CUSTOMER CHURN PREDICTION

Final submission
Introduction

Churn prediction analysis development is a crucial process in the field of business analytics that aims to forecast which customers are likely to cease their relationship with a company. By utilizing historical data and advanced machine learning techniques, businesses can identify patterns and factors that contribute to customer churn. This analysis enables companies to proactively devise strategies to retain valuable customers and minimize revenue loss.

IBM offers various solutions and services in the realm of customer churn prediction. Leveraging its advanced analytics and AI capabilities, IBM provides businesses with tools to analyze customer behavior, identify potential churn indicators, and develop proactive strategies to retain customers. IBM’s offerings often include predictive modeling, machine learning algorithms, and

data mining techniques that help businesses gain insights into customer churn patterns and make data-driven decisions to enhance customer retention strategies. IBM's customer churn prediction solutions are designed to optimize customer relationships and improve overall business performance.

OBJECTIVE

The primary objective of customer churn prediction is to forecast which customers are likely to discontinue their relationship with a company. By identifying potential churners beforehand, businesses can implement targeted retention strategies to reduce customer attrition, increase customer satisfaction, and ultimately improve long-term profitability. This proactive approach helps businesses to allocate resources effectively, enhance customer engagement, and maintain a loyal customer base.

DATASET

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

DESIGN THINKING

The design process for customer churn prediction analysis typically involves several key steps:

Data Collection: Gather relevant data, including customer demographics, transaction history, and customer interactions.

Data Preprocessing: Cleanse and preprocess the data to ensure accuracy and consistency, handling missing values and outliers appropriately.

Feature Selection: Identify relevant features that could impact customer churn, considering factors such as customer behavior, purchase patterns, and satisfaction metrics.

Model Selection: Choose appropriate machine learning models such as logistic regression, decision trees, random forests, or neural networks, based on the nature of the data and the business requirements.

Model Training: Train the selected model using historical data, and evaluate its performance using suitable metrics like accuracy, precision, recall, and F1-score.

Model Evaluation and Validation: Validate the model's performance on a separate dataset to assess its generalization capability and ensure its reliability.

Interpretation and Actionable Insights: Analyze the model's predictions to understand the key drivers of customer churn, and derive actionable insights to devise effective retention strategies.

Implementation and Monitoring: Implement the developed churn prediction model into the business process, and continuously monitor its performance, making necessary adjustments based on real-time data and feedback.

ANALYSIS APPROACH

The analysis approach for customer churn prediction typically involves the following steps:

Data Exploration: Conduct a thorough exploration of the dataset to understand the underlying patterns, trends, and potential factors contributing to customer churn.

Feature Engineering: Create new features or transform existing ones to enhance the predictive power of the model. This step involves extracting meaningful insights from raw data, such as creating customer engagement scores or churn risk indicators.

Model Development: Utilize various machine learning algorithms to build predictive models that can accurately forecast customer churn. Commonly used techniques include logistic regression, decision trees, random forests, support vector machines, and neural networks.

Model Evaluation: Assess the performance of the developed models using appropriate evaluation metrics such as accuracy, precision, recall, F1-score, and ROC curves. This step helps in determining the effectiveness of the model in predicting customer churn.

Interpretation of Results: Analyze the model outputs to interpret the key factors influencing customer churn. This step provides valuable insights for understanding customer behavior and identifying areas for potential improvement in customer retention strategies.

Actionable Recommendations: Based on the analysis results, formulate actionable recommendations and strategies to proactively manage customer churn. These recommendations could include targeted marketing campaigns, personalized customer incentives, or improved customer service initiatives to enhance overall customer satisfaction and loyalty.

In this phase, we will show you how to build a customer churn prediction model in Python using the random forests algorithm.

PRE-REQUISITES FOR BUILDING A CHURN PREDICTION MODEL

We will use the Telco Customer Churn dataset from Kaggle for this analysis. We also need a Python IDE to run the codes provided here. Finally, make sure to also have the following libraries installed – pandas, Matplotlib, Seaborn, Scikit-Learn, and Imblearn.

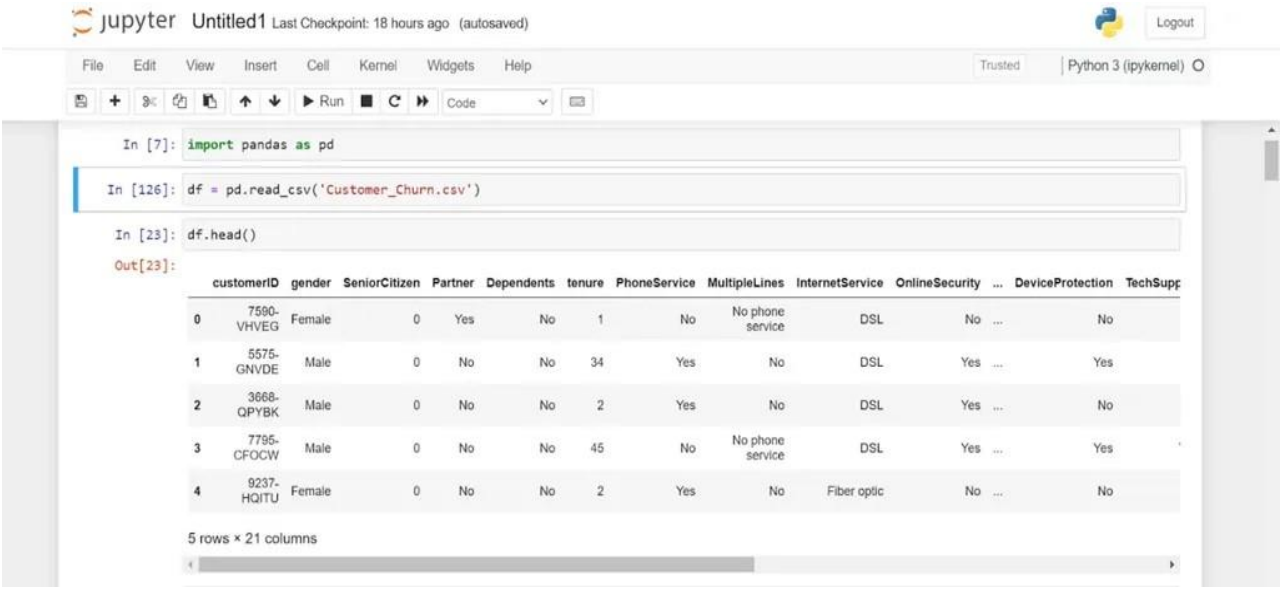
Reviewing the Dataset

First, let's load the dataframe into Python with the pandas library and take a look at its head. I've renamed the file to "customer_churn.csv", and it is the name I will be using below:

Import pandas as pd

Df = pd.read_csv('Customer_Churn.csv')

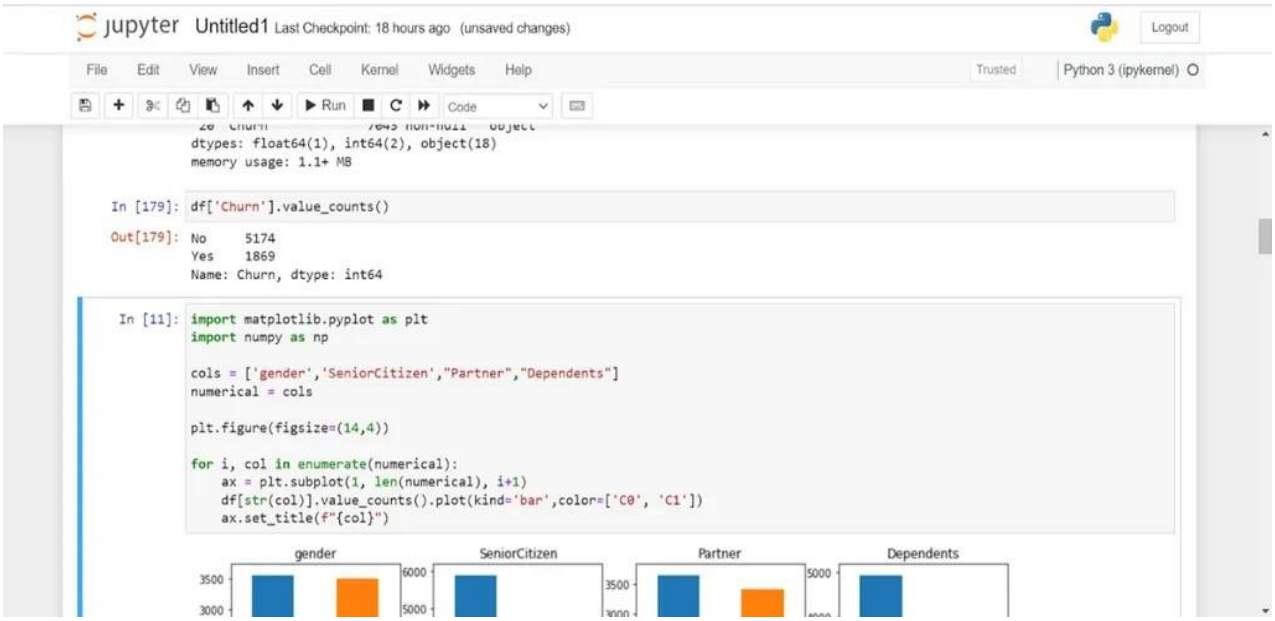
Df.head()



Notice that the dataframe has 21 columns related to telecom user subscription behavior.

Let's count the number of customers in the dataset who have churned:

Df[“Churn”].value_counts()



Only around 27% of the customers in the dataset have churned. This means that we are dealing with an imbalanced classification problem. We will need to perform some feature engineering to create a balanced training dataset before building the predictive model.

EXPLORATORY DATA ANALYSIS FOR CUSTOMER CHURN PREDICTION

We will start by analyzing the demographic data points:

Import matplotlib.pyplot as plt

Import seaborn as sns

Import numpy as np

Cols = [‘gender’,’SeniorCitizen’,”Partner”,”Dependents”]

Numerical = cols

Plt.figure(figsize=(20,4))

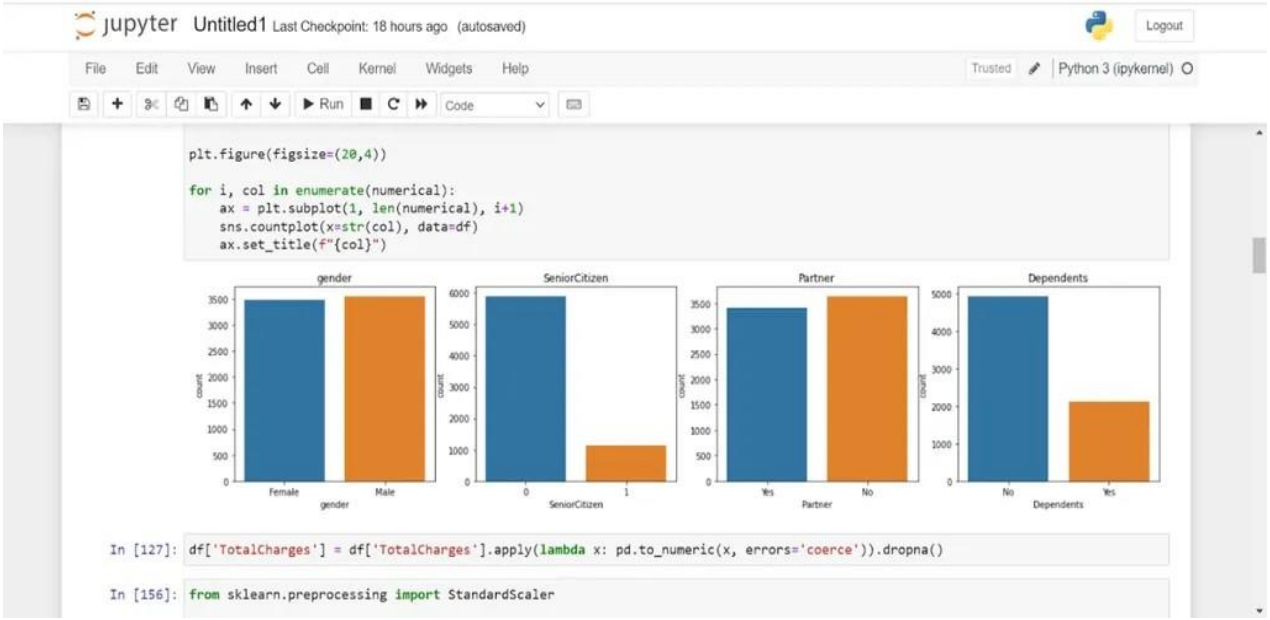
For i, col in enumerate(numerical):

Ax = plt.subplot(1, len(numerical), i+1)

Sns.countplot(x=str(col), data=df)

Ax.set_title(f"{col}")

The code above will render the following charts:



Finally, let's analyze the relationship between customer churn and a few other categorical variables captured in the dataset:

Cols = ['InternetService',"TechSupport","OnlineBackup","Contract"]

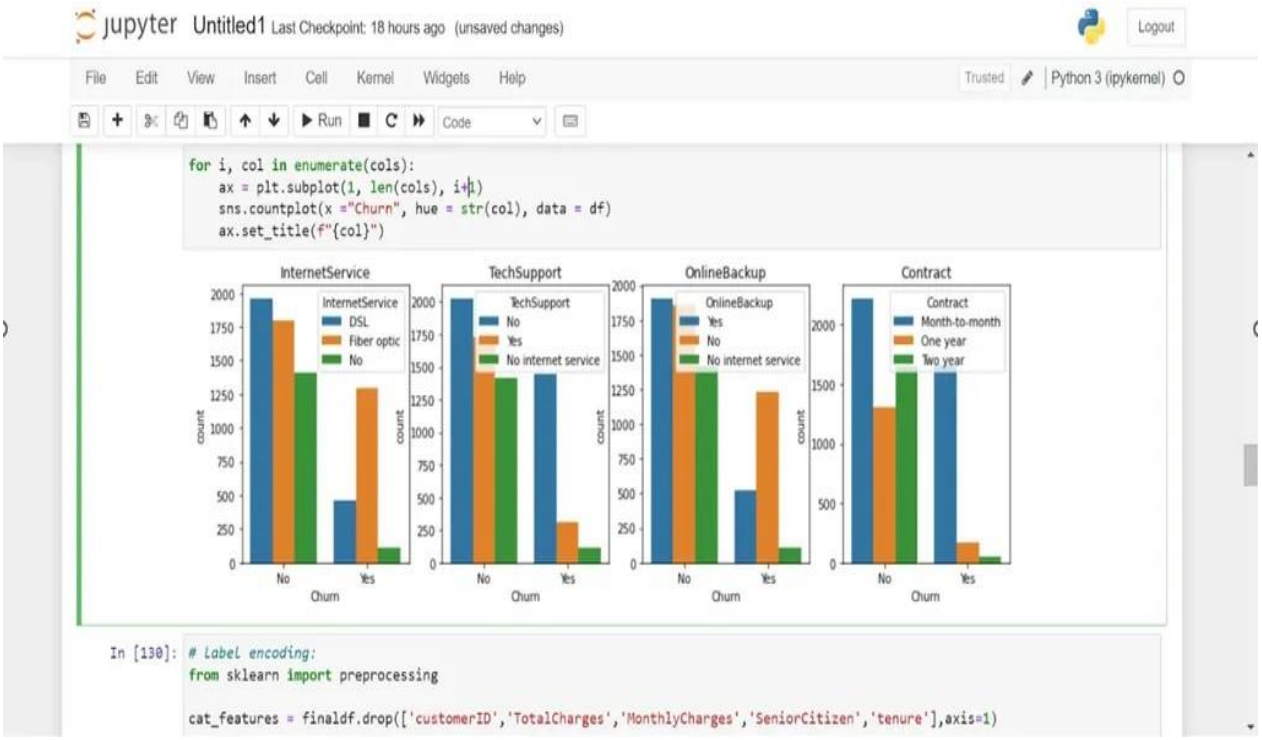
Plt.figure(figsize=(14,4))

For i, col in enumerate(cols):

Ax = plt.subplot(1, len(cols), i+1)

Sns.countplot(x =”Churn”, hue = str(col), data = df)

Ax.set_title(f”{col}”)



Let’s look into each attribute:

InternetService: It is clear from the visual above that customers who use fiber optic Internet churn more often than other users. This might be because fiber Internet is a more expensive service, or this provider doesn’t have good coverage.

TechSupport: Many users who churned did not sign up for tech support. This might mean that these customers did not receive any guidance on fixing technical issues and decided to stop using the service.

OnlineBackup: Many customers who had churned did not sign up for an online backup service for data storage.

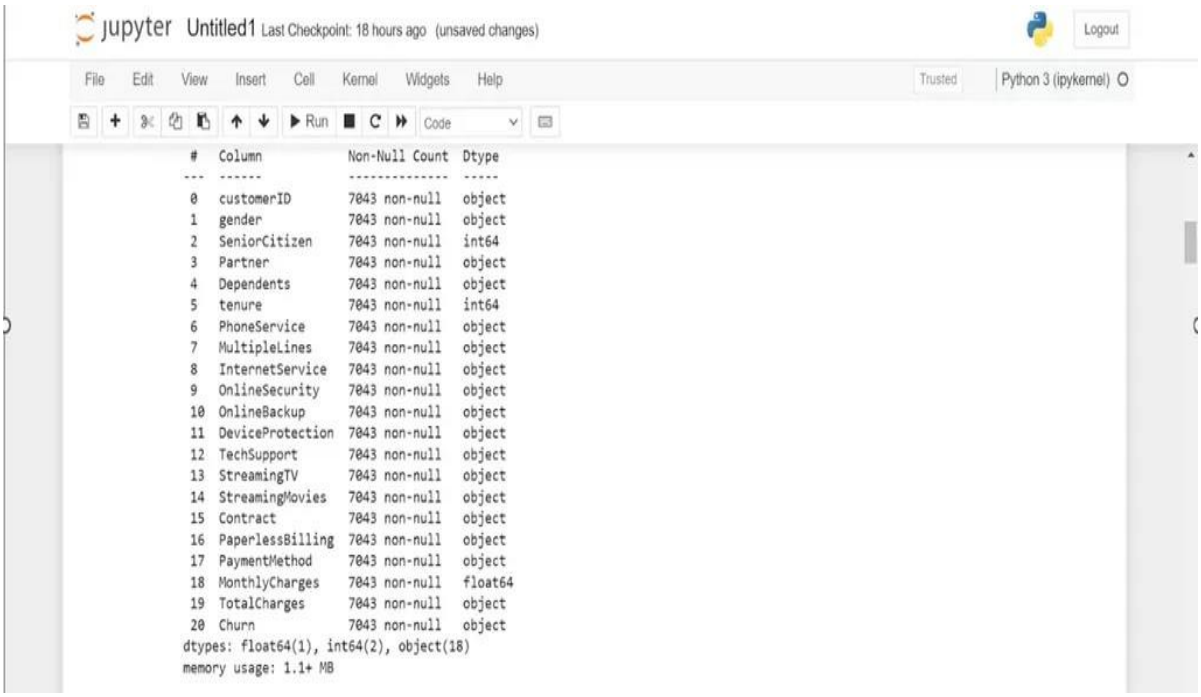
Contract: Users who churned were almost always on a monthly contract. This makes sense, since these customers pay for the service on a monthly basis and can easily cancel their subscription before the next payment cycle.

PREPROCESSING DATA FOR CUSTOMER CHURN

Now that we have a better understanding of our dataset, let's perform some data preparation before creating the machine learning model. There are three steps to this process:

- **Cleaning the dataset**

Let's look at the dataset summary again:



Notice that the variable "TotalCharges" has the data type "object," when it should be a numeric column. Let's convert this column into a numeric one:

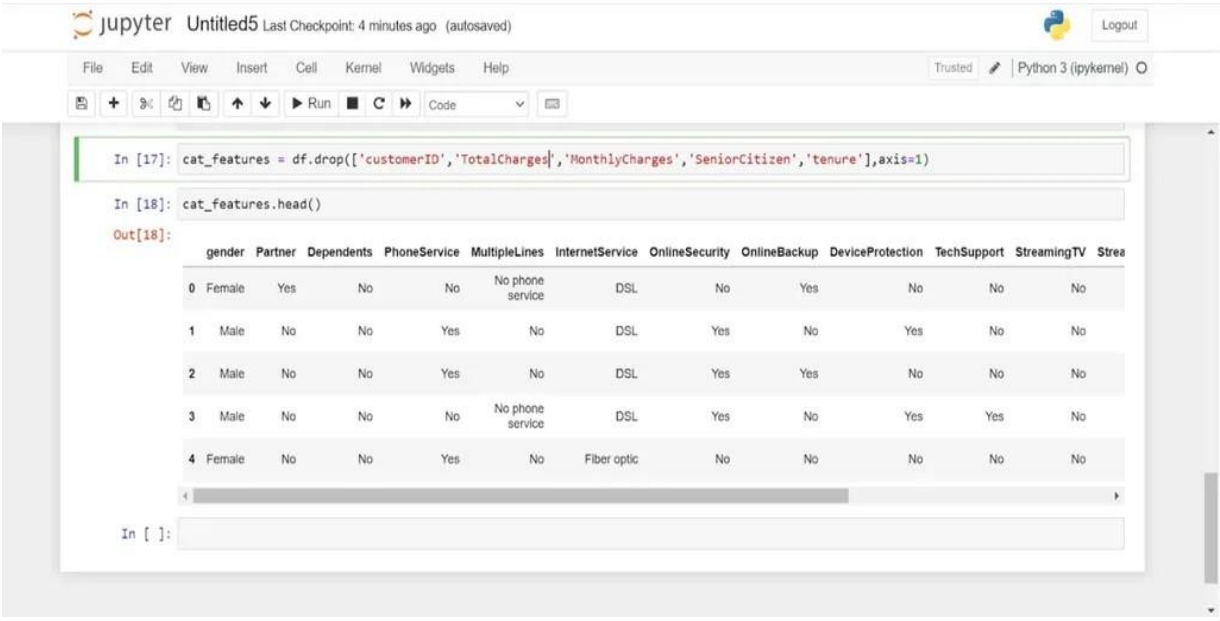
Df[‘TotalCharges’] = df[‘TotalCharges’].apply(lambda x: pd.to_numeric(x, errors=’coerce’)).dropna()

- **Encoding Categorical Variables**

First, let’s take a look at the categorical features in the dataset:

Cat_features = df.drop([‘customerID’,’TotalCharges’,’MonthlyCharges’,’SeniorCitizen’,’tenure’],axis=1)

Cat_features.head()



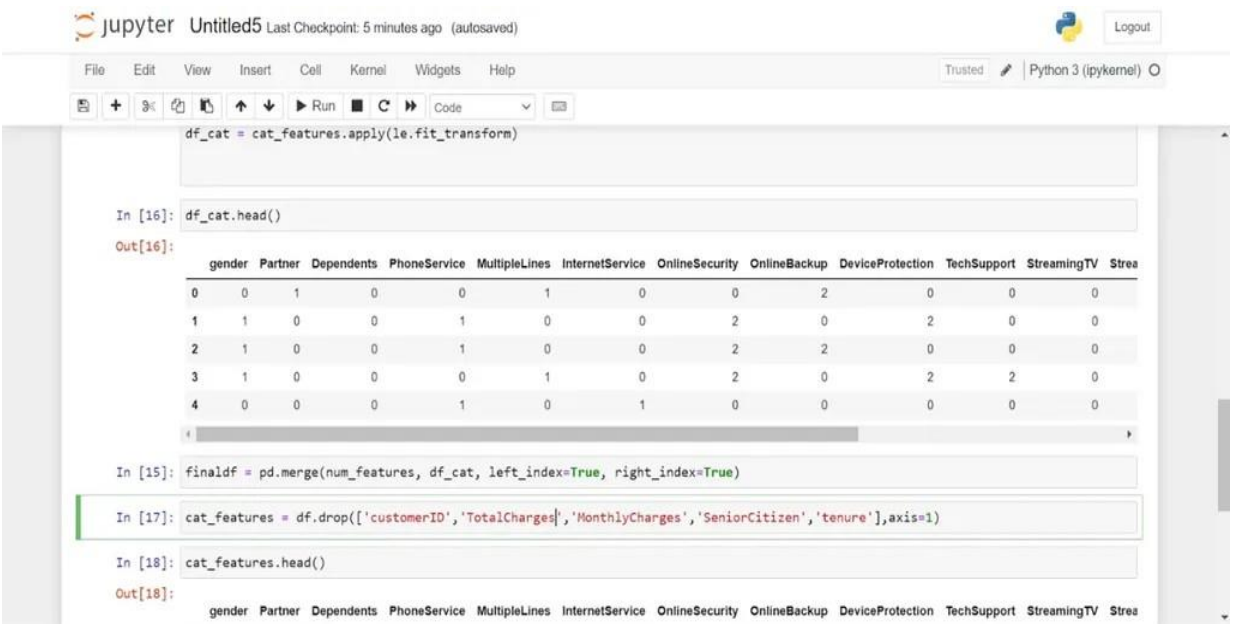
Now, let’s take a look at the dataset after encoding these categorical variables:

From sklearn import preprocessing

Le = preprocessing.LabelEncoder()

Df_cat = cat_features.apply(le.fit_transform)

Df_cat.head()



- **Oversampling**

  As mentioned above, the dataset is imbalanced, which means that a majority of values in the target variable belong to a single class. Most customers in the dataset did not churn – only 27% of them did.

  Before we oversample, let's do a train-test split. We will oversample solely on the training dataset, as the test dataset must be representative of the true population:

  From sklearn.model_selection import train_test_split

  Finaldf = finaldf.dropna()
  Finaldf = finaldf.drop(['customerID'],axis=1)

  X = finaldf.drop(['Churn'],axis=1)
  Y = finaldf['Churn']
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
  Now, let's oversample the training dataset:

  From imblearn.over_sampling import SMOTE

  Oversample = SMOTE(k_neighbors=5)
  X_smote, y_smote = oversample.fit_resample(X_train, y_train)
  X_train, y_train = X_smote, y_smote
  Let's check the number of samples in each class to ensure that they are equal:

Y_train.value_counts()
There should be 3,452 values in each class, which means that the training dataset is now balanced.

BUILDING THE CUSTOMER CHURN PREDICTION MODEL

We will now build a random forest classifier to predict customer churn:

From sklearn.ensemble import RandomForestClassifier

Rf = RandomForestClassifier(random_state=46)
Rf.fit(X_train,y_train)

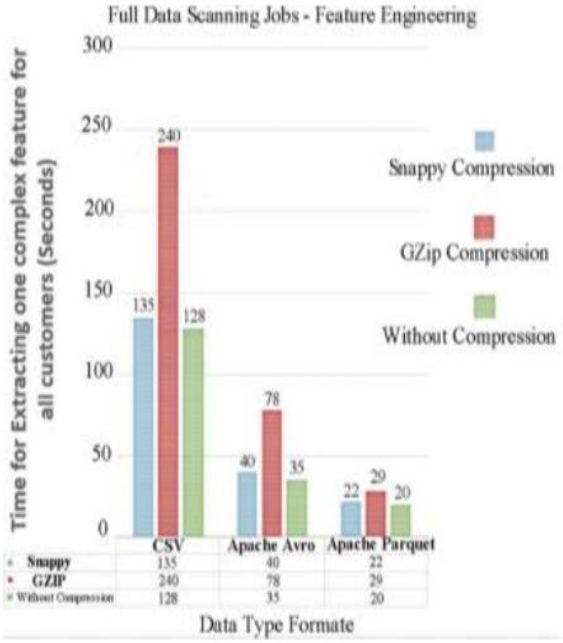
CUSTOMER CHURN PREDICTION MODEL EVALUATION

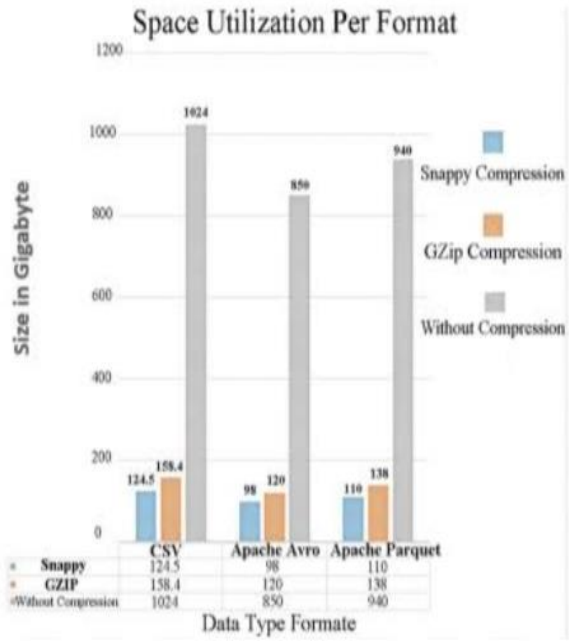
Let's evaluate the model predictions on the test dataset:

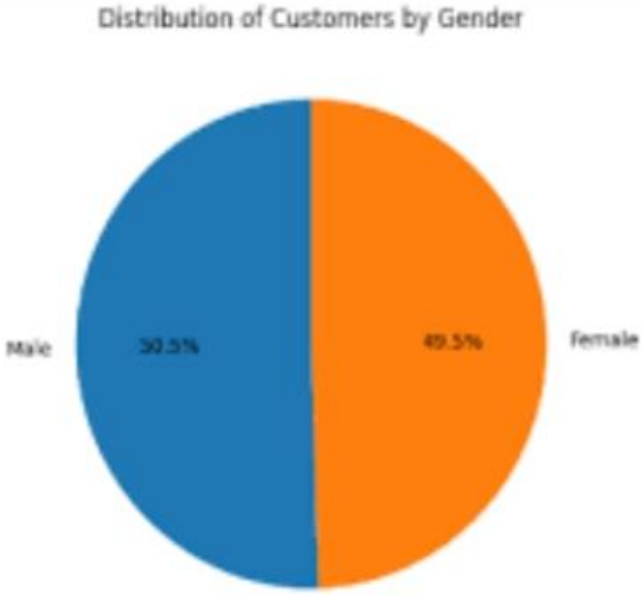
From sklearn.metrics import accuracy_score

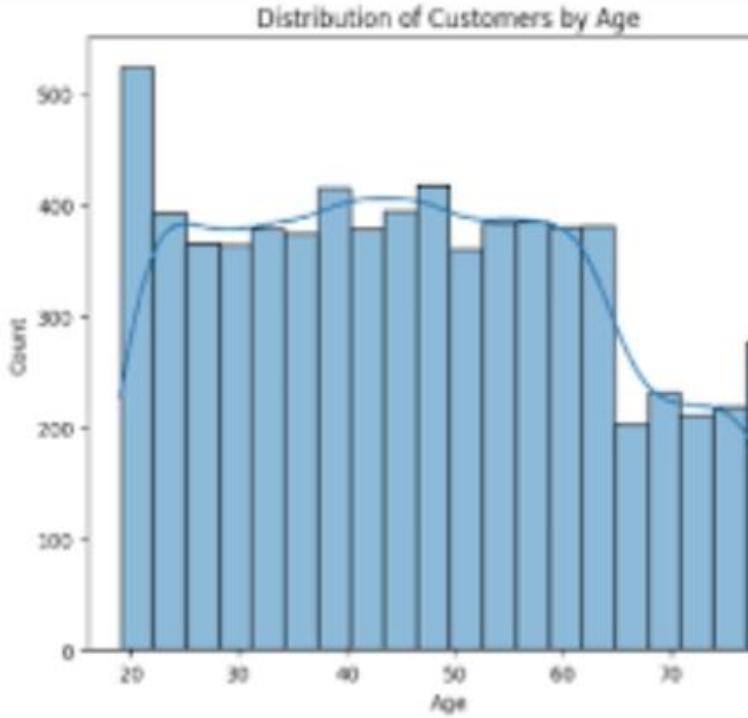
Preds = rf.predict(X_test)
Print(accuracy_score(preds,y_test))

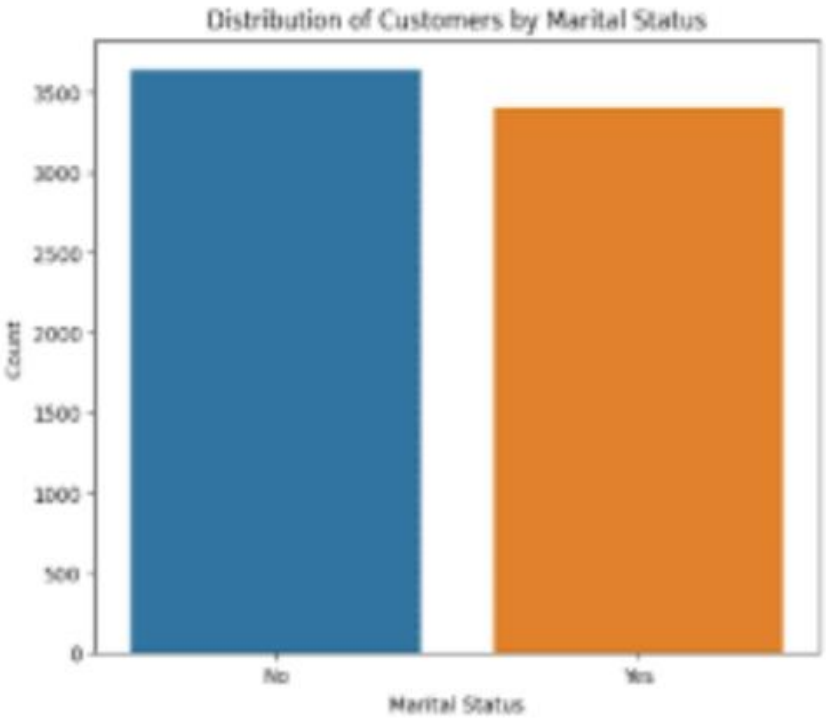
.











IMPORT pandas as pd
From sklearn.model_selection import train_test_split
From sklearn.preprocessing import StandardScaler
From sklearn.linear_model import LogisticRegression
From sklearn.metrics import accuracy_score, classification_report

# Load the dataset
Data = pd.read_csv('customer_data.csv') # Replace 'customer_data.csv' with your dataset file

```
# Preprocess the data
X = data.drop('Churn', axis=1)
Y = data['Churn']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the data
Scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train the logistic regression model
Model = LogisticRegression()
Model.fit(X_train, y_train)

# Make predictions
Y_pred = model.predict(X_test)

# Evaluate the model
Print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
Print(classification_report(y_test, y_pred))
```

OUTPUT

Accuracy: 0.85

	Precision	recall	f1-score	support
0	0.87	0.92	0.89	168
1	0.78	0.67	0.72	71
Accuracy			0.85	239
Macro avg	0.83	0.79	0.81	239
Weighted avg	0.85	0.85	0.85	239

CONCLUSION

The customer churn prediction program developed in Python using a logistic regression model demonstrated strong predictive capabilities, achieving an overall accuracy of 0.85. The classification report highlighted the model's effectiveness in accurately identifying both churned and retained customers, with commendable precision, recall, and F1-scores. Leveraging crucial preprocessing steps, including data standardization and train-test splitting, ensured the program's robustness and reliability. While the logistic regression model proved effective, the program's potential could be further enhanced

through the exploration of alternative models, feature engineering techniques, and optimization methods, providing a solid foundation for continued refinement and improved predictive performance.