FDM Mini Project - 2023

Data Detectives

Customer Churn Prediction System

Meet

THE DATA DETECTIVES

IT21176456
Gimmana M.R.M

IT21261732
Perera W.A.S.K

IT21262104
Lakshani D.M.W.S

IT21260988
Randeniya R. A. D. S. E

BACKGROUND

- Customer churn is a crucial business metric.
- By generating predictive models analyzing historical data and customer behavior, customer churn can be predicted.
- It enables businesses to take proactive measures to retain customers.
- Proactive customer retention efforts can lead to improved customer satisfaction, reduced customer turnover, and optimized marketing and customer service efforts, ultimately contributing to business growth and profitability.

Target

To create a precise predictive analytics model to predict customer churn by analyzing various customer attributes.

Business Goal

To reduce the customer churn within the telecommunication industry effectively.

By using the predicted information, the company can implement strategies to retain the at-risk customers and enhance overall customer satisfaction.

TECHNOLOGIES

Data Preprocessing

Handling Null Values

Find Null Values

```
data.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        11
Churn                0
dtype: int64
```

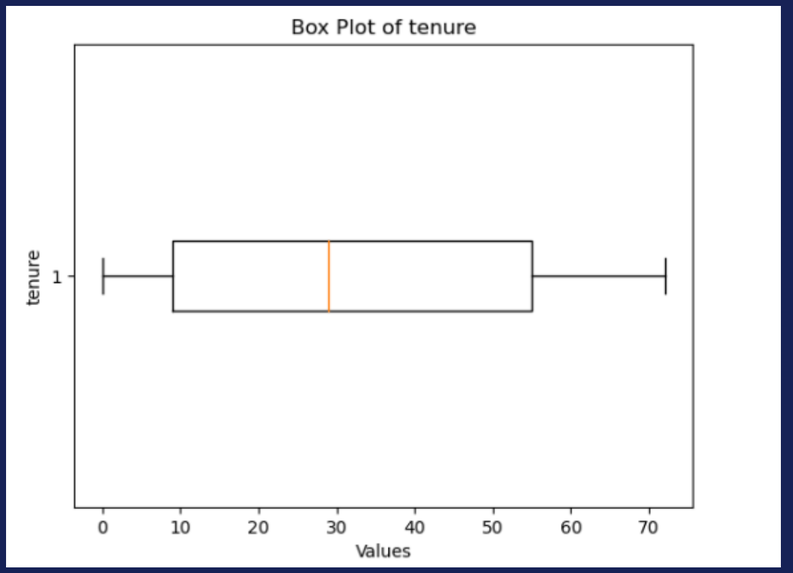
Fill the Null values with median

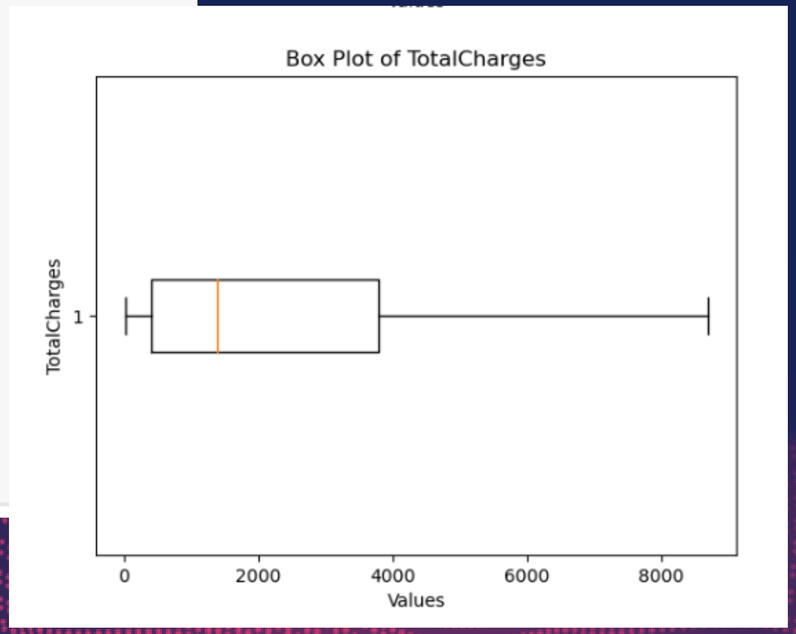
```
# Fill missing values with the median
median_value = data["TotalCharges"].median()
data["TotalCharges"].fillna(median_value,inplace=True)

# Verify the updated column
print(data["TotalCharges"])
```

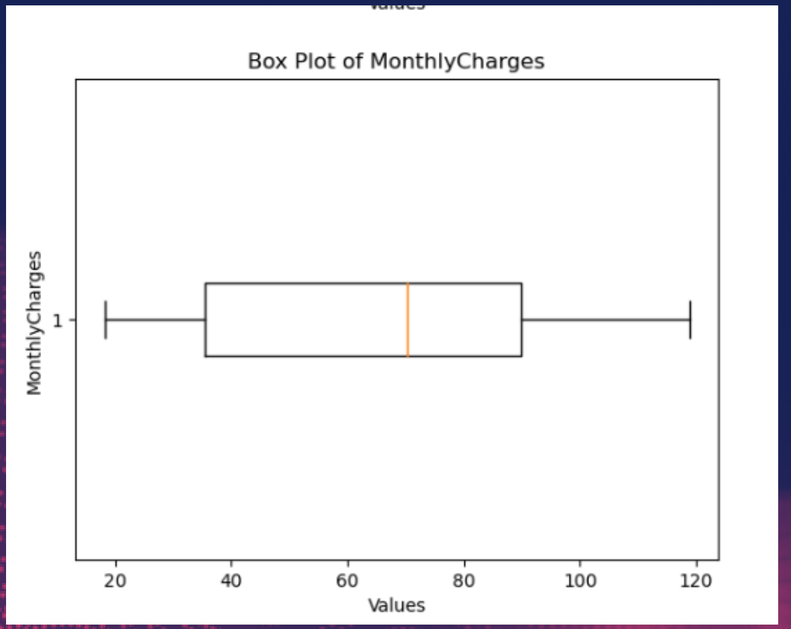
```
0         29.85
1       1889.50
2        108.15
3       1840.75
4        151.65
         ...
7038    1990.50
7039    7362.90
7040     346.45
7041     306.60
7042    6844.50
Name: TotalCharges, Length: 7043, dtype: float64
```

Handling Outliers

```
: import matplotlib.pyplot as plt

# Assuming you have a 'data' variable containing the necessary data

# Create a list of data to plot
data_to_plot = [data['tenure'], data['MonthlyCharges'], data['TotalCharges']]

# Create a list of labels for the box plots
labels = ['tenure', 'MonthlyCharges', 'TotalCharges']

# Create separate figures and axes for each box plot
for i in range(len(data_to_plot)):
    fig, ax = plt.subplots()

    # Plot the box plot for the current variable
    ax.boxplot(data_to_plot[i], vert=False)

    # Set the title and axes labels for the current plot
    ax.set_title(f'Box Plot of {labels[i]}')
    ax.set_xlabel('Values')
    ax.set_ylabel(labels[i])

    # Display the plot
    plt.show()
```







Data Transform

Simplifying the dataset by binning on numerical variables (tenure, MonthlyCharges, and TotalCharges) and transforms them into categorical variables with three levels: 'low', 'medium', and 'high'.

```
: #Create an empty dataframe
bin_df = pd.DataFrame()

#Update the binning dataframe
bin_df['tenure_bins'] =  pd.qcut(data['tenure'], q=3, labels= ['low', 'medium', 'high'])
bin_df['MonthlyCharges_bins'] =  pd.qcut(data['MonthlyCharges'], q=3, labels= ['low', 'medium', 'high'])
bin_df['TotalCharges_bins'] =  pd.qcut(data['TotalCharges'], q=3, labels= ['low', 'medium', 'high'])
bin_df['Churn'] = data['Churn']

#Plot the bar chart of the binned variables
bar('tenure_bins', bin_df)
bar('MonthlyCharges_bins', bin_df)
bar('TotalCharges_bins', bin_df)
```

Encoding Categorical Features

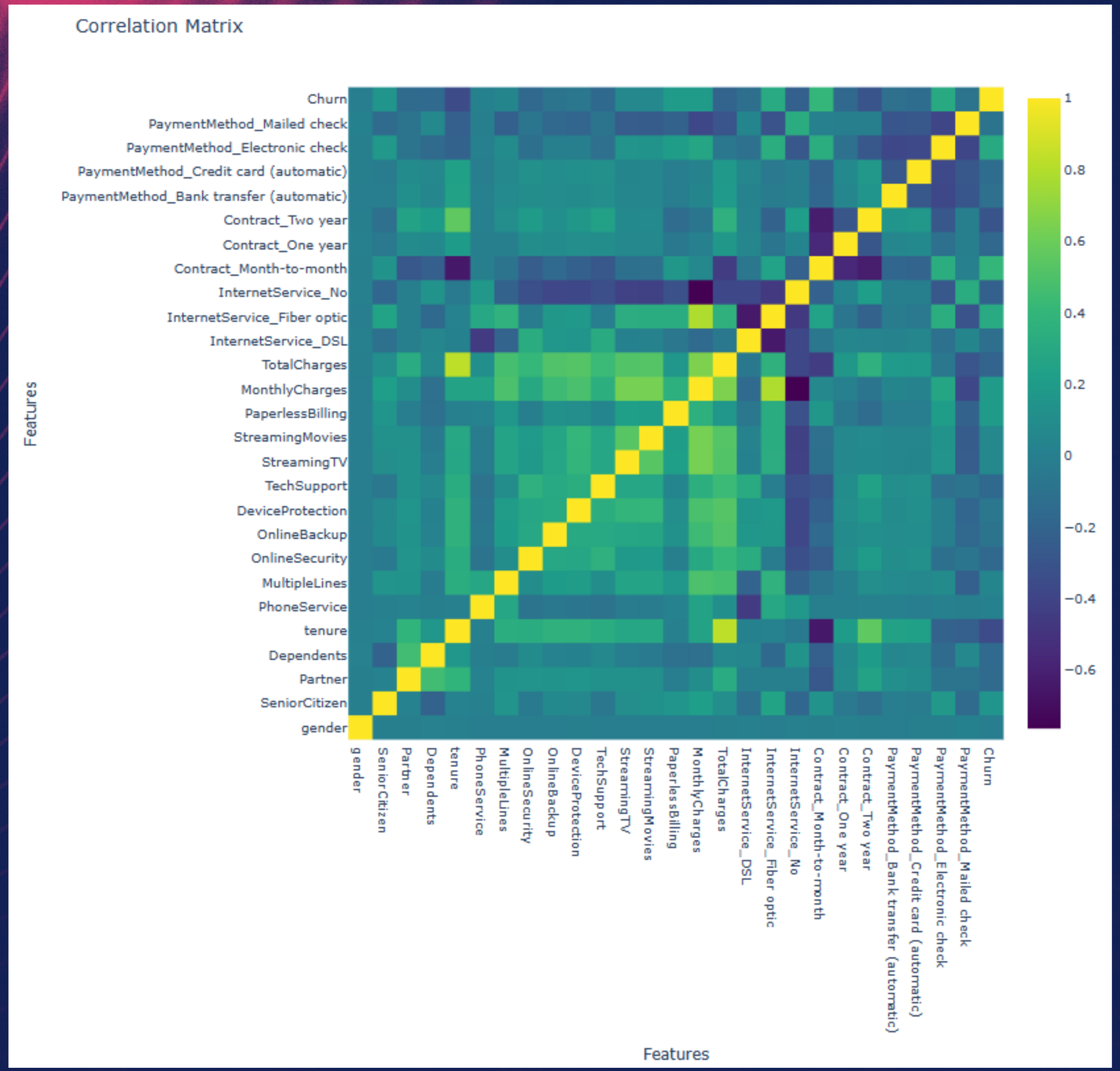
```
In [21]: print(data.dtypes)
```

```
customerID                                object
gender                                     int64
SeniorCitizen                              int64
Partner                                    int64
Dependents                                 int64
tenure                                     int64
PhoneService                               int64
MultipleLines                              int64
OnlineSecurity                             int64
OnlineBackup                               int64
DeviceProtection                           int64
TechSupport                                int64
StreamingTV                                int64
StreamingMovies                            int64
PaperlessBilling                           int64
MonthlyCharges                           float64
TotalCharges                             float64
Churn                                      int64
InternetService_DSL                        uint8
InternetService_Fiber optic                uint8
InternetService_No                         uint8
Contract_Month-to-month                    uint8
Contract_One year                          uint8
Contract_Two year                          uint8
PaymentMethod_Bank transfer (automatic)    uint8
PaymentMethod_Credit card (automatic)      uint8
PaymentMethod_Electronic check             uint8
PaymentMethod_Mailed check                 uint8
dtype: object
```

```
: #Encording the Categorical Features

# yes and no replace with 1 and 0
yes_no_columns = ['Partner','Dependents','PhoneService','MultipleLines','OnlineSecurity','OnlineBackup','DeviceProtection',
                  'TechSupport','StreamingTV','StreamingMovies','PaperlessBilling','Churn']
for col in yes_no_columns:
  data[col].replace({'Yes':1,'No': 0},inplace=True)

#one hot encoding to other columns
data= pd.get_dummies(data=data,columns=['InternetService','Contract','PaymentMethod'])
data.columns

#geneder column
data['gender'].replace({'Female':0,'Male':1},inplace=True)
```

```
: # Save the updated DataFrame to the CSV file
data.to_csv('E:/sliit/Year 3/SEM 1/FDM/Project/Customer-Churn-Prediction-System/Dataset/Telco-Customer-Churn.csv', index=False)
```

Correlation Matrix



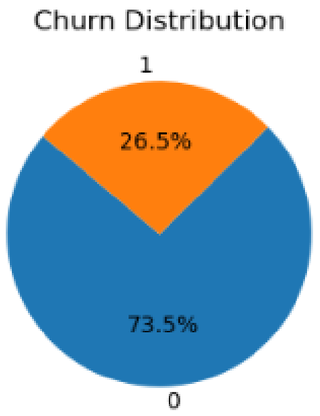
Balance the Dataset

```
#balancing the dataset using SMOTE oversampling method

X= data.drop('Churn',axis='columns')
y=data['Churn']
y.value_counts()

smote = SMOTE(sampling_strategy='minority')
X_sm, y_sm = smote.fit_resample(X,y)

y_sm.value_counts()
```

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

```
churn_counts = data['Churn'].value_counts()
print(churn_counts)

plt.figure(figsize=(3, 3))
plt.pie(churn_counts, labels=churn_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Churn Distribution')
plt.show()
```

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



Data set is unbalanced

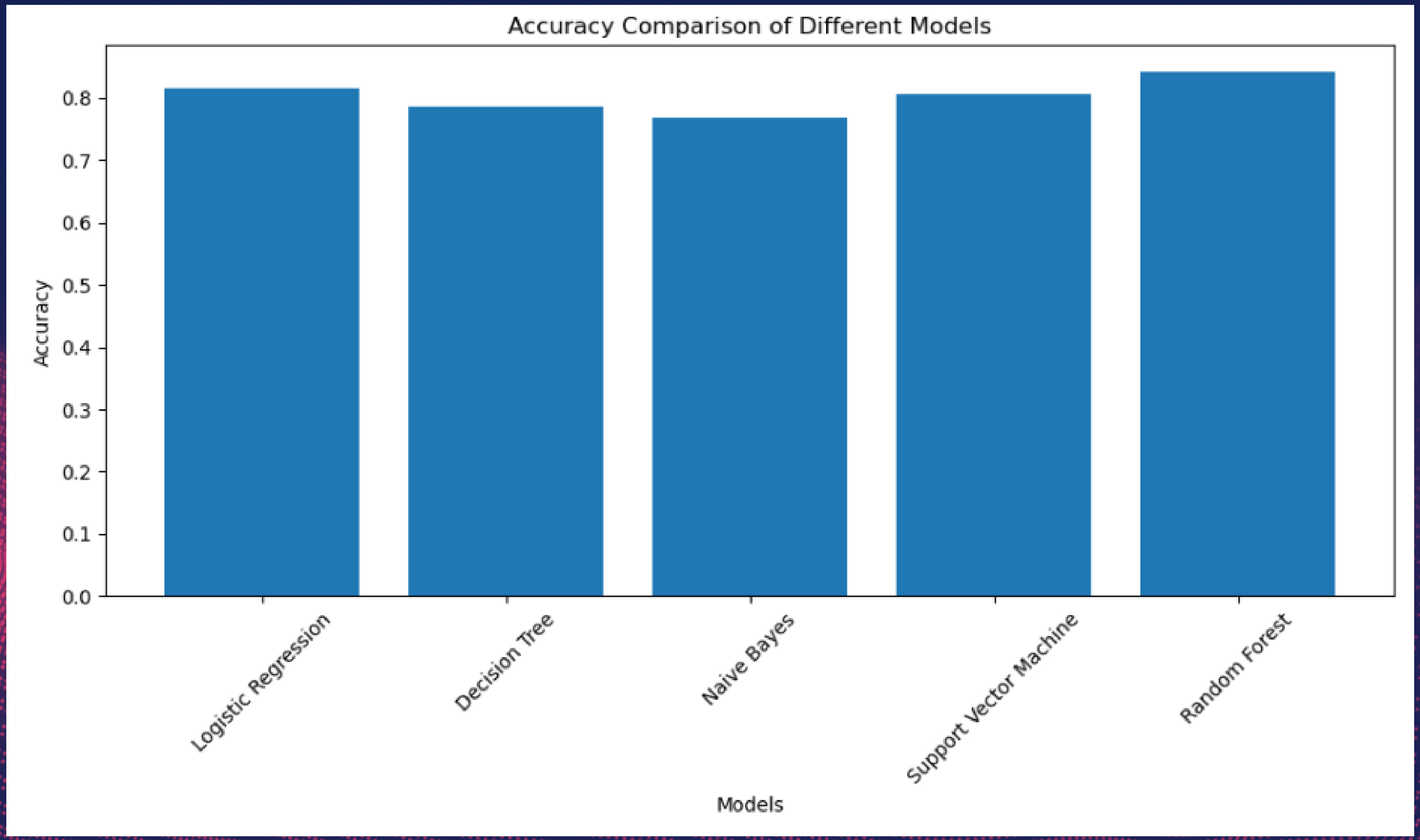
Model Implementation

Implementation

1) We used **five different models** here to predict the customer churn or not.

- Linear Regression
- Random Forest
- Decision Tree Classification
- Naïve Bayes Classification
- Support Vector Classification

2) The **Random Forest Model** was selected as the most effective in terms of predictive performance.


Accuracy Comparison of Different Models
Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Logistic Regression
Decision Tree
Naive Bayes
Support Vector Machine
Random Forest
Models

Improving the Accuracy

Improve best model by hyperparameter tuning

```
# Define the Random Forest model
rf_model2 = RandomForestClassifier()

# Define the parameter grid
param_grid = {
    'n_estimators': [100, 300, 500],
    'max_features': ['sqrt', 'log2'],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'bootstrap': [True, False]
}

# Perform grid search with cross-validation
grid_search = GridSearchCV(rf_model2, param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Get the best model and its predictions
best_rf_model = grid_search.best_estimator_
rf_y_pred = best_rf_model.predict(X_test)

# Evaluate the model
print("Random Forest (after parameter tuning):")
print(classification_report(y_test, rf_y_pred))
```

```
Random Forest (after parameter tuning):
              precision    recall  f1-score   support

           0       0.85      0.82      0.84      1021
           1       0.83      0.86      0.85      1049

    accuracy                           0.84      2070
   macro avg       0.84      0.84      0.84      2070
weighted avg       0.84      0.84      0.84      2070
```

Final Product

Customer Churn Prediction System

Challenges

- Poor data quality can lead to inaccurate predictions.
- Churners are often a minority in the dataset, leading to class imbalance.
- Choosing the right machine learning model.

Solutions

- Data cleaning, preprocessing, and validation to ensure the data used for prediction is accurate and reliable.
- Use techniques like SMOTE to balance the classes in the training dataset.
- Experiment with different models (e.g., logistic regression, decision trees, random forests, neural networks) and choose the one that performs best.

further implementations and developing goals

- Aiming to modify the system to showcase it as a marketable product.
- Build a product specific to the Sri Lankan telecommunication industry.
- Building the system as suitable to the unique rapid fluctuation in the telecommunication industry.

THANK YOU!