CUSTOMER CHURN PREDICTION

Data Analytics with cognos – Phase 3

DOCUMENTATION

Team Members:

1.ANANDHI V(au613021205002)

2.DEEPANA S(au613021205004)

3.HARITHA R(au613021205015)

4.KARTHIKA B(au613021205023)

5.SHARMILA C(au613021205049)

Problem Definition:

Start the data analysis by loading and preprocessing the dataset. Load the dataset using python and data manipulation libraries (e.g., pandas).

Dataset Link:

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

Overview of the process:

1.Import Libraries:

Begin by importing the necessary libraries, such as pandas for data manipulation.

2.Load the Dataset:

Use pd.read_csv() or other appropriate methods to load your dataset into a pandas DataFrame.

3.Explore the Dataset:

Display the initial rows, check for missing values, and explore basic statistics to understand the structure and content of the data.

4.Handle Missing Values:

Decide on an appropriate strategy for dealing with missing values, such as dropping rows or filling values based on a specific strategy.

5.Additional Preprocessing Steps:

Depending on the nature of your data, consider additional preprocessing steps such as feature scaling, handling outliers, processing date-time features, dealing with text data, feature engineering, or discretization.

6.Save Preprocessed Dataset (Optional):

Save the preprocessed dataset to a new file if significant changes have been made.

STEP 1:Loading the dataset

1.Importing libraries

Here, for preprocessing the dataset and manipulate the data, pandas is the library used to frame the data.

Code:

```
import pandas as pd
```

2.Loading the dataset

In this step, we are framing the data into the table using DataFrame in pandas, and display the head or 5 rows of the dataset.

Code:

```
# Replace with the actual filename
data=pd.read_csv("C:/Users/91962/Desktop/phase3_dataset.csv")
```

STEP 2:Explore the dataset:

After framing data, the first few or five rows of the data in displayed using the head() function.

Code:

```
data
```

OUTPUT:

```
In [10]: data=pd.read_csv("C:/Users/91962/Desktop/phase3_dataset.csv")
         data
```

Out[10]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechS
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	
...	...	...	...	...	...	...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	Yes	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	Yes	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	No	
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...	No	
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	...	Yes	

7043 rows × 21 columns

```
In [5]:
```

STEP 3:Check for missing values

In this step, the missing values or null values, if it present in the data are separated and number of null values are shown through this code.

Code:

print("Missing values:\n", data.isnull().sum())

Output:

```
In [12]: print("Missing values:\n", data.isnull().sum())

Missing values:
 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
```

STEP 4:Check datatype

In this step, the data type of the columns are discussed

Code:

print("Data Types:\n", data.dtypes)

OUTPUT:

```
In [13]: print("Data Types:\n", data.dtypes)

Data Types:
 customerID           object
gender                object
SeniorCitizen          int64
Partner               object
Dependents            object
tenure                 int64
PhoneService          object
MultipleLines         object
InternetService       object
OnlineSecurity        object
OnlineBackup          object
DeviceProtection      object
TechSupport           object
StreamingTV           object
StreamingMovies       object
Contract              object
PaperlessBilling      object
PaymentMethod         object
MonthlyCharges       float64
TotalCharges          object
Churn                 object
dtype: object
```

STEP 5:Check basic statistics

The statistics of the columns such as count, mean, std, min, max, 25%, 50%, 75% are shown through the describe() function command.

Code:

print("Summary Statistics:\n", data.describe())

OUTPUT:

```
In [14]: print("Summary Statistics:\n", data.describe())

Summary Statistics:
       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
```

STEP 6:Saving Preprocessed dataset

In this step, if we made substantial changes to the dataset and want to sa ve the preprocessed version, you can use the following Code.

Code:

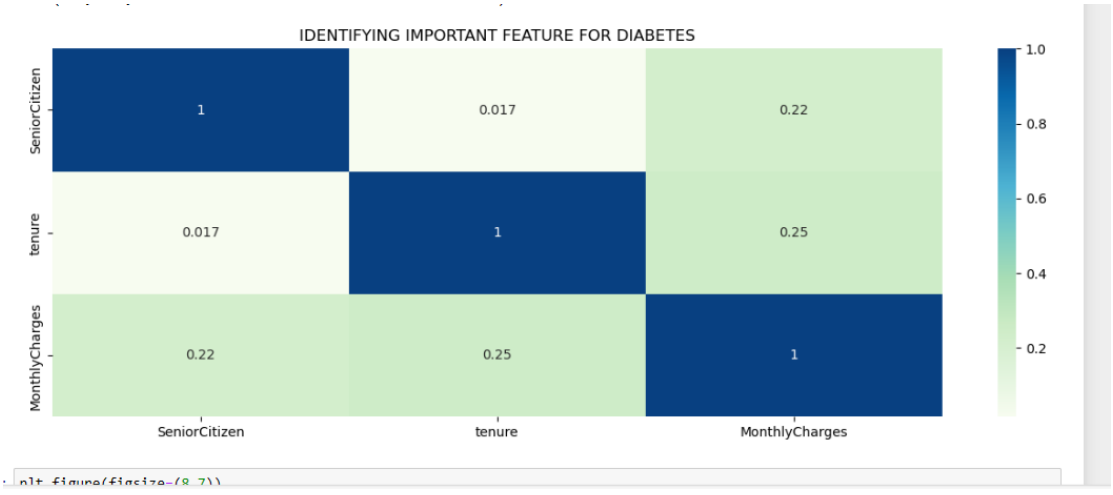
```
# Save the preprocessed dataset to a new CSV file
df.to_csv('preprocessed_dataset.csv', index=False)
```

STEP 7:DATA VISUALIZATION

CORRELATION GRAPH:

```
import matplotlib.pyplot as plt
import seaborn as sns


plt.figure(figsize=(15,5))
sns.heatmap(data.corr(),cmap="GnBu",annot=True)
plt.title("IDENTIFYING IMPORTANT FEATURE FOR DIABETES")
```

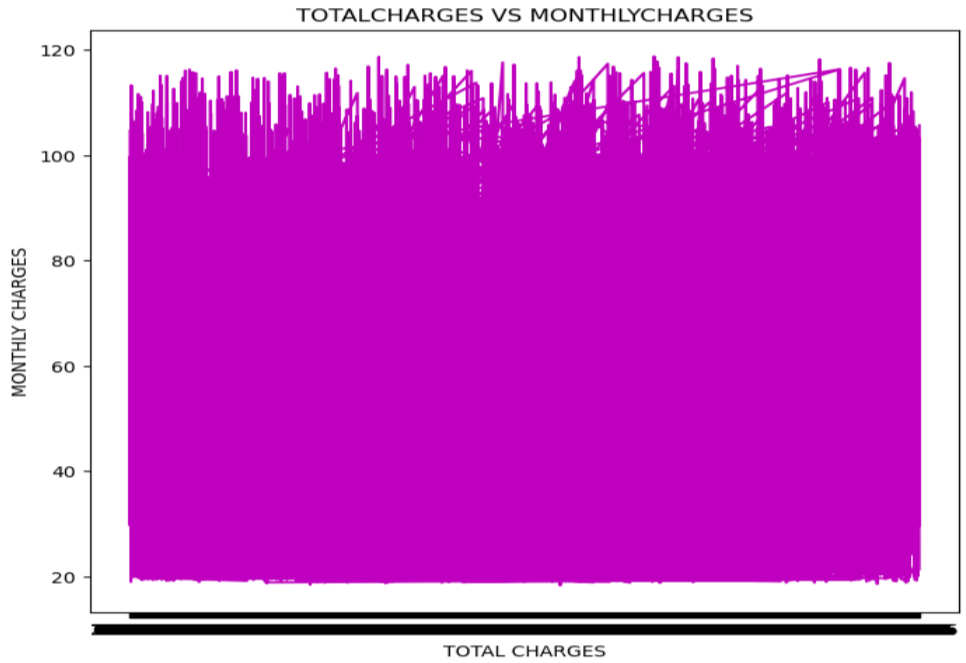
OUTPUT:



PLOT:

```
plt.figure(figsize=(8,7))
plt.plot(data.TotalCharges,data.MonthlyCharges,'m')
plt.title("TOTALCHARGES VS MONTHLYCHARGES")
plt.xlabel("TOTAL CHARGES")
plt.ylabel("MONTHLY CHARGES")
```

OUTPUT:

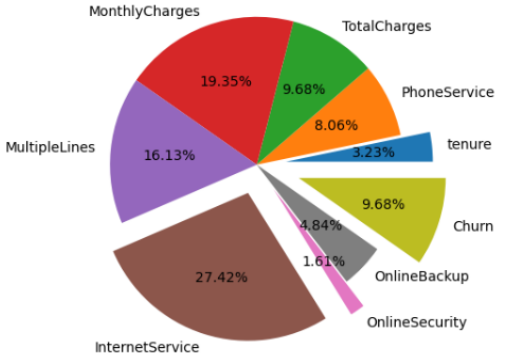


COMPOSITION PLOT:

```
x=[2000,5000,6000,12000,10000]
y=['Travel','Savings','Rent','Home','food']
plt.pie(x,labels=y,autopct='%0.2f%%',explode=[0.2,0,0,0,0])
```

OUTPUT:

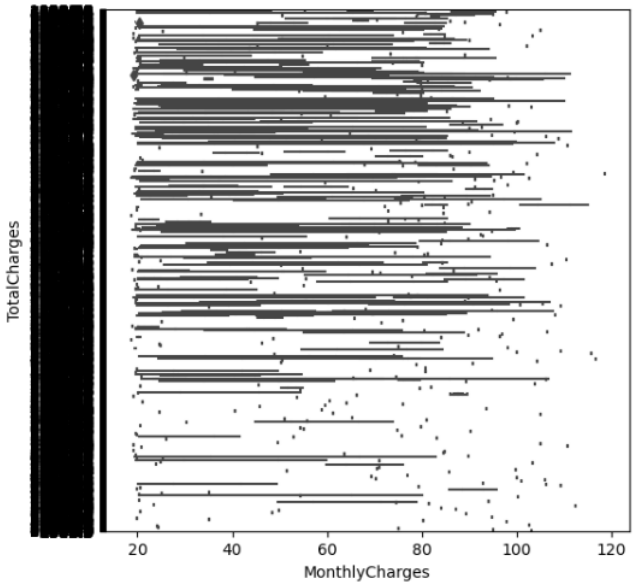
```
Out[12]: ([<matplotlib.patches.Wedge at 0x1649831b880>,
           <matplotlib.patches.Wedge at 0x1649831b790>,
           <matplotlib.patches.Wedge at 0x16498344490>,
           <matplotlib.patches.Wedge at 0x16498344b20>,
           <matplotlib.patches.Wedge at 0x164983451b0>,
           <matplotlib.patches.Wedge at 0x16498345840>,
           <matplotlib.patches.Wedge at 0x16498345ed0>,
           <matplotlib.patches.Wedge at 0x16498346560>,
           <matplotlib.patches.Wedge at 0x16498346b90>],
          [Text(1.2933301208066794, 0.13151881467752297, 'tenure'),
           Text(0.9875850001117432, 0.484433553291149, 'PhoneService'),
           Text(0.5818604395536556, 0.933508665670773, 'TotalCharges'),
           Text(-0.38203571893599053, 1.031527367284582, 'MonthlyCharges'),
           Text(-1.0943562456479174, 0.111285253342008, 'MultipleLines'),
           Text(-0.38917216475955435, -1.2403810004092946, 'InternetService'),
           Text(0.7426486103384069, -1.0669925217931162, 'OnlineSecurity'),
           Text(0.7972720121625676, -0.7578636675697357, 'OnlineBackup'),
           Text(1.3357949235177018, -0.41910848512567406, 'Churn')],
          [Text(0.7958954589579563, 0.08093465518616798, '3.23%'),
           Text(0.5386827273336781, 0.26423648361335395, '8.06%'),
           Text(0.3173784215747212, 0.5091865449113306, '9.68%'),
           Text(-0.20838311941963117, 0.5626512912461356, '19.35%'),
           Text(-0.5969215885352277, 0.060701047277458904, '16.13%'),
           Text(-0.2394905629289565, -0.7633113848672582, '27.42%'),
           Text(0.45701452943901955, -0.6566107826419175, '1.61%'),
           Text(0.4348756429977641, -0.41338018231076484, '4.84%'),
           Text(0.8587253079756655, -0.26942688329507614, '9.68%')])
```



DISTRIBUTION PLOT

plt.figure(figsize=(6,6))
sns.boxplot(x='MonthlyCharges',y='TotalCharges',data=data)

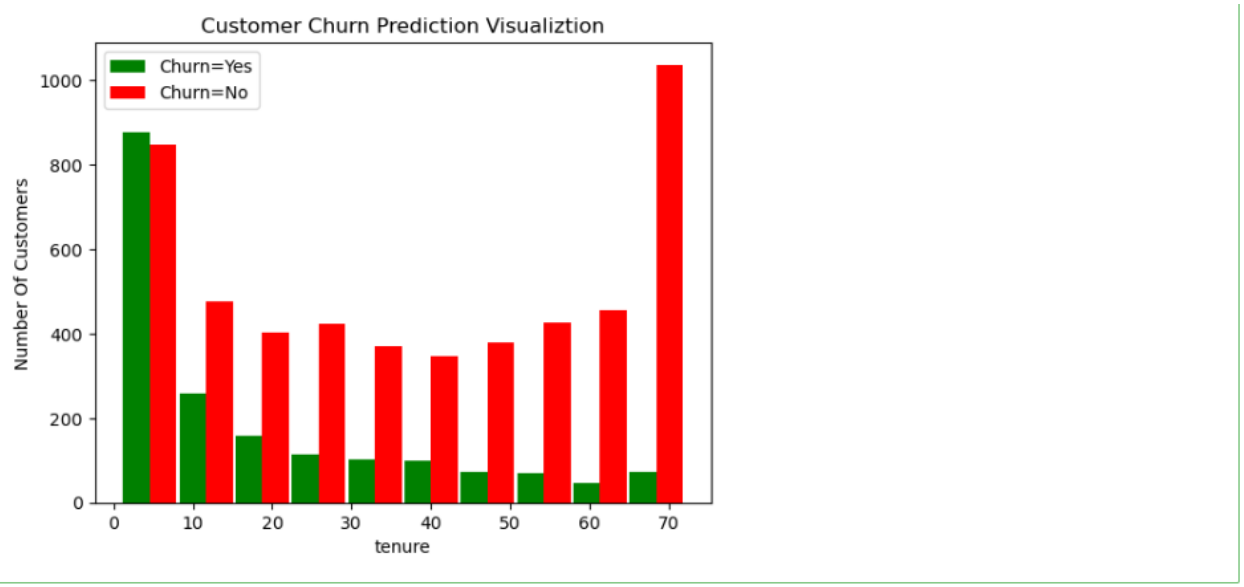
OUTPUT:



HISTOGRAM

```
tenure_churn_no = df1[df1.Churn=='No'].tenure
tenure_churn_yes = df1[df1.Churn=='Yes'].tenure

plt.xlabel("tenure")
plt.ylabel("Number Of Customers")
plt.title("Customer Churn Prediction Visualiztion")

plt.hist([tenure_churn_yes, tenure_churn_no], rwidth=0.95, color=['green',
'red'],label=['Churn=Yes','Churn=No'])
plt.legend();
```

OUTPUT:

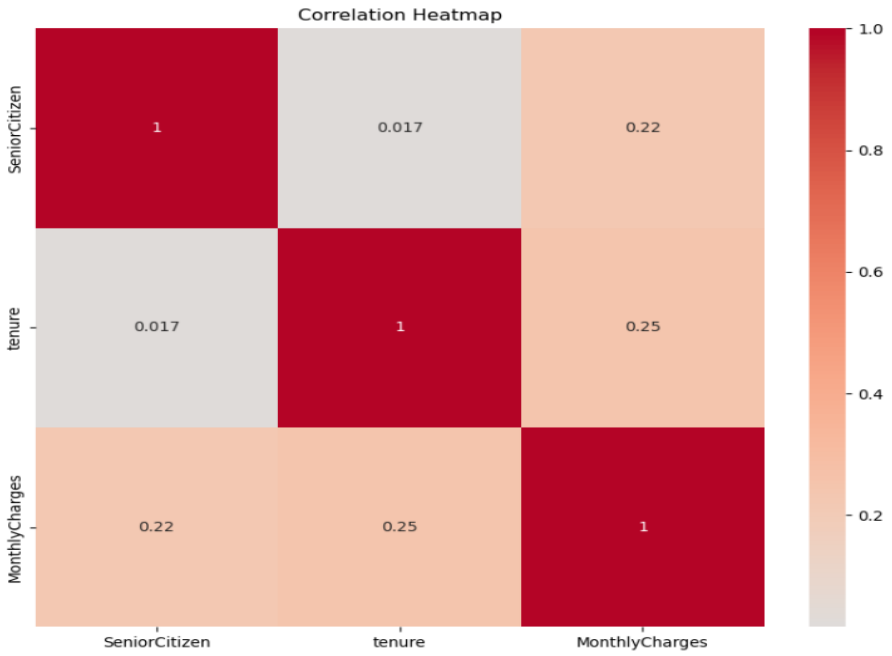


BOX PLOT:

```
def EDA (data):
   def corr(data):
      numeric_columns = data.select_dtypes(exclude=['object']).col
umns
      corr_matrix= data[numeric_columns].corr()


      plt.figure(figsize=(10, 8))
      sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', ce
nter=0)
      plt.title('Correlation Heatmap')
      plt.show()

   def box(data,cat_feature,target):
     if target.dtype  !='object':
      fig, axes = plt.subplots(len(cat_feature)//3+1, 3, figsize=(10, 2
0))
```

```
    for i, f in enumerate(cat_feature):
        row = i // 3  # Row index of the subplot
        col = i % 3  # Column index of the subplot
        sns.boxplot(x=f,y=target,data=data, ax=axes[row, col])
    for i in range(len(cat_feature), len(axes.ravel())):
        fig.delaxes(axes.ravel()[i])

    plt.tight_layout()  # Adjust subplot spacing
    plt.show()

#box(data,cat_feature,data.iloc[:,-1])


def scatter(data,num_feature):
    #_,ax = plt.subplots(3,3,figsize=(12,4))
    #fig, axes = plt.subplots(len(cat_feature)//3+1, 3, figsize=(10,
20))

    #for i, f in enumerate(num_feature):
        #row = i // 3  # Row index of the subplot
        #col = i % 3  # Column index of the subplot

    #df.plot.scatter(x=num_feature[i-1],y=num_feature[i])
    #data.scatter(x=
    sns.pairplot(data)
    plt.tight_layout()
    plt.show()


corr(data)
print('box plot:')
box(data,cat_feature,data.iloc[:,-1])
print('scatter plot')
scatter(data,num_feature)
```

OUTPUT:



CONCLUSION:

In conclusion, the outlined data loading and preprocessing steps provide a foundational framework for preparing a dataset for analysis in Python using th e pandas library. By following these steps, you can ensure that your data is in a suitable format and quality for further exploration and visualization tasks.