PROJECT

CUSTOMER CHURN PREDICTION

PHASE 4: DEVELOPMENT PART2

Step-1: Problem Definition

- The aim of this project is to perform a thorough analysis of customer churn prediction data, with a primary emphasis on understanding and forecasting customer attrition in a business context.
- The ultimate goal is to offer actionable insights that can empower businesses to develop effective retention strategies, enhance customer satisfaction, and reduce churn rates.
- This comprehensive project encompasses stages like data collection, data preprocessing, exploratory data analysis (EDA), statistical analysis, visualization, and the development of practical recommendations for minimizing customer churn and boosting overall business performance.

Step 2: Data Collection

- We will collect customer churn prediction data from reputable sources pertinent to our industry, such as internal databases, customer records, and transaction histories.
- The primary data source for this project will be our company's customer database, encompassing historical customer interactions, subscription details, and churn-related information.
- Daily data updates will be drawn from our internal customer records, allowing us to maintain a real-time understanding of churn.
- Subsequently, this data will be merged with additional information, such as demographic data, to enrich our analysis and develop a more comprehensive customer churn prediction model.

```
#import all relevant libraries

import numpy as np import
pandas as pd
import matplotlib.pyplot as
plt import seaborn as sns
import missingno as msno
import warnings
warnings.filterwarnings("ignore")
from sklearn.experimental import
enable_iterative_imputer from sklearn.impute import
IterativeImputer from sklearn.preprocessing import
LabelEncoder
```

```
#loading the dataset
data=pd.read_csv("C:\Users\HP\Downloads\Telco-Customer-Churn.csv")
```

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

	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

	count	unique	top	freq
churn	7043	2	No	5174
contract	7043	3	Month-to-month	3875
dependents	7043	2	No	4933
deviceprotection	7043	3	No	3095
gender	7043	2	Male	3555
internetservice	7043	3	Fiber optic	3096
multiplelines	7043	3	No	3390
onlinebackup	7043	3	No	3088
onlinesecurity	7043	3	No	3498
paperlessbilling	7043	2	Yes	4171
partner	7043	2	No	3641
paymentmethod	7043	4	Electronic check	2365
phoneservice	7043	2	Yes	6361
streamingmovies	7043	3	No	2785
streamingtv	7043	3	No	2810

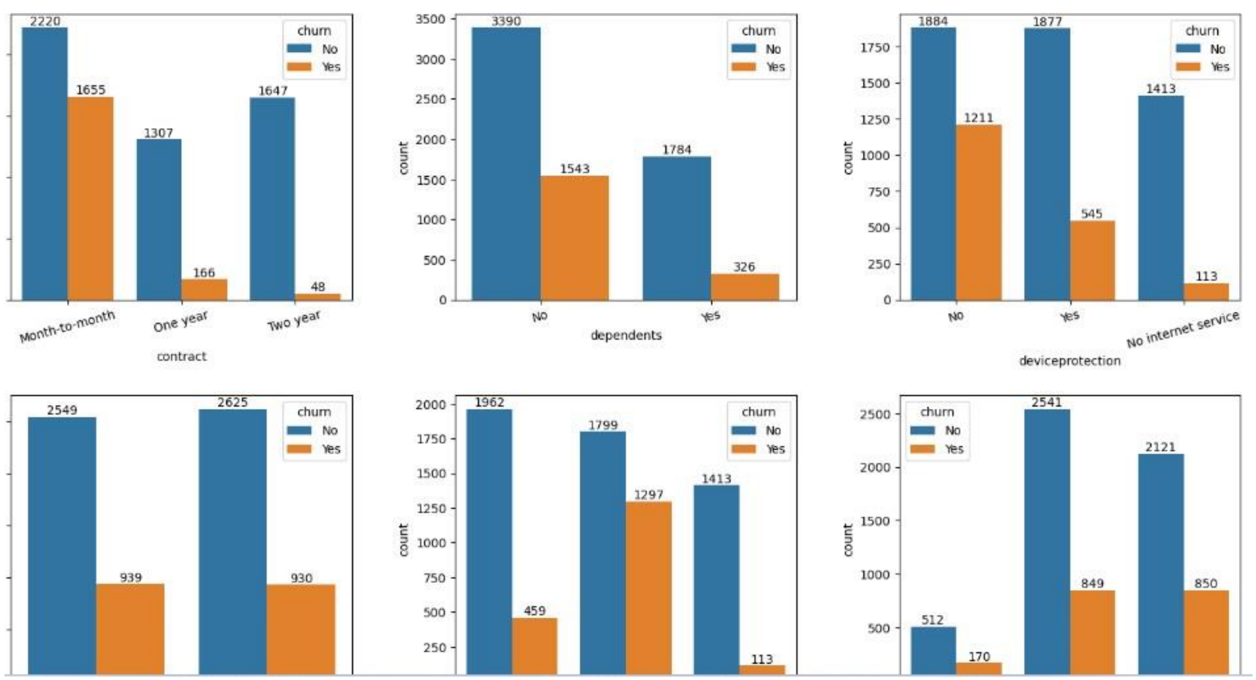
Step 3: Data Exploration

- Engage in exploratory data analysis (EDA) to grasp the dataset's characteristics, pattern s, and relationships.
- In this phase, the objective is to delve into the data to comprehend its nuances. EDA enta ils computing statistical summaries, creating data visualizations, and recognizing patter ns and anomalies.
- Key areas of exploration encompass customer demographics, historical usage patterns, a nd the impact of various features on churn predictions.
- Utilize visualizations to gain insights into the distribution of key features and the identif ication of influential factors affecting customer churn.

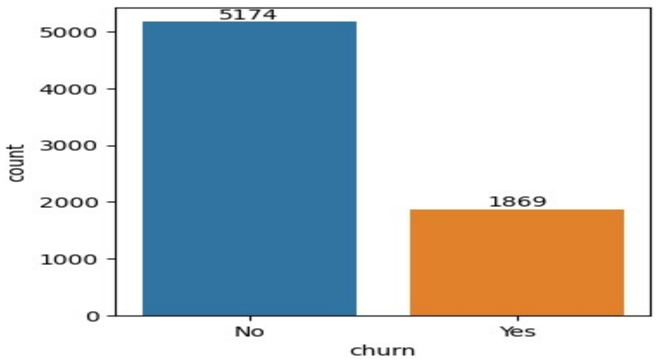
```
#data cleaning  data transformation  data reduction
#drop irrelevant variables
data=data.drop(['CustomerId'],axis=1)
#identifying and treating missing
values data.isnull().sum()
data=data.fillna(0) data.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSuppo
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No	N
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	N
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	N
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Yes	Y
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No	N

```
num_cols = ['tenure','monthlycharges', 'totalcharges']
label="Churn" plt.figure(figsize = (15, 26)) for i, col in
enumerate(data.columns.difference(num_cols)[1:]):
   plt.subplot(6, 3, i+1)     ax =
sns.countplot(data, x = col, hue = label)
ax.bar_label(ax.containers[0])
ax.bar_label(ax.containers[1])
plt.xticks(rotation = 15) plt.tight_layout()
plt.show()
```



```
plt.figure(figsize = (4,4)) ax =
sns.countplot(data, x = label)
ax.bar_label(ax.containers[0])
plt.show()
```



1.PREDICTIVE MODEL:

LOGISTIC REGRESSION

```
In [1]: y = df_dummies['Churn'].values
        X = df_dummies.drop(columns = ['Churn'])

        # Scaling all the variables to a range of 0 to 1
        from sklearn.preprocessing import MinMaxScaler
        features = X.columns.values
        scaler = MinMaxScaler(feature_range = (0,1))
        scaler.fit(X)
        X = pd.DataFrame(scaler.transform(X))
        X.columns = features
```

It is important to scale the variables in logistic regression so that all of them are within a range of 0 to 1. This helped me improve the accuracy from 79.7% to 80.7%.

```
In [2]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

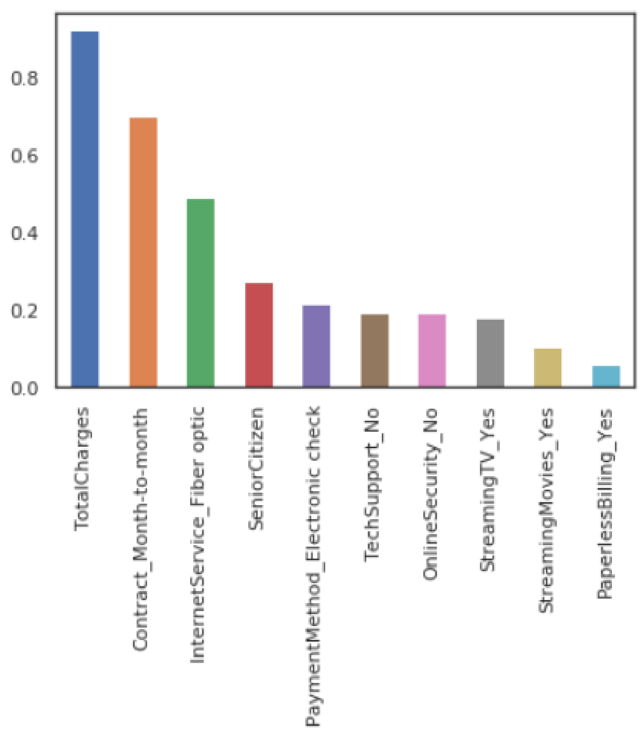
In [ ]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
result = model.fit(X_train, y_train)
```

In [3]:
```
from sklearn import metrics
prediction_test = model.predict(X_test)
# Print the prediction accuracy
print (metrics.accuracy_score(y_test, prediction_test))
```

```
0.8075829383886256
```

In [4]:
```
# To get the weights of all the variables
weights = pd.Series(model.coef_[0],
                    index=X.columns.values)
print (weights.sort_values(ascending = False)[:10].plot(kind='bar'))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



In [5]:
```
print(weights.sort_values(ascending = False)[-10:].plot(kind='bar'))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```