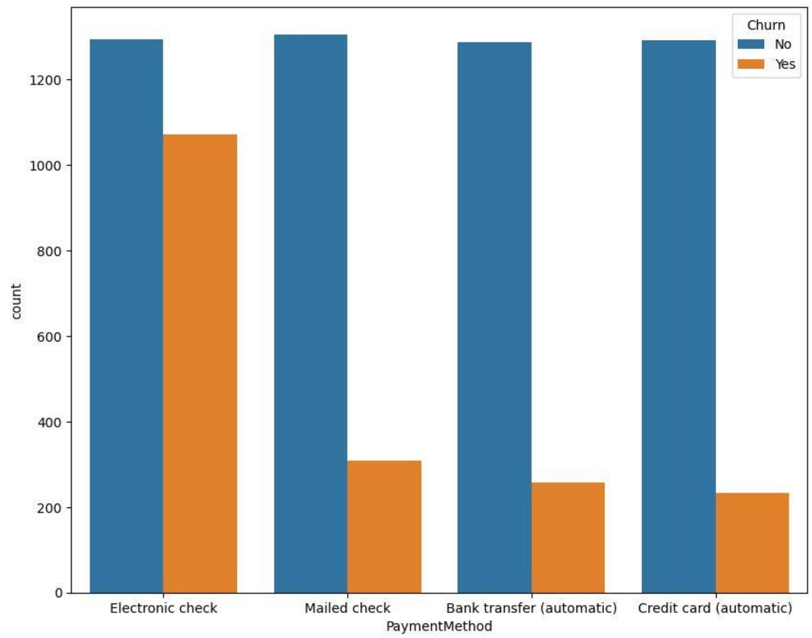
Customers who left within the last month – the column is called Churn



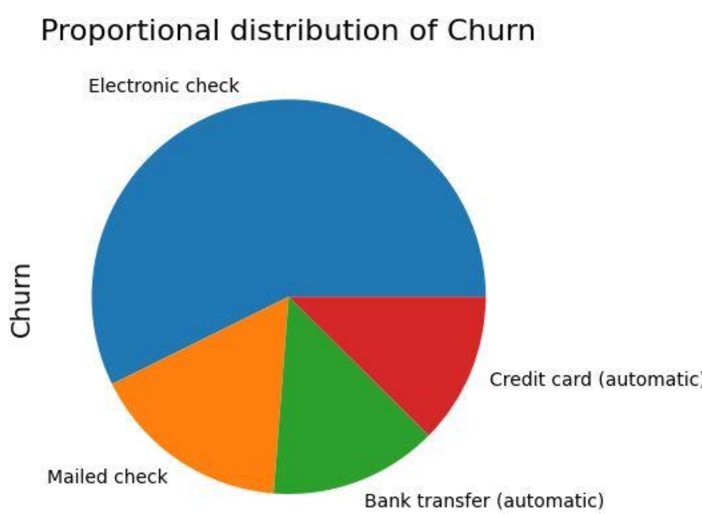
5.2. Bivariate Analysis

We do bivariate analysis using count plots to show the relationship between 'Payment Method' and 'Churn'. It also creates pie charts to display the proportional distribution of churn based on payment method for both churned and non-churned customers.

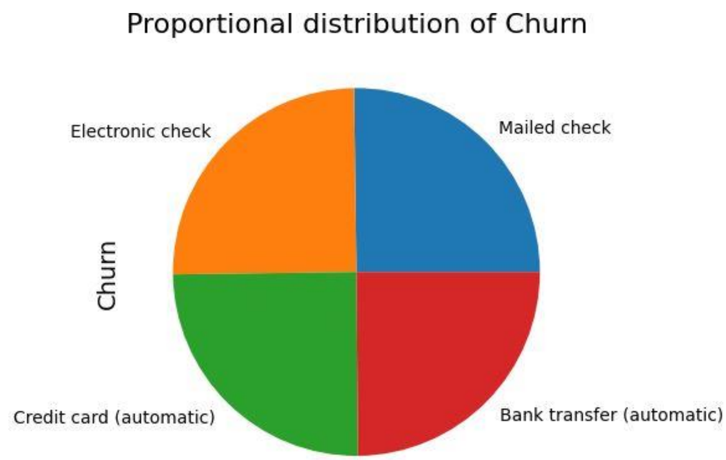
- **Relationship between the Payment Method and Churn:**



- **Proportional distribution of churned customers based on payment method:**



- **Proportional distribution of non-churned customers based on payment method:**



- ## Step 6: Feature Engineering

6.1. Handling Zero Tenure Values

We can identify rows where tenure is zero and replaces those values with the mean tenure value, assuming zero tenure is missing or invalid data.

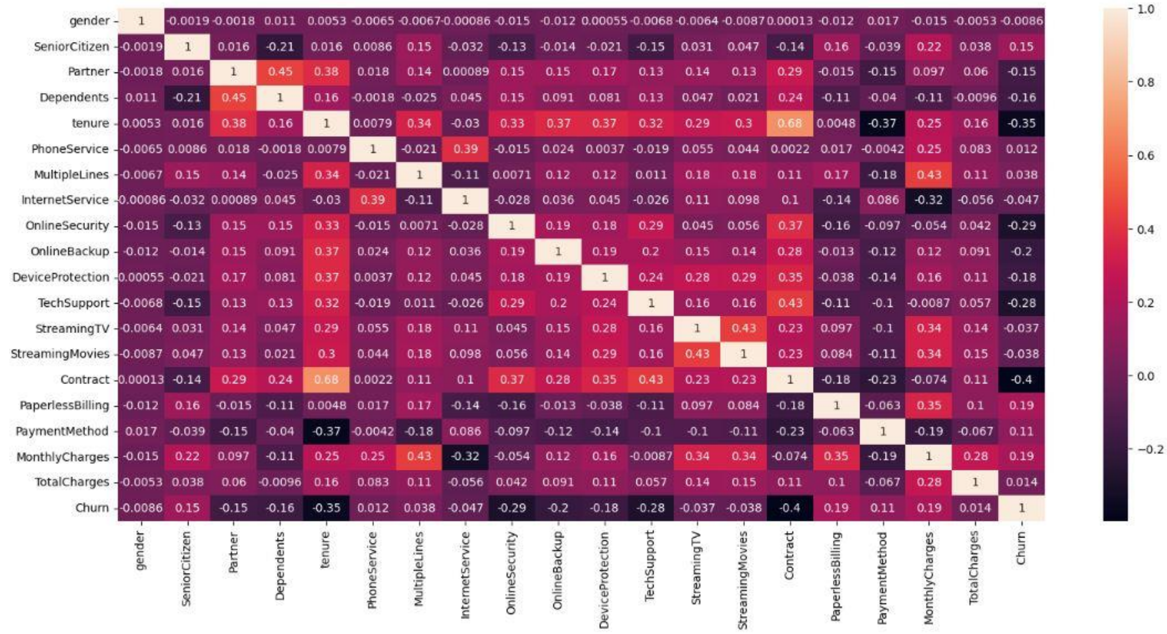
6.2. Label Encoding

We apply label encoding to convert categorical features into numerical representations using the Label Encoder from sklearn.preprocessing. This allows the machine learning algorithms to work with categorical data.

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	0	0	1	0	1.0	0	1	0	0	2	0	0	0	0	0	1	2	29.85	2505	0
1	1	0	0	0	34.0	1	0	0	2	0	2	0	0	0	1	0	3	56.95	1466	0
2	1	0	0	0	2.0	1	0	0	2	2	0	0	0	0	0	1	3	53.85	157	1
3	1	0	0	0	45.0	0	1	0	2	0	2	2	0	0	1	0	0	42.30	1400	0
4	0	0	0	0	2.0	1	0	1	0	0	0	0	0	0	0	1	2	70.70	925	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7038	1	0	1	1	24.0	1	2	0	2	0	2	2	2	2	1	1	3	84.80	1597	0
7039	0	0	1	1	72.0	1	2	1	0	2	2	0	2	2	1	1	1	103.20	5698	0
7040	0	0	1	1	11.0	0	1	0	2	0	0	0	0	0	0	1	2	29.60	2994	0
7041	1	1	1	0	4.0	1	2	1	0	0	0	0	0	0	0	1	3	74.40	2660	1
7042	1	0	0	0	66.0	1	0	1	2	0	2	2	2	2	2	1	0	105.65	5407	0

- **Step 7: Heatmap (After Feature Engineering)**

We create another heatmap is created to visualize the correlation between features after feature engineering and label encoding**.**



- **Step 8: Data Resampling with SMOTE:**

We use Synthetic Minority Oversampling Technique (SMOTE) from imblearn to handle the imbalanced class distribution in the dataset. It oversamples the minority class (churned customers) to balance the dataset. This shows we balanced the dataset 50% each.

```
% of each class in the dataset
0    0.5
1    0.5
Name: Churn, dtype: float64
```

- **Step 9: Train-Test Split**

The code splits the dataset into training and testing sets using the train_test_split function from sklearn.model_selection.

- ## Step 10: Model Building and Evaluation

We train and evaluate three classification models:

10.1. Logistic Regression:

Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set. A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. It gives accuracy around 80%.

```
lr = LogisticRegression()
lr.fit(X_train,y_train)
y_pred = lr.predict(X_test)
lr_acc = lr.score(X_test,y_test)
print("Accuracy: ",lr_acc)
```

```
Accuracy:  0.8001215066828675
```

10.2. Random Forest

Random forest is an ensemble machine learning algorithm. It is perhaps the most popular and widely used machine learning algorithm given its good or excellent performance across a wide range of classification and regression predictive modeling problems. It works well on this model and gives around 85% accuracy.

```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
rf_acc = rf.score(X_test, y_test)
print("Accuracy:", rf_acc)
```

```
Accuracy: 0.8535844471445929
```

10.3. Support Vector Machine (SVM)

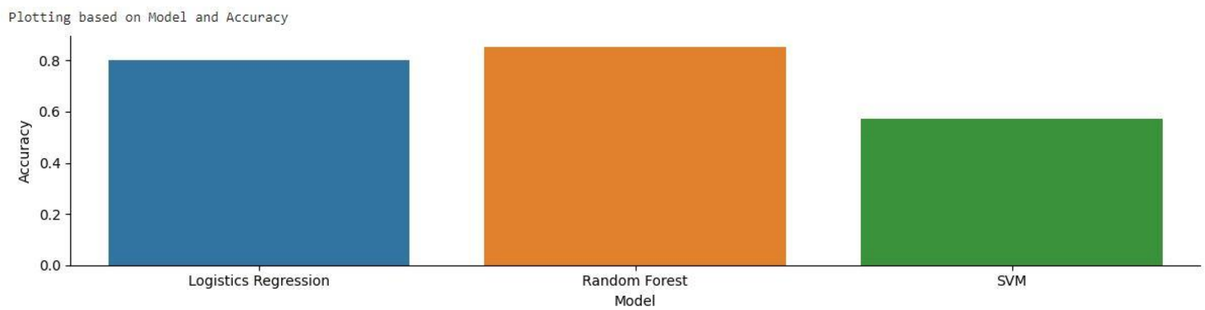
SVM (Support Vector Machines) is a machine learning algorithm used for classification and regression tasks. It finds an optimal hyperplane to separate data points of different classes, maximizing the margin between them. SVMs are effective for handling complex datasets, can handle non-linear data through the kernel trick, and are memory-efficient by relying on a subset of support vectors but in this problem SVM not doing well and gives around 57% accuracy.

```
svm = SVC()
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
svm_acc = svm.score(X_test, y_test)
print("Accuracy:", svm_acc)
```

```
Accuracy: 0.571688942891859
```

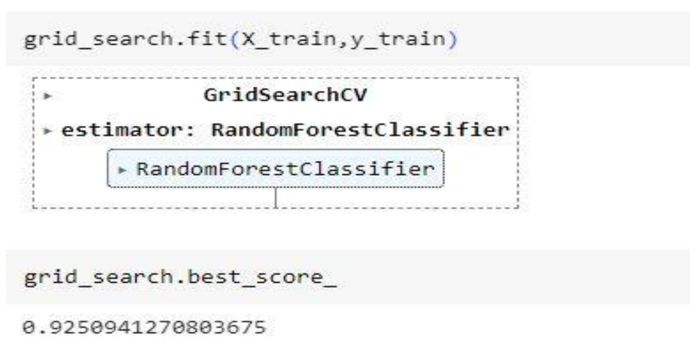
- **Step 11: Final Predictions and Model Comparison**

We compare the accuracy of different models and creates a data frame with the model names and their respective accuracies.



- **Step 12: Model Hyperparameter Tuning**

GridSearchCV is a technique for finding the optimal parameter values from a given set of parameters in a grid. It's essentially a cross-validation technique. The model as well as the parameters must be entered. After extracting the best parameter values, predictions are made. GridSearchCV from sklearn.model_selection is used to perform hyperparameter tuning for the Random Forest model. We give 300 n_estimators and max depth of 20 for hyper parameter tunning which gives around 92% accuracy.



- **Step 13: Final Model Evaluation**

The code evaluates the final Random Forest model with the best hyperparameters using the training and testing sets.

- **Confusion matrix:**

The confusion matrix helps assess the performance of a classification model by comparing the actual and predicted class labels. It provides insights into the model's ability to correctly classify instances as either negative or positive. In this case, the model accurately predicted 711 negative instances (true negatives) and 697 positive instances (true positives). However, it misclassified 141 negative instances as positive (false positives) and 97 positive instances as negative (false negatives).

	Predicted Negative	Predicted Positive
Actual Negative	711	141
Actual Positive	97	697

- **Accuracy:**

Accuracy is the ratio of number of correct predictions to the total number of input samples. We got around 85% accuracy in a model.

```
# Accuracy score on the test set.
print('Accuracy score for test data is:', accuracy_score(y_test, y_test_pred))
```

```
Accuracy score for test data is: 0.8554070473876063
```

- **Precision:**

Precision is defined as the ratio of correctly classified positive samples (True Positive) to a total number of classified positive samples (either correctly or incorrectly).

```
# Precision score on the training set
print('Accuracy score for train data is:', precision_score(y_train, y_train_pred))
```

```
Accuracy score for train data is: 0.9943045563549161
```

```
 #Precision score on the test set.
print('Accuracy score for test data is:', precision_score(y_test, y_test_pred))
```

```
Accuracy score for test data is: 0.8317422434367542
```

- **Recall:**

The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples. The recall measures the model's ability to detect positive samples. The higher the recall, the more positive samples detected.

ℓall

```
# recall score on the training set
print('Accuracy score for train data is:', recall_score(y_train, y_train_pred))
```

```
Accuracy score for train data is: 0.9993974088580898
```

```
# recall score on the test set.
print('Accuracy score for test data is:', recall_score(y_test, y_test_pred))
```

```
Accuracy score for test data is: 0.8778337531486146
```

- **F1-Score:**

F1 score is a machine learning evaluation metric that measures a model's accuracy. It combines the precision and recall scores of a model. The accuracy metric computes how many times a model made a correct prediction across the entire dataset.

```
Classification_report:
              precision    recall  f1-score   support

           0       0.88      0.83      0.86       852
           1       0.83      0.88      0.85       794

    accuracy                           0.86      1646
   macro avg       0.86      0.86      0.86      1646
weighted avg       0.86      0.86      0.86      1646
```

- ## Conclusion

This documentation provides a step-by-step explanation of the code for customer churn prediction. It covers data loading, exploratory data analysis, data preprocessing, feature engineering, model training, evaluation, and hyperparameter tuning. The code helps in understanding and predicting customer churn, which can be valuable for businesses to take proactive measures to retain customers.