College code:4212
Register number:421221243029

DATA ANALYTICS WITH COGNOS-GROUP:2

IBM NAAN MUDHALVAN

CUSTOMER CHURN PREDICTION-PHASE5

In the previous phases we defined the problem statements, solving of those problems, data manipulation and the visualizing techniques using the python in this phase we are going to discuss about some problem statements that are given in phase5 they are:

Phase 5: Project Documentation & Submission

In this part you will document your project and prepare it for submission.

Document the customer churn prediction project and prepare it for submission.

Documentation

- Outline the project's objective, design thinking process, and development phases.
- Describe the analysis objectives, data collection process, data visualization using IBM Cognos, and predictive modeling.
- Explain how the insights and prediction model can help businesses reduce customer churn.

Submission

- Share the GitHub repository link containing the project's code and files.
- Provide instructions on how to replicate the analysis and generate visualizations using IBM Cognos and build the predictive model using Python.
- Include example outputs of the visualizations and model evaluation.

College code:4212
Register number:421221243029

OBJECTIVES:

Predicting the customer churn is a critical business problem in a variety of sectors including telephones, subscriptions services, e-commerce and more. Business may take proactive steps to retain consumers by utilizing churn prediction to identify the customers who are likely to discontinue their goods or services.

DESIGN THINKING PROCESS:

In the each documentation phases we are discussed about the step by step explanation of the project like Clearly define the problem, Data collection, Preparing of the data, Exploratory data analyses, Feature selection, Model selection, Model training and validation, Model evaluation, Result representation, Reporting and visualization, Business action,etc…

DEVELOPMENT PHASES:

In the each phases we have developed the project from scratch like **development steps, methods steps to develop the project ,how to develop the project and also visualizations using the IBM cognos and the in this phase we are developed the total project in Jupyter notebook.**

★ANALYSIS OBJECTIVE:

To this problem this dataset is given to us so by using this dataset we are going

to solve our problem.

Clearly define the problem:

We have clearly understood the problem that we are going to understand about

the customers who are likely stop the usage of the services in the telecom sector. We defined dataset Also we have defined the missing if null values, data types, visualization of missing values and **CREATED DADHBOARD IN IBM COGNOS** also we defined the data explanation in python and we finally created a confusion matrix result on **Random forest, Adaboost,** as well as in the **Gradient Descent Algorithm.**

College code:4212
Register number:421221243029

DATA COLLECTION PROCESS:

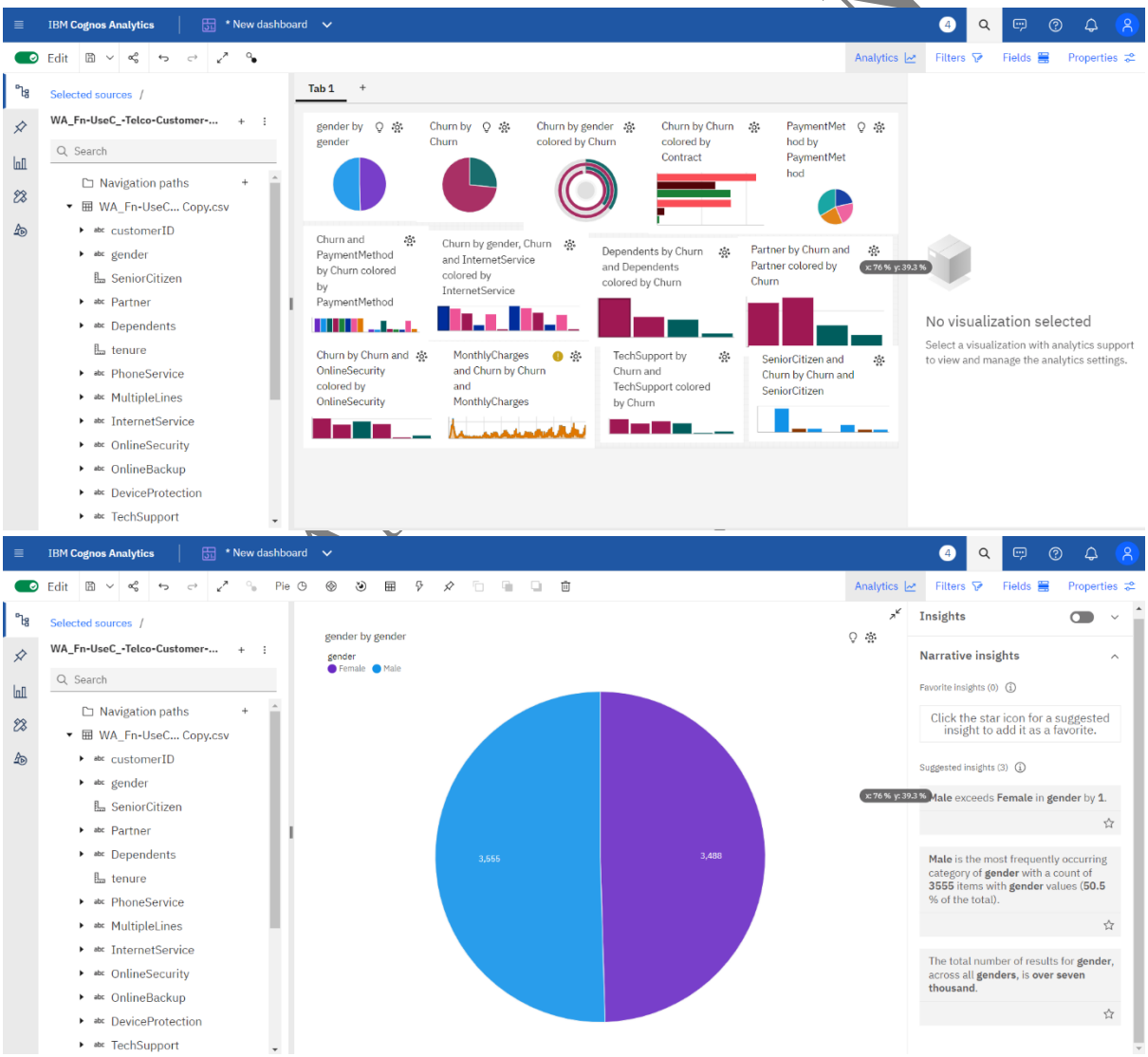
The data had been got from the IBM Itself under the Phase problems

The dataset is already given for us:

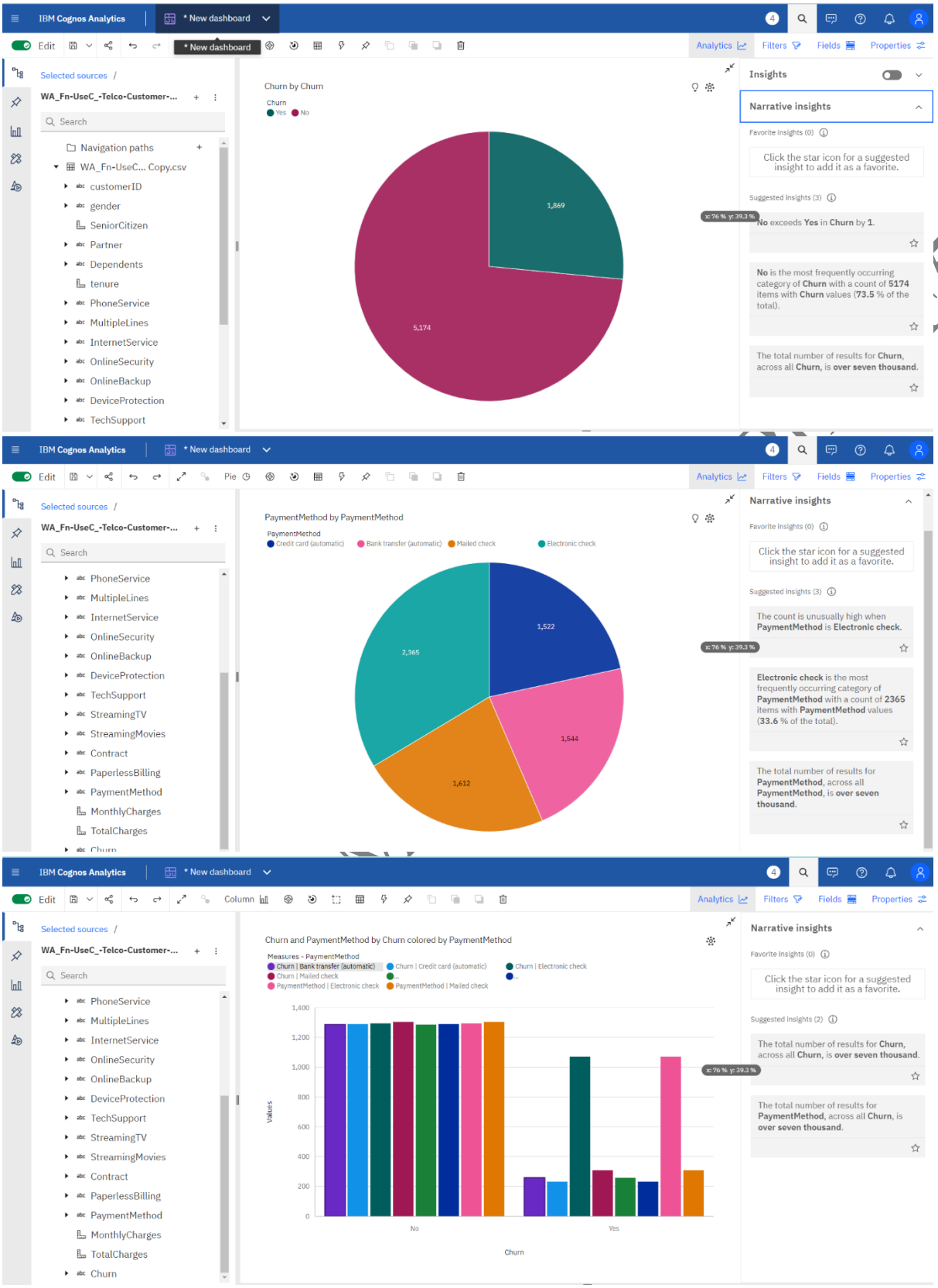
Datasetlink: https://www.kaggle.com/datasets/blastchar/telco-customer-churn

DATA VISUALIZATION USING IBM COGNOS:

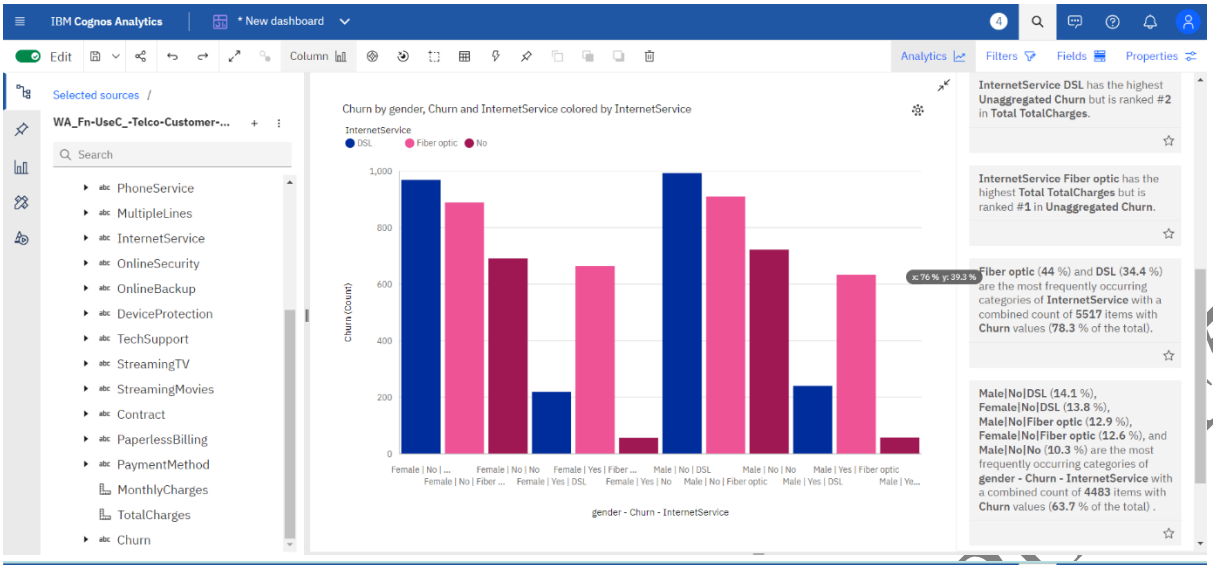
Below are the dash board and the explanation of each technique we defined in dashboard as follows:

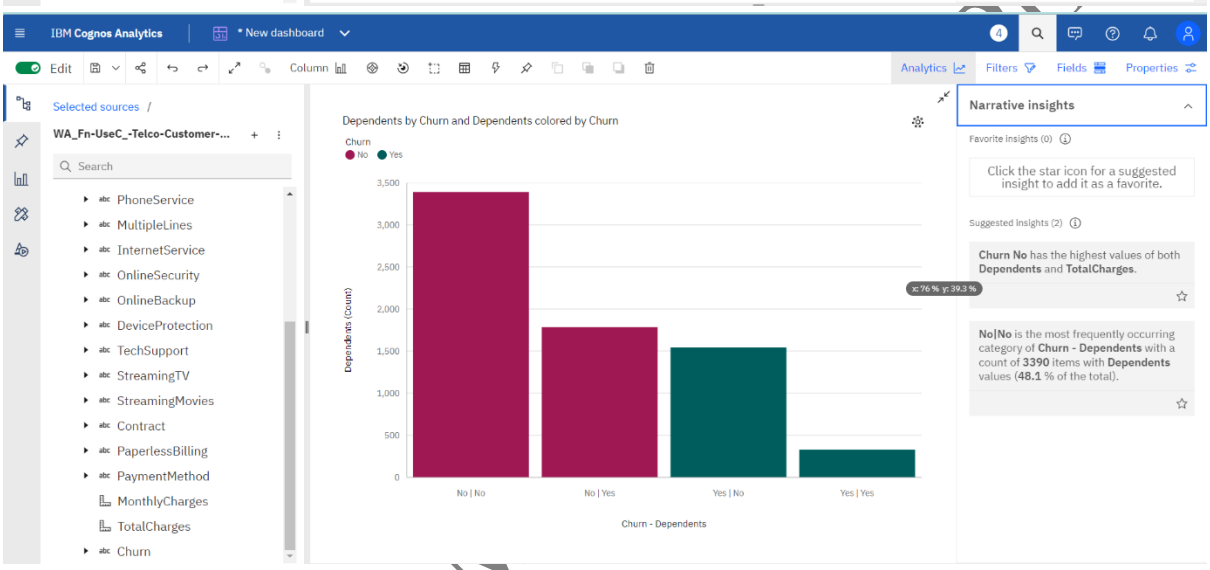


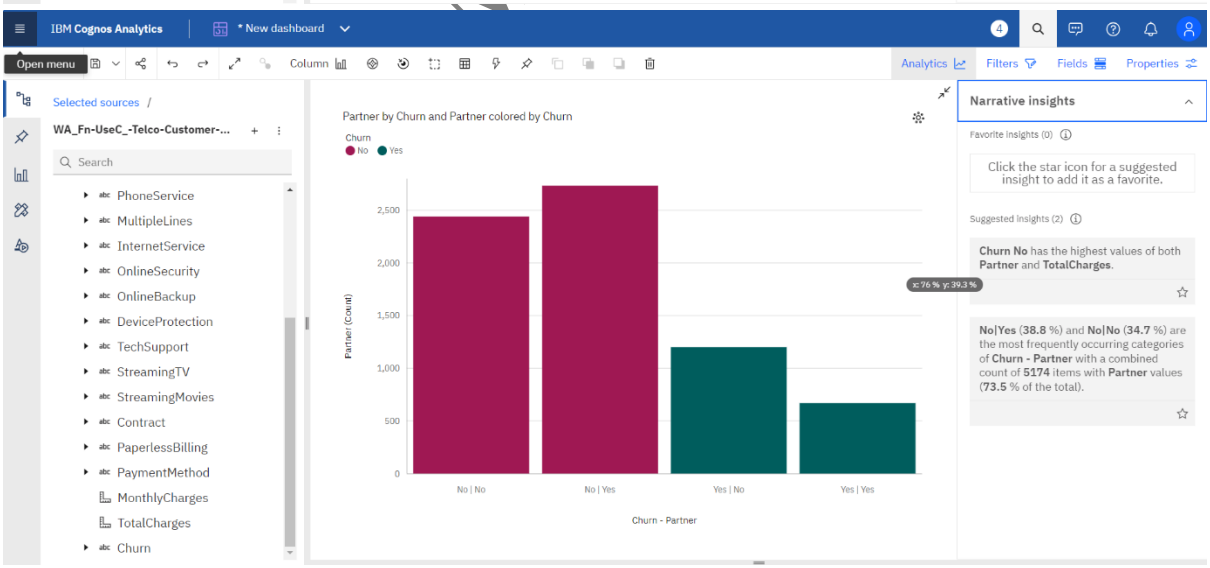
College code:4212
Register number:421221243029



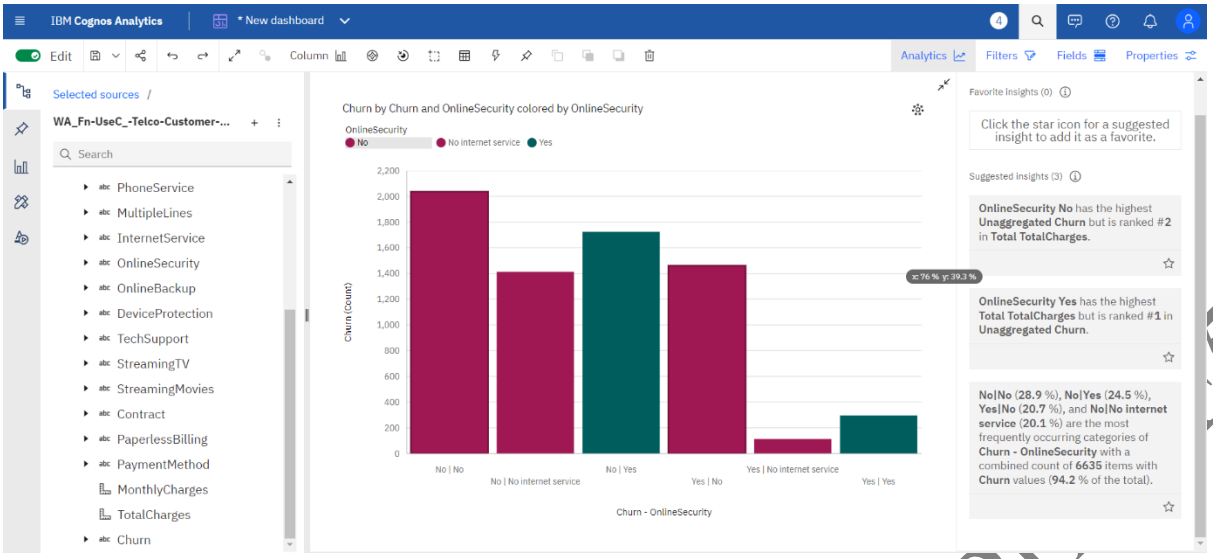
College code:4212
Register number:421221243029

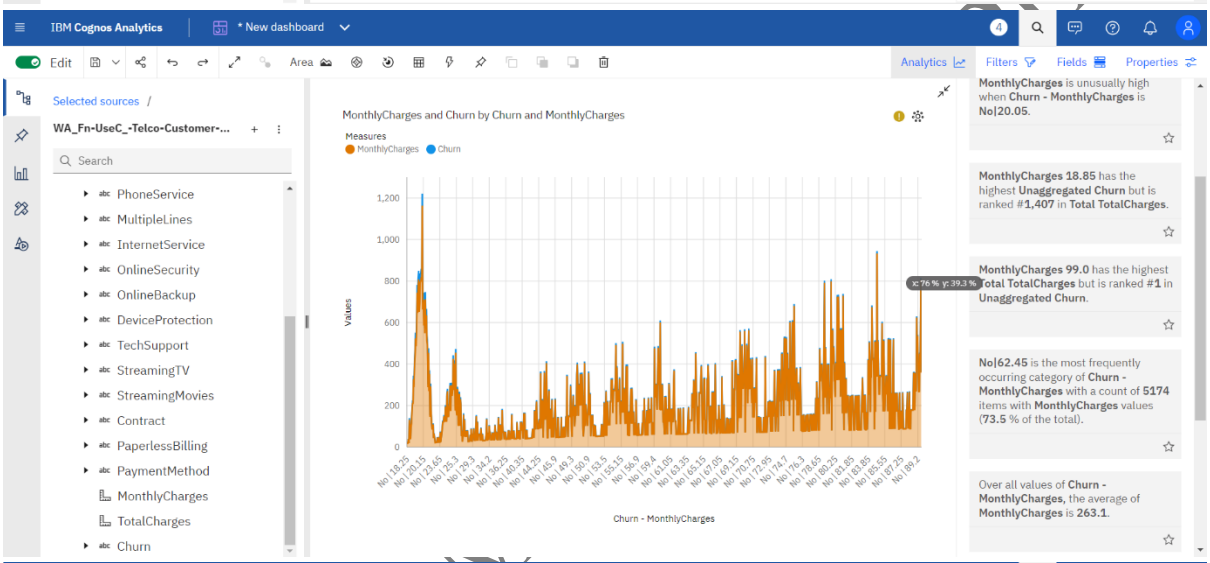


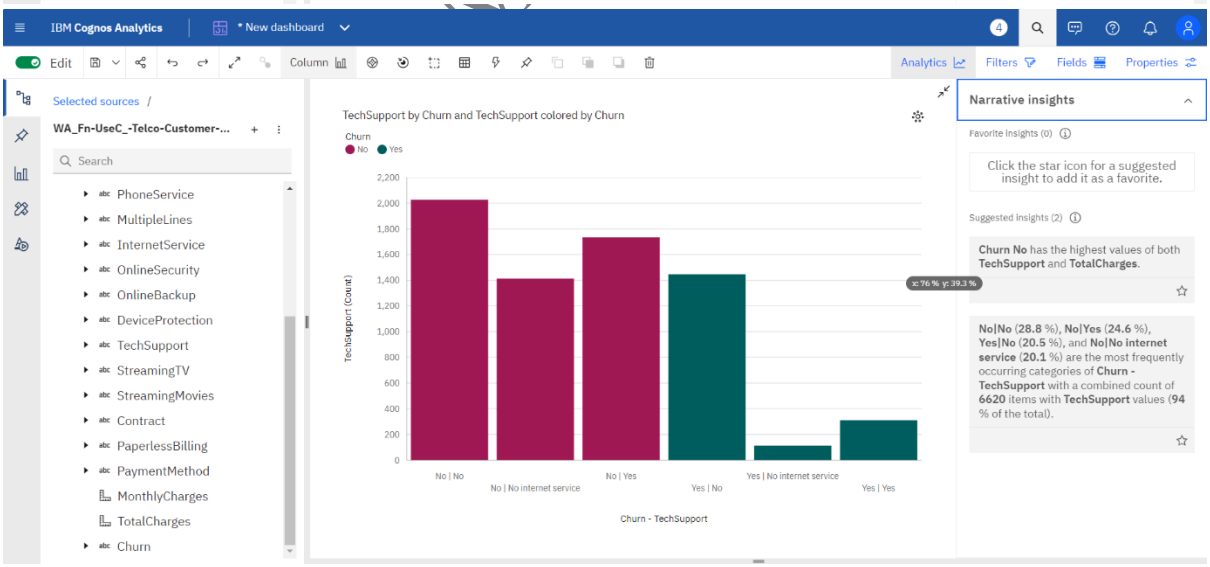




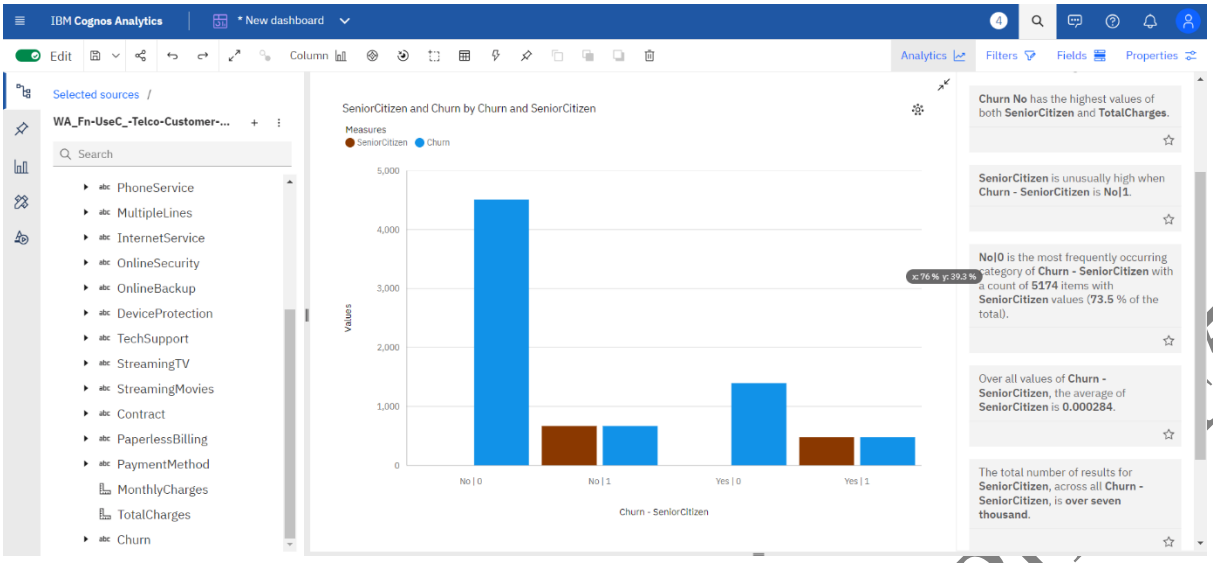
College code:4212
Register number:421221243029

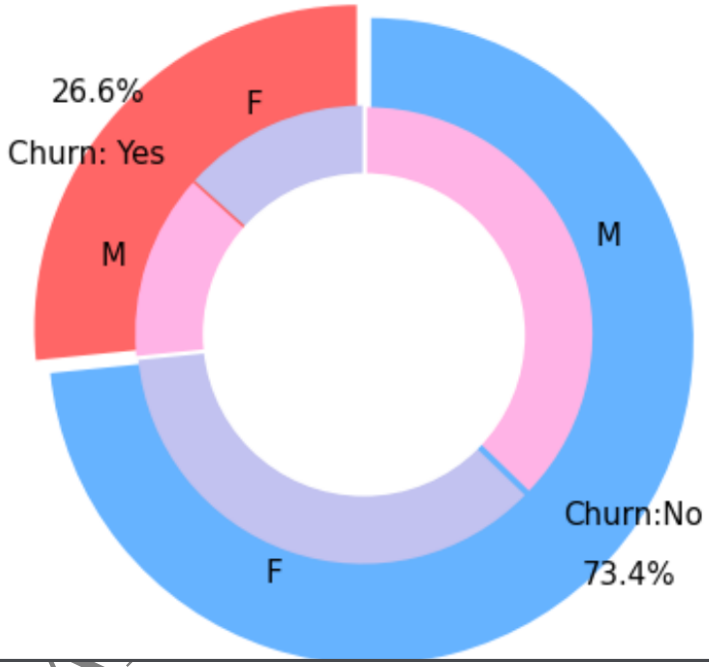




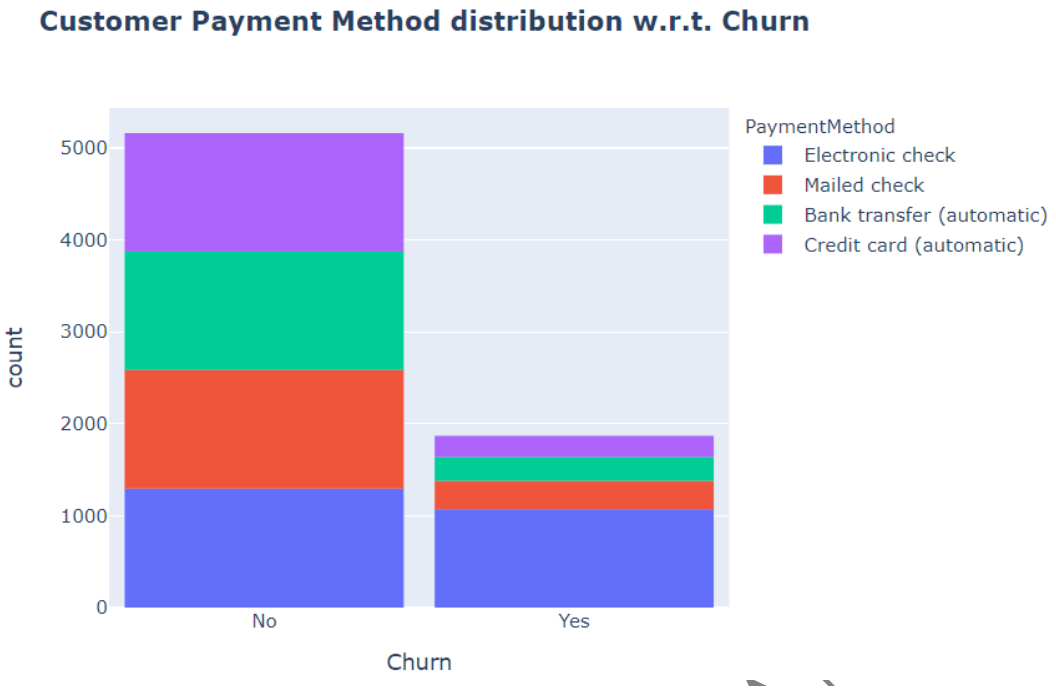


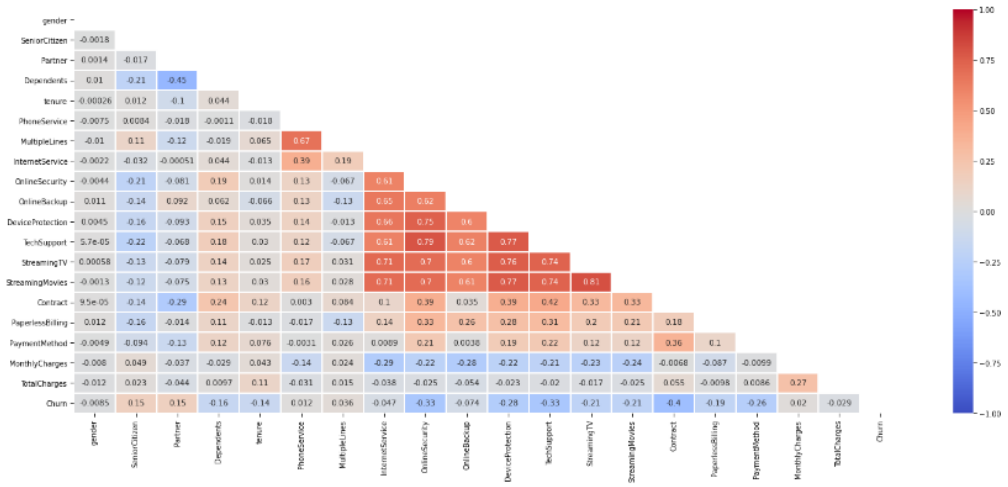
College code:4212
Register number:421221243029





College code:4212
Register number:421221243029





INSIGHTS DELIVERED FROM THE ABOVE PICTURES:

In the above picture of IBM dashboards we have explained about

In picture 1:

The **dashboard using the IBM Cognos** has been created.

In picture 2:

The gender in the dataset has been defined in the defined dataset the male was 3555 and female was 3488 the visualization clearly tells there is almost equal number of the gender and male is slightly more.

The pie chart clearly defined the male v/s female so that's why we use the piechart.

In picture3:

Overall boys and girls or men and women NO is the most frequently occurring category of churn with a count of 5174 and items with churn values (73.5%of the total)

In picture4:

In this visualization in the dashboard we created about the payment method that need to be paid by the people in different ways like credit card, Bank transfer, Mailed check, Electronic check. **Out of these Electronic check is the most frequently occurring category of the payment method with a count of 2365 items with payment method values.**

In picture5:

In this bar graph method of visualization we have defined the **churn in the various methods**

In picture6:

In this we have defined about the churn by gender,churn and internet service coloured by the internet service

The insights delivered here was about the **Internet service DSL has the highest unaggregated churn in the combined count of the 5517 items with churn values 78.3 of total.**

In picture7:

In picture7 we have defined a **bar graph of the dependents of the churn as well as the coloured of the churn has the highest values of both dependents and total charges.**

In picture8:

In this bar graph we have defined about **the partner by churn and has we delivered the insights of the churn no has the highest values of both partners and total charges.**

In picture9:

In this we have created about the various churn by churn and online security

The insights delivered from here was the online security no has the highest unaggregated churn but is ranked in the total charges.

Online security yes has the highest total charges but ranked 1 in the unaggregated churn.

In picture 10:

In this dashboard technique we have delivered an **insight with the help of the chart monthly charges 99.0 has the highest total charges but is ranked 1 in the aggregated churn.**

In picture11:

In these bar graph we have defined about the Tech support by churn and tech support coloured by churn and **insights are churn no has the highest values of both tech support and total charges.**

In picture 12:

In this method of visualization we verified about the chorn in senior citizen

The insights delivered here was the senior citizen is usually high when churn citizen is no.overall churn the senior citizen churn was about the average of 0.000284.

In picture:13

This type of visualization techniques helps to know about the different types of methods that have been deployed

The insights delivered here was the no churn have the equal payment method.

In picture:14

In this spiral visualization we come to know that the **churn and gender in a single visualization.**

In picture15:

We have created **heatmap** about the data.

These various insights in this dashboard makes the use of the customer at each individual about the data classification as well as the company will exactly focuses on various types of methodologies and also with the help of these techniques if the person wants to really leave the company so that each of them have provide some special offers to the customer as well as we have to create the some technological standards in the telecom industry so that everyone can use their product very efficiently for that we need to use these and have to provide the visualization insights to the company and need to tell the telecom company on which area they need to focus.

dac-ccp-phase-5

October 31, 2023

1 import necessary functions

[1]:
```
import pandas as pd
import numpy as np
import missingno as msno
```

2 import the csv file {dataset}

[2]:
```
df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

[3]:
```
df.head()
```

[3]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	…	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	7590-VHVEG	Female	0	Yes	No	1	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	…	Yes	No	No	No	One year	No	Mailed check	56.95	1889.5	No
2	3668-QPYBK	Male	0	No	No	2	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	…	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	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes

```
2               Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4           Electronic check          70.70        151.65   Yes

[5 rows x 21 columns]
```

[4]:
```
df.shape
```

[4]:
```
(7043, 21)
```

[5]:
```
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
```

[6]:
```
df.columns.values
```

[6]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
```

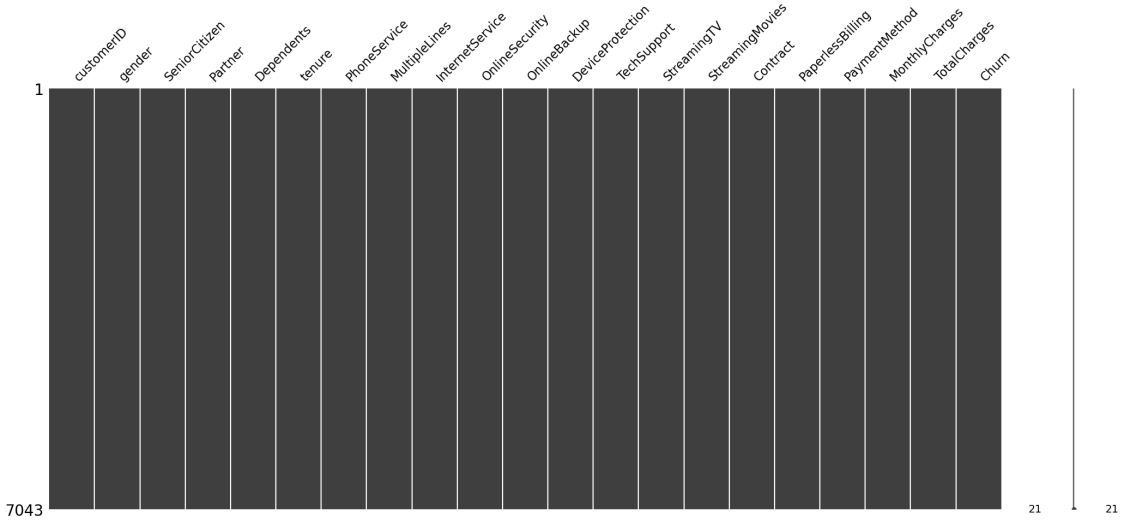
```
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

[7]:
```
df.dtypes
```

[7]:
```
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
```

3 visualize the missing values

[8]:
```
msno.matrix(df);
```



[9]:
```
df = df.drop(['customerID'], axis = 1)
df.head()
```

[9]:

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

4 Data manipulation

[10]:
```
df['TotalCharges'] = pd.to_numeric(df.TotalCharges, errors='coerce')
df.isnull().sum()
```

[10]:
```
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
```

[11]:
```
df[np.isnan(df['TotalCharges'])]
```

[11]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity
488	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes
753	Male	0	No	Yes	0	Yes	No	No	No internet service
936	Female	0	Yes	Yes	0	Yes	No	DSL	Yes
1082	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service
1340	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes
3331	Male	0	Yes	Yes	0	Yes	No	No	No internet service
3826	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service
4380	Female	0	Yes	Yes	0	Yes	No	No	No internet service
5218	Male	0	Yes	Yes	0	Yes	No	No	No internet service
6670	Female	0	Yes	Yes	0	Yes	Yes	DSL	No
6754	Male	0	No	Yes	0	Yes	No	DSL	Yes

```
6670              Yes             DSL                  No
6754              Yes             DSL                 Yes

           OnlineBackup     DeviceProtection          TechSupport  \
488                  No                  Yes                  Yes
753   No internet service  No internet service  No internet service
936                 Yes                  Yes                   No
1082  No internet service  No internet service  No internet service
1340                Yes                  Yes                  Yes
3331  No internet service  No internet service  No internet service
3826  No internet service  No internet service  No internet service
4380  No internet service  No internet service  No internet service
5218  No internet service  No internet service  No internet service
6670                Yes                  Yes                  Yes
6754                Yes                   No                  Yes

             StreamingTV      StreamingMovies  Contract PaperlessBilling  \
488                  Yes                   No  Two year              Yes
753   No internet service  No internet service  Two year               No
936                  Yes                  Yes  Two year               No
1082  No internet service  No internet service  Two year               No
1340                 Yes                   No  Two year               No
3331  No internet service  No internet service  Two year               No
3826  No internet service  No internet service  Two year               No
4380  No internet service  No internet service  Two year               No
5218  No internet service  No internet service  One year              Yes
6670                 Yes                   No  Two year               No
6754                  No                   No  Two year              Yes

                 PaymentMethod  MonthlyCharges  TotalCharges Churn
488   Bank transfer (automatic)           52.55           NaN    No
753                Mailed check           20.25           NaN    No
936                Mailed check           80.85           NaN    No
1082               Mailed check           25.75           NaN    No
1340    Credit card (automatic)           56.05           NaN    No
3331               Mailed check           19.85           NaN    No
3826               Mailed check           25.35           NaN    No
4380               Mailed check           20.00           NaN    No
5218               Mailed check           19.70           NaN    No
6670               Mailed check           73.35           NaN    No
6754  Bank transfer (automatic)           61.90           NaN    No
```

[12]:
```
df[df['tenure'] == 0].index
```

[12]:
```
Int64Index([488, 753, 936, 1082, 1340, 3331, 3826, 4380, 5218, 6670, 6754],
dtype='int64')
```

[13]:
```
df.drop(labels=df[df['tenure'] == 0].index, axis=0, inplace=True)
df[df['tenure'] == 0].index
```

[13]:
```
Int64Index([], dtype='int64')
```

[14]:
```
df.fillna(df["TotalCharges"].mean())
```

[14]:

```
      gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0     Female              0     Yes         No       1           No
1       Male              0      No         No      34          Yes
2       Male              0      No         No       2          Yes
3       Male              0      No         No      45           No
4     Female              0      No         No       2          Yes
…       …               …      …          …      …           …
7038    Male              0     Yes        Yes      24          Yes
7039  Female              0     Yes        Yes      72          Yes
7040  Female              0     Yes        Yes      11           No
7041    Male              1     Yes         No       4          Yes
7042    Male              0      No         No      66          Yes

         MultipleLines InternetService OnlineSecurity OnlineBackup  \
0     No phone service             DSL             No          Yes
1                   No             DSL            Yes           No
2                   No             DSL            Yes          Yes
3     No phone service             DSL            Yes           No
4                   No     Fiber optic             No           No
…                 …              …              …          …
7038               Yes             DSL            Yes           No
7039               Yes     Fiber optic             No          Yes
7040  No phone service             DSL            Yes           No
7041               Yes     Fiber optic             No           No
7042                No     Fiber optic            Yes           No

     DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
0                  No          No          No              No  Month-to-month
1                 Yes          No          No              No        One year
2                  No          No          No              No  Month-to-month
3                 Yes         Yes          No              No        One year
4                  No          No          No              No  Month-to-month
…                …           …           …               …              …
7038              Yes         Yes         Yes             Yes        One year
7039              Yes          No         Yes             Yes        One year
7040               No          No          No              No  Month-to-month
7041               No          No          No              No  Month-to-month
7042              Yes         Yes         Yes             Yes        Two year

     PaperlessBilling              PaymentMethod  MonthlyCharges  \
```

```
0                 Yes           Electronic check           29.85
1                  No               Mailed check           56.95
2                 Yes               Mailed check           53.85
3                  No  Bank transfer (automatic)           42.30
4                 Yes           Electronic check           70.70
…                 …                       …              …
7038              Yes               Mailed check           84.80
7039              Yes    Credit card (automatic)          103.20
7040              Yes           Electronic check           29.60
7041              Yes               Mailed check           74.40
7042              Yes  Bank transfer (automatic)          105.65

      TotalCharges Churn
0            29.85    No
1          1889.50    No
2           108.15   Yes
3          1840.75    No
4           151.65   Yes
…             …   …
7038       1990.50    No
7039       7362.90    No
7040        346.45    No
7041        306.60   Yes
7042       6844.50    No

[7032 rows x 20 columns]
```

[15]:
```
df.isnull().sum()
```

[15]:
```
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
```

[16]:
```
df["SeniorCitizen"]= df["SeniorCitizen"].map({0: "No", 1: "Yes"})
df.head()
```

[16]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	Female	No	Yes	No	1	No	No phone service	DSL	No	Yes	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
1	Male	No	No	No	34	Yes	No	DSL	Yes	No	Yes	No	No	No	One year	No	Mailed check	56.95	1889.50	No
2	Male	No	No	No	2	Yes	No	DSL	Yes	Yes	No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
3	Male	No	No	No	45	No	No phone service	DSL	Yes	No	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
4	Female	No	No	No	2	Yes	No	Fiber optic	No	No	No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes

[17]:
```
df["InternetService"].describe(include=['object', 'bool'])
```

[17]:
```
count            7032
unique              3
top       Fiber optic
```

```
freq              3096
Name: InternetService, dtype: object
```

[18]:
```
numerical_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']
df[numerical_cols].describe()
```

[18]:

	tenure	MonthlyCharges	TotalCharges
count	7032.000000	7032.000000	7032.000000
mean	32.421786	64.798208	2283.300441
std	24.545260	30.085974	2266.771362
min	1.000000	18.250000	18.800000
25%	9.000000	35.587500	401.450000
50%	29.000000	70.350000	1397.475000
75%	55.000000	89.862500	3794.737500
max	72.000000	118.750000	8684.800000

5 visualization of the churn and gender using pie chart

[19]:
```
import plotly.subplots
from plotly.subplots import make_subplots
import plotly.graph_objs as go

g_labels = ['Male', 'Female']
c_labels = ['No', 'Yes']
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':
 ↪'domain'}]])
fig.add_trace(go.Pie(labels=g_labels, values=df['gender'].value_counts(),␣
 ↪name="Gender"),
              1, 1)
fig.add_trace(go.Pie(labels=c_labels, values=df['Churn'].value_counts(),␣
 ↪name="Churn"),
              1, 2)
fig.update_traces(hole=.4, hoverinfo="label+percent+name", textfont_size=16)

fig.update_layout(
    title_text="Gender and Churn Distributions",
    annotations=[dict(text='Gender', x=0.16, y=0.5, font_size=20,␣
 ↪showarrow=False),
                 dict(text='Churn', x=0.84, y=0.5, font_size=20,␣
 ↪showarrow=False)])
fig.show()
```

[20]:
```
df["Churn"][df["Churn"]=="No"].groupby(by=df["gender"]).count()
```

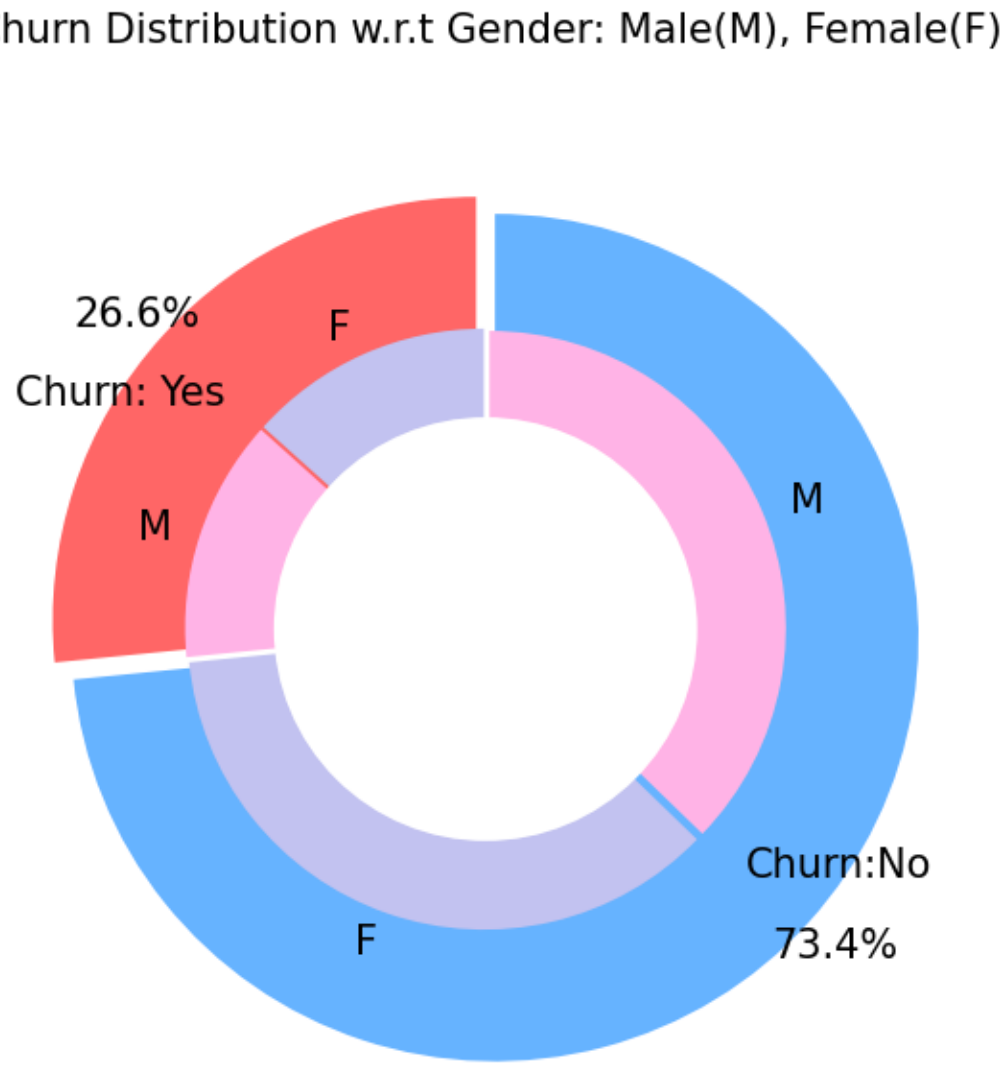
[20]:
```
gender
Female    2544
Male      2619
Name: Churn, dtype: int64
```

[21]:
```
df["Churn"][df["Churn"]=="Yes"].groupby(by=df["gender"]).count()
```

[21]:
```
gender
Female    939
Male      930
Name: Churn, dtype: int64
```

6 Churn Distribution w.r.t Gender: Male(M), Female(F)

[22]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(6, 6))
labels =["Churn: Yes","Churn:No"]
values = [1869,5163]
labels_gender = ["F","M","F","M"]
sizes_gender = [939,930 , 2544,2619]
colors = ['#ff6666', '#66b3ff']
colors_gender = ['#c2c2f0','#ffb3e6', '#c2c2f0','#ffb3e6']
explode = (0.3,0.3)
explode_gender = (0.1,0.1,0.1,0.1)
textprops = {"fontsize":15}
plt.pie(values, labels=labels,autopct='%1.1f%%',pctdistance=1.08,␣
 ↪labeldistance=0.8,colors=colors, startangle=90,frame=True,␣
 ↪explode=explode,radius=10, textprops =textprops, counterclock = True, )
plt.pie(sizes_gender,labels=labels_gender,colors=colors_gender,startangle=90,␣
 ↪explode=explode_gender,radius=7, textprops =textprops, counterclock = True, )
centre_circle = plt.Circle((0,0),5,color='black', fc='white',linewidth=0)
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title( 'Churn Distribution w.r.t Gender: Male(M), Female(F)', fontsize=15,␣
 ↪y=1.1)
plt.axis('equal')
plt.tight_layout()
plt.show()
```



7 Customer contract distribution

[23]:
```
import plotly.express as px

fig = px.histogram(df, x="Churn", color="Contract", barmode="group",␣
 ↪title="<b>Customer contract distribution<b>")
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

8 Payment Method Distribution

[24]:
```
labels = df['PaymentMethod'].unique()
values = df['PaymentMethod'].value_counts()

fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])
fig.update_layout(title_text="<b>Payment Method Distribution</b>")
fig.show()
```

9 Customer Payment Method distribution

[25]:
```
fig = px.histogram(df, x="Churn", color="PaymentMethod", title="<b>Customer␣
 ↪Payment Method distribution w.r.t. Churn</b>")
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

10 Chrun distribution w.r.t. Partners

[26]:
```
color_map = {"Yes": '#FFA15A', "No": '#00CC96'}
fig = px.histogram(df, x="Churn", color="Partner", barmode="group", title="",␣
 ↪color_discrete_map=color_map)
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

11 Chrun distribution w.r.t. Senior Citizen

[27]:
```
color_map = {"Yes": '#00CC96', "No": '#B6E880'}
fig = px.histogram(df, x="Churn", color="SeniorCitizen", title="",␣
 ↪color_discrete_map=color_map)
fig.update_layout(width=700, height=400, bargap=0.1)
fig.show()
```

12 Churn Online Security

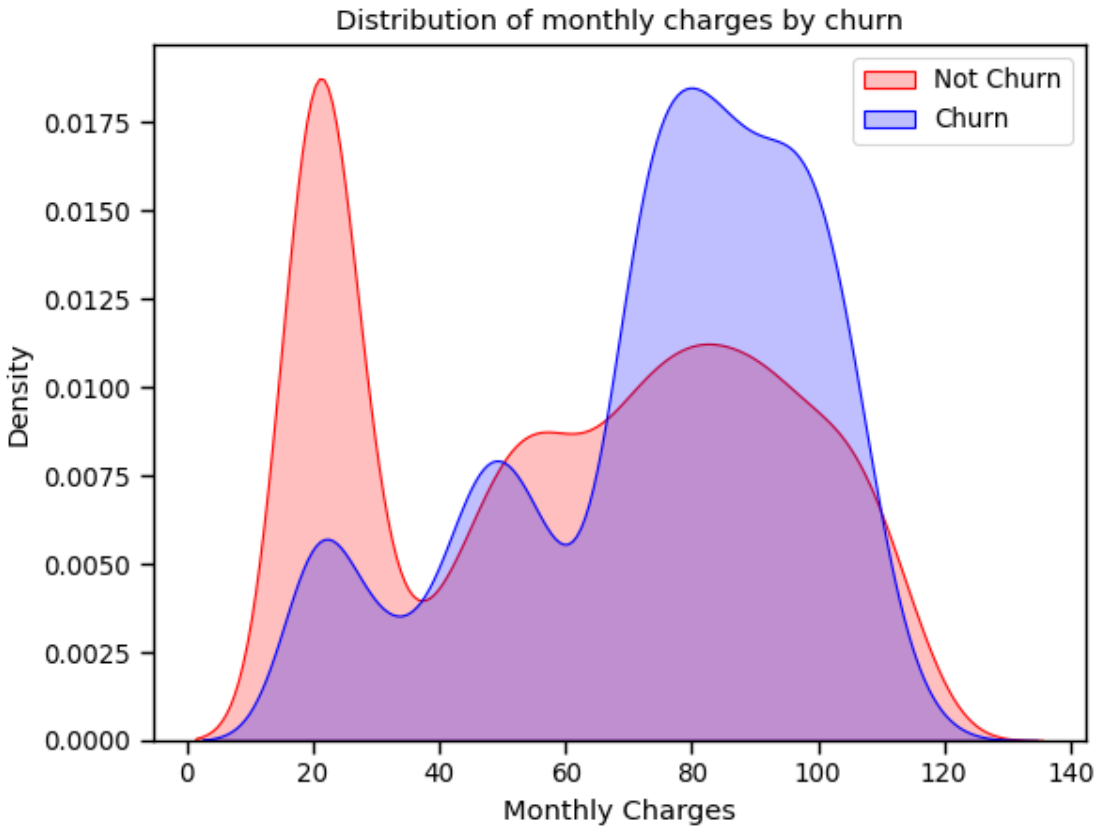
[28]:
```
color_map = {"Yes": "#FF97FF", "No": "#AB63FA"}
fig = px.histogram(df, x="Churn", color="OnlineSecurity", barmode="group",␣
 ↪title="", color_discrete_map=color_map)
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

[29]:
```
color_map = {"Yes": '#FFA15A', "No": '#00CC96'}
fig = px.histogram(df, x="Churn", color="PaperlessBilling",  title="<b>Chrun␣
 ↪distribution w.r.t. Paperless Billing</b>", color_discrete_map=color_map)
```

```
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

[30]:
```
fig = px.histogram(df, x="Churn", color="TechSupport",barmode="group", ␣
↪title="<b>Chrun distribution w.r.t. TechSupport</b>")
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

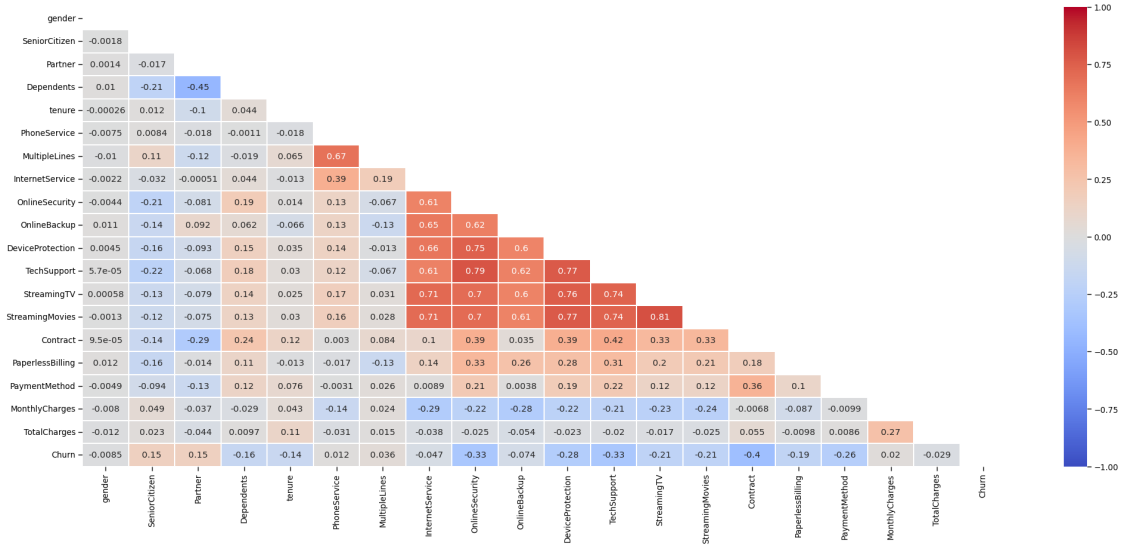
[31]:
```
import seaborn as sns
sns.set_context("paper",font_scale=1.1)
ax = sns.kdeplot(df.MonthlyCharges[(df["Churn"] == 'No') ],
                 color="Red", shade = True);
ax = sns.kdeplot(df.MonthlyCharges[(df["Churn"] == 'Yes') ],
                 ax =ax, color="Blue", shade= True);
ax.legend(["Not Churn","Churn"],loc='upper right');
ax.set_ylabel('Density');
ax.set_xlabel('Monthly Charges');
ax.set_title('Distribution of monthly charges by churn');
```



[32]:
```
fig = px.box(df, x='Churn', y = 'tenure')

fig.update_yaxes(title_text='Tenure (Months)', row=1, col=1)

fig.update_xaxes(title_text='Churn', row=1, col=1)

fig.update_layout(autosize=True, width=750, height=600,
    title_font=dict(size=25, family='Courier'),
    title='<b>Tenure vs Churn</b>',)

fig.show()
```

13 heatmap

[33]:
```
plt.figure(figsize=(25, 10))

corr = df.apply(lambda x: pd.factorize(x)[0]).corr()

mask = np.triu(np.ones_like(corr, dtype=bool))

ax = sns.heatmap(corr, mask=mask, xticklabels=corr.columns, yticklabels=corr.
↪columns, annot=True, linewidths=.2, cmap='coolwarm', vmin=-1, vmax=1)
```



[34]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from sklearn import metrics
from sklearn.metrics import roc_curve
from sklearn.metrics import recall_score, confusion_matrix, precision_score,␣
 ↪f1_score, accuracy_score, classification_report
```

[35]:
```
def object_to_int(dataframe_series):
    if dataframe_series.dtype=='object':
        dataframe_series = LabelEncoder().fit_transform(dataframe_series)
    return dataframe_series
```

[36]:
```
df = df.apply(lambda x: object_to_int(x))
df.head()
```

[36]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	\
0	0	0	1	0	1	0	
1	1	0	0	0	34	1	
2	1	0	0	0	2	1	
3	1	0	0	0	45	0	
4	0	0	0	0	2	1	

	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	\
0	1	0	0	2	
1	0	0	2	0	
2	0	0	2	2	
3	1	0	2	0	
4	0	1	0	0	

	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	\
0	0	0	0	0	0	
1	2	0	0	0	1	
2	0	0	0	0	0	
3	2	2	0	0	1	
4	0	0	0	0	0	

	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn

```
0                1              2           29.85         29.85      0
1                0              3           56.95       1889.50      0
2                1              3           53.85        108.15      1
3                0              0           42.30       1840.75      0
4                1              2           70.70        151.65      1
```

[37]:
```
plt.figure(figsize=(14,7))
df.corr()['Churn'].sort_values(ascending = False)
```

[37]:
```
Churn                1.000000
MonthlyCharges       0.192858
PaperlessBilling     0.191454
SeniorCitizen        0.150541
PaymentMethod        0.107852
MultipleLines        0.038043
PhoneService         0.011691
gender              -0.008545
StreamingTV         -0.036303
StreamingMovies     -0.038802
InternetService     -0.047097
Partner             -0.149982
Dependents          -0.163128
DeviceProtection    -0.177883
OnlineBackup        -0.195290
TotalCharges        -0.199484
TechSupport         -0.282232
OnlineSecurity      -0.289050
tenure              -0.354049
Contract            -0.396150
Name: Churn, dtype: float64

<Figure size 1400x700 with 0 Axes>
```

[38]:
```
X = df.drop(columns = ['Churn'])
y = df['Churn'].values
```

14 TRAINING AND TESTING

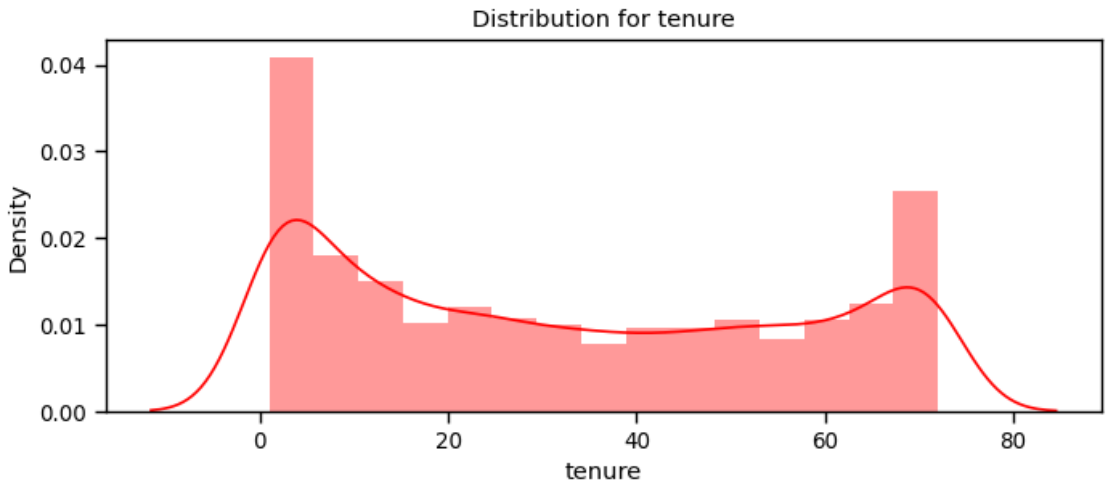
[39]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.30,␣
 ↪random_state = 40, stratify=y)
```

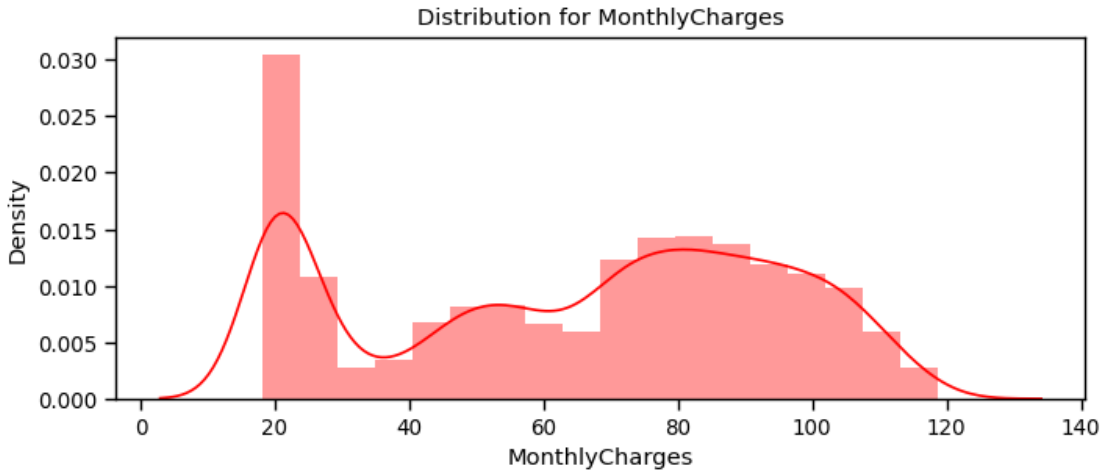
[40]:
```
def distplot(feature, frame, color='r'):
    plt.figure(figsize=(8,3))
    plt.title("Distribution for {}".format(feature))
    ax = sns.distplot(frame[feature], color= color)
```

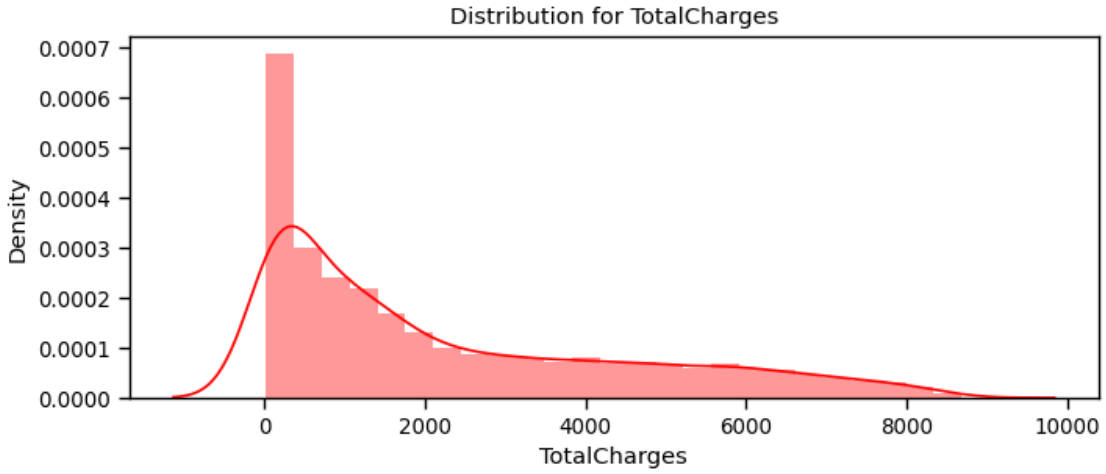
15 DISTRIBUTION

[41]:
```
num_cols = ["tenure", 'MonthlyCharges', 'TotalCharges']
for feat in num_cols: distplot(feat, df)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```







16 Random forest

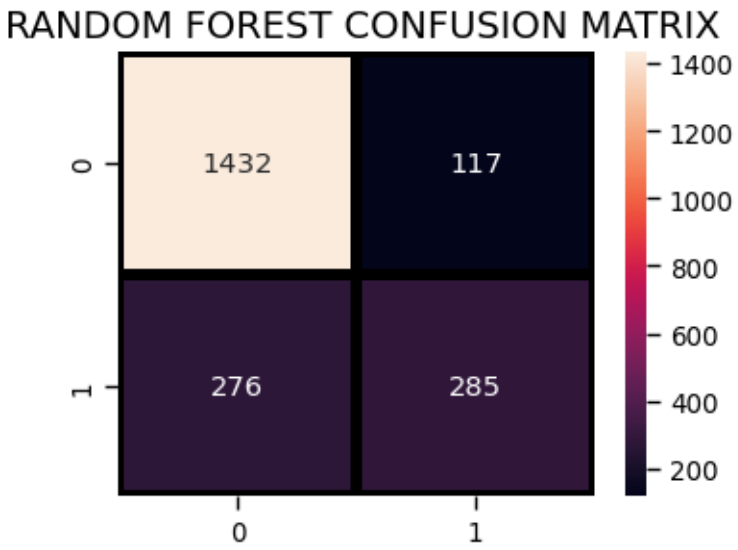
[42]:
```
model_rf = RandomForestClassifier(n_estimators=500 , oob_score = True, n_jobs =␣
↪-1,
                                  random_state =50, max_features = "auto",
                                  max_leaf_nodes = 30)
model_rf.fit(X_train, y_train)

# Make predictions
prediction_test = model_rf.predict(X_test)
print (metrics.accuracy_score(y_test, prediction_test))
```

```
0.8137440758293839
```

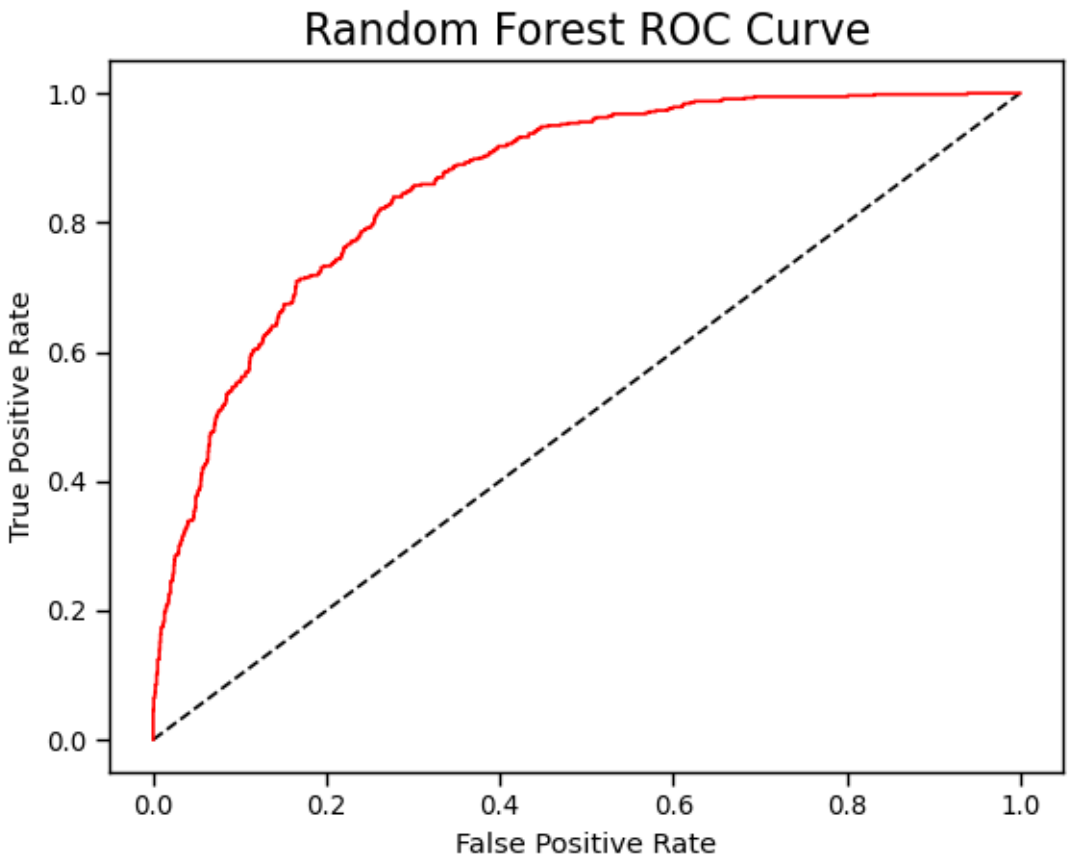
[43]:
```
print(classification_report(y_test, prediction_test))
```

```
              precision    recall  f1-score   support

           0       0.84      0.92      0.88      1549
           1       0.71      0.51      0.59       561

    accuracy                           0.81      2110
   macro avg       0.77      0.72      0.74      2110
weighted avg       0.80      0.81      0.80      2110
```

[44]:
```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test, prediction_test),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title(" RANDOM FOREST CONFUSION MATRIX",fontsize=14)
plt.show()
```



[45]:
```
y_rfpred_prob = model_rf.predict_proba(X_test)[:,1]
fpr_rf, tpr_rf, thresholds = roc_curve(y_test, y_rfpred_prob)
plt.plot([0, 1], [0, 1], 'k--' )
plt.plot(fpr_rf, tpr_rf, label='Random Forest',color = "r")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random Forest ROC Curve',fontsize=16)
plt.show();
```



17 AdaBoostClassifier

[46]:
```
a_model = AdaBoostClassifier()
a_model.fit(X_train,y_train)
a_preds = a_model.predict(X_test)
print("AdaBoost Classifier accuracy")
metrics.accuracy_score(y_test, a_preds)
```

```
AdaBoost Classifier accuracy
```

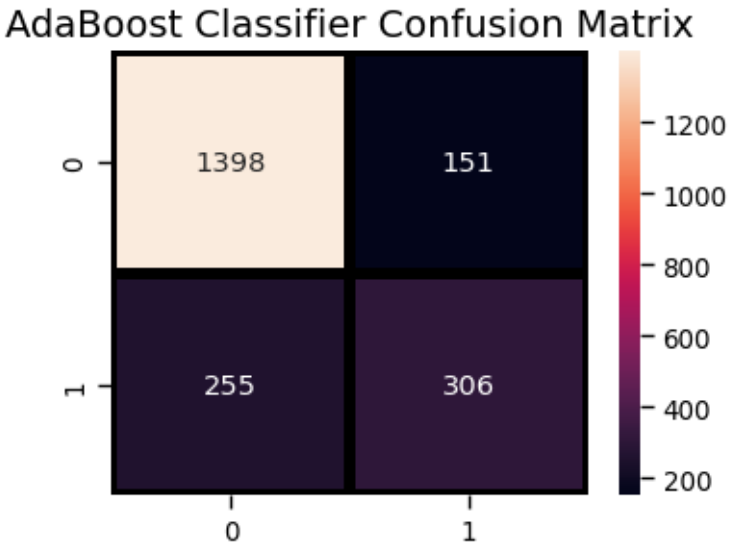
[46]:
```
0.8075829383886256
```

[47]:
```
print(classification_report(y_test, a_preds))
```

```
              precision    recall  f1-score   support

           0       0.85      0.90      0.87      1549
           1       0.67      0.55      0.60       561
```

```
              accuracy                           0.81      2110
             macro avg       0.76      0.72      0.74      2110
          weighted avg       0.80      0.81      0.80      2110
```

[48]:
```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test, a_preds),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("AdaBoost Classifier Confusion Matrix",fontsize=14)
plt.show()
```



18 GradientBoostingClassifier

[49]:
```
gb = GradientBoostingClassifier()
gb.fit(X_train, y_train)
gb_pred = gb.predict(X_test)
print("Gradient Boosting Classifier", accuracy_score(y_test, gb_pred))
```

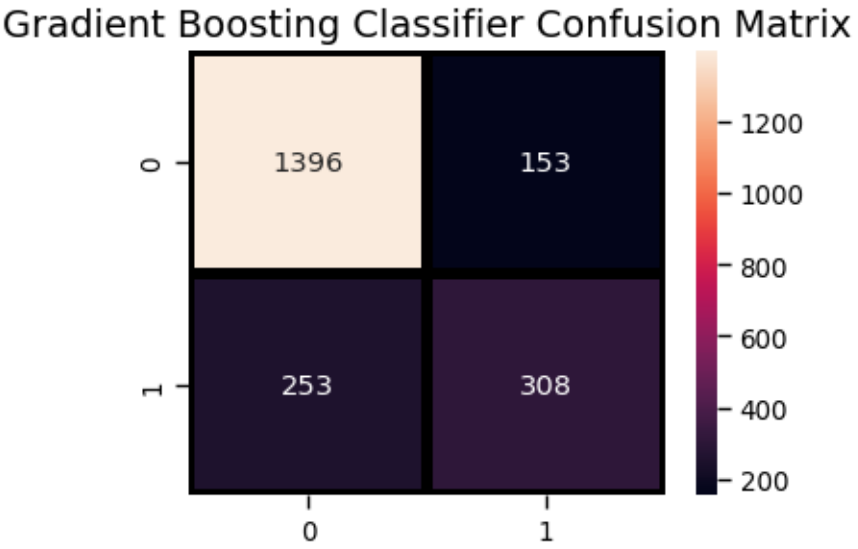
```
Gradient Boosting Classifier 0.8075829383886256
```

[50]:
```
print(classification_report(y_test, gb_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.90      0.87      1549
           1       0.67      0.55      0.60       561
```

```
              accuracy                           0.81      2110
             macro avg       0.76      0.73      0.74      2110
          weighted avg       0.80      0.81      0.80      2110
```

[51]:
```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test, gb_pred),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("Gradient Boosting Classifier Confusion Matrix",fontsize=14)
plt.show()
```



The above code helps us to explain the all model of customer churn prediction in the python here we have derived the 51 input lines each of them have helped in creating the model.

Few of the overall steps is defined as follows:

1 import necessary functions

2 import the csv file {dataset}

3 visualize the missing values

4 Data manipulation

5 visualization of the churn and gender using pie chart

6 Churn Distribution w.r.t Gender: Male(M), Female(F)

7 Customer contract distribution

8 Payment Method Distribution

9 Customer Payment Method distribution

10 Chrun distribution w.r.t. Partners

11 Chrun distribution w.r.t. Senior Citizen

12 Churn Online Security

13 heatmap

14 TRAINING AND TESTING

15 DISTRIBUTIONS

16 Random Forest

17 AdaBoostClassifier

18 GradientBoostingClassifier

In these steps we defined our model and some of hose visualizations we done also in the IBM Cognos dashboard.

College code:4212
Register number:421221243029

The overall document helps to the particular telecom company to analyse the situation in the various times on the model that have been used by the visualization using these types of techniques helps the industry to maintain the consistent customers, focuses on the lagging of the customers and helps to better improvement and the quality of service and also the model that we have been developed have create a massive insights for the telecom company to focus where they were slowly and they were used this model to improve their service as well as the customer service to better business.

churn prediction is an ongoing process, and its effectiveness depends on the quality of data, the choice of features, and the accuracy of the predictive model. Regularly assessing and improving your churn prediction system is essential for retaining customers and ensuring business success.