== ENSEMBLED TECHNIQUES PROJECT==

In [433…

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import time
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
#from sklearn.feature_extraction.text import CountVectorizer  #DT does not take strings as input for the model fit step....
from IPython.display import Image
#import pydotplus as pydot
from sklearn import tree
from os import system
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier,RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier,RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier,RandomForestClassifier
from sklearn import metrics
from sklearn.model_selection import GridSearchCV
%matplotlib inline
```

DOMAIN: Telecom

• **CONTEXT**: A telecom company wants to use their historical customer data and leverage machine learning to predict behaviour in an attempt to retain customers. The end goal is to develop focused customer retention programs

• **DATA DESCRIPTION**: Each row represents a customer, each column contains customer's attributes described on the column Metadata. The data set includes information about:

• Customers who left within the last month – the column is called Churn • Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies • Customer account information – how long they've been a customer, contract, payment method, paperless billing, monthly charges, and total charges • Demographic info about customers – gender, age range, and if they have partners and dependents

• **PROJECT OBJECTIVE**: The objective, as a data scientist hired by the telecom company, is to build a model that will help to identify the potential customers who have a higher probability to churn. This will help the company to understand the pain points and patterns of customer churn and will increase the focus on strategising customer retention.

• STEPS AND TASK [60 Marks]:

1. Data Understanding & Exploration: [5 Marks]

1 A. Read 'TelcomCustomer-Churn_1.csv' as a DataFrame and assign it to a variable. [1 Mark]

In [434…
```
df1=pd.read_csv("C:\\Users\\CHIRAG\\Desktop\\Greatlearning- Projects\\AIML project - enssembled technique\\TelcomCustomer-Churn_1
```

In [435…
```
df1.head()
```

Out[435]:

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

In [436…
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 10 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
dtypes: int64(2), object(8)
memory usage: 550.4+ KB
```

In [437…

```
df1.shape
```

Out[437]:

```
(7043, 10)
```

1 B. Read 'TelcomCustomer-Churn_2.csv' as a DataFrame and assign it to a variable. [1 Mark]

In [438…

```
df2=pd.read_csv("C:\\Users\\CHIRAG\\Desktop\\Greatlearning- Projects\\AIML project - enssembled technique\\TelcomCustomer-Churn_2
```

In [439…

```
df2.head()
```

Out[439]:

	customerID	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharg
0	7590-VHVEG	Yes	No	No	No	No	Month-to-month	Yes	Electronic check	29.
1	5575-GNVDE	No	Yes	No	No	No	One year	No	Mailed check	56.
2	3668-QPYBK	Yes	No	No	No	No	Month-to-month	Yes	Mailed check	53.
3	7795-CFOCW	No	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.
4	9237-HQITU	No	No	No	No	No	Month-to-month	Yes	Electronic check	70.

In [440…

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   OnlineBackup      7043 non-null   object 
 2   DeviceProtection  7043 non-null   object 
 3   TechSupport       7043 non-null   object 
 4   StreamingTV       7043 non-null   object 
 5   StreamingMovies   7043 non-null   object 
 6   Contract          7043 non-null   object 
 7   PaperlessBilling  7043 non-null   object 
 8   PaymentMethod     7043 non-null   object 
 9   MonthlyCharges    7043 non-null   float64
 10  TotalCharges      7043 non-null   object 
 11  Churn             7043 non-null   object 
dtypes: float64(1), object(11)
memory usage: 660.4+ KB
```

In [441…
```
df2.shape
```

Out[441]:
```
(7043, 12)
```

1 C. Merge both the DataFrames on key 'customerID' to form a single DataFrame [2 Mark]

In [442…
```
df=pd.merge(df1,df2,how='outer',on='customerID')
```

In [443…
```
df.head()
```

Out[443]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	T
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	

5 rows × 21 columns

In [444…
```
df.shape
```

Out[444]:
```
(7043, 21)
```

1 D. Verify if all the columns are incorporated in the merged DataFrame by using simple comparison Operator in Python. [1 Marks]

In [445…
```
df1.columns
```

Out[445]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity'],
      dtype='object')
```

In [446…
```
df2.columns
```

Out[446]:
```
Index(['customerID', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

In [447…
```
print("The lenght of column in Df1 is",len(df1.columns))
print("The length of column in Df2 is",len(df2.columns))
```

```
The lenght of column in Df1 is 10
The length of column in Df2 is 12
```

In [448…
```
print("lets check all columns are incorporated or not from df1 to df")
x=0
for i in df1.columns:
    for j in range(len(df1.columns)):
        if i==df.columns[j]:
            x=x+1
            print(i)
if(x==len(df1.columns)):
    print("all columns are present from dataframe df1 and that is",x)
    print(" \n")
#similarly will check for dataframe 2
print("lets check all columns are incorporated or not from df2 to df")
y=0
for k in df2.columns:
    for l in range(len(df.columns)):
        if k==df.columns[l-x]:
            y=y+1
            print(k)
if(y==len(df2.columns)):
```

```
    print("all columns are present from dataframe df2 and that is ",y)
```

```
lets check all columns are incorporated or not from df1 to df
customerID
gender
SeniorCitizen
Partner
Dependents
tenure
PhoneService
MultipleLines
InternetService
OnlineSecurity
all columns are present from dataframe df1 and that is 10


lets check all columns are incorporated or not from df2 to df
customerID
OnlineBackup
DeviceProtection
TechSupport
StreamingTV
StreamingMovies
Contract
PaperlessBilling
PaymentMethod
MonthlyCharges
TotalCharges
Churn
all columns are present from dataframe df2 and that is  12
```

2. Data Cleaning & Analysis: [15 Marks]

2 A. Impute missing/unexpected values in the DataFrame. [2 Marks]

In [449…

```
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
```

In [450…

```
print(df.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        0
Churn               0
dtype: int64
```

- so no missing value present in dataframe let see other type of data which is not require for analysis
- here we can see that from dataset information that type of "Total Charges" is object but it has to be in numeric so there must be in appropriate data

In [451…
```
df["TotalCharges"] = (pd.to_numeric(df["TotalCharges"],errors="coerce"))
```

In [452…
```
df["TotalCharges"].dtype
```

Out[452]:
```
dtype('float64')
```

ype('float64') 1

- Note : we convrted TotalCharges to numric but eleven values could not
- so we inplaced nan values

In [453…
```
df[np.isnan(df["TotalCharges"])]
```

Out[453]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtectio
488	4472-LVYGI	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	...	Ye
753	3115-CZMZD	Male	0	No	Yes	0	Yes	No	No	No internet service	...	No interne servic
936	5709-LVOEQ	Female	0	Yes	Yes	0	Yes	No	DSL	Yes	...	Ye
1082	4367-NUYAO	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	...	No interne servic
1340	1371-DWPAZ	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	...	Ye
3331	7644-OMVMY	Male	0	Yes	Yes	0	Yes	No	No	No internet service	...	No interne servic
3826	3213-VVOLG	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	...	No interne servic
4380	2520-SGTTA	Female	0	Yes	Yes	0	Yes	No	No	No internet service	...	No interne servic
5218	2923-ARZLG	Male	0	Yes	Yes	0	Yes	No	No	No internet service	...	No interne servic
6670	4075-WKNIU	Female	0	Yes	Yes	0	Yes	Yes	DSL	No	...	Ye
6754	2775-SEFEE	Male	0	No	Yes	0	Yes	Yes	DSL	Yes	...	N

11 rows × 21 columns

- It can also be noted that the Tenure column is 0 for these entries even though the MonthlyCharges column is not empty.
- Let's see if there are any other 0 values in the tenure column.

In [454...

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

Out[454]:
```
Index([488, 753, 936, 1082, 1340, 3331, 3826, 4380, 5218, 6670, 6754], dtype='int64')
```

- there are no additional missing values in the Tenure column.
- Let's delete the rows with missing values in Tenure columns since there are only 11 rows and deleting them will not affect the data.

In [455…
```
df.drop(labels=df[df['tenure'] == 0].index, axis=0, inplace=True)
df[df['tenure'] == 0].index
```

Out[455]:
```
Index([], dtype='int64')
```

In [456…
```
df.isnull().sum()
```

Out[456]:
```
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
```

- So there are no missing values now

2 B. Make sure all the variables with continuous values are of 'Float' type. [2 Marks]

- [For Example: MonthlyCharges, TotalCharges]

In [457…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7032 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7032 non-null   object 
 1   gender            7032 non-null   object 
 2   SeniorCitizen     7032 non-null   int64  
 3   Partner           7032 non-null   object 
 4   Dependents        7032 non-null   object 
 5   tenure            7032 non-null   int64  
 6   PhoneService      7032 non-null   object 
 7   MultipleLines     7032 non-null   object 
 8   InternetService   7032 non-null   object 
 9   OnlineSecurity    7032 non-null   object 
 10  OnlineBackup      7032 non-null   object 
 11  DeviceProtection  7032 non-null   object 
 12  TechSupport       7032 non-null   object 
 13  StreamingTV       7032 non-null   object 
 14  StreamingMovies   7032 non-null   object 
 15  Contract          7032 non-null   object 
 16  PaperlessBilling  7032 non-null   object 
 17  PaymentMethod     7032 non-null   object 
 18  MonthlyCharges    7032 non-null   float64
 19  TotalCharges      7032 non-null   float64
 20  Churn             7032 non-null   object 
dtypes: float64(2), int64(2), object(17)
memory usage: 1.2+ MB
```

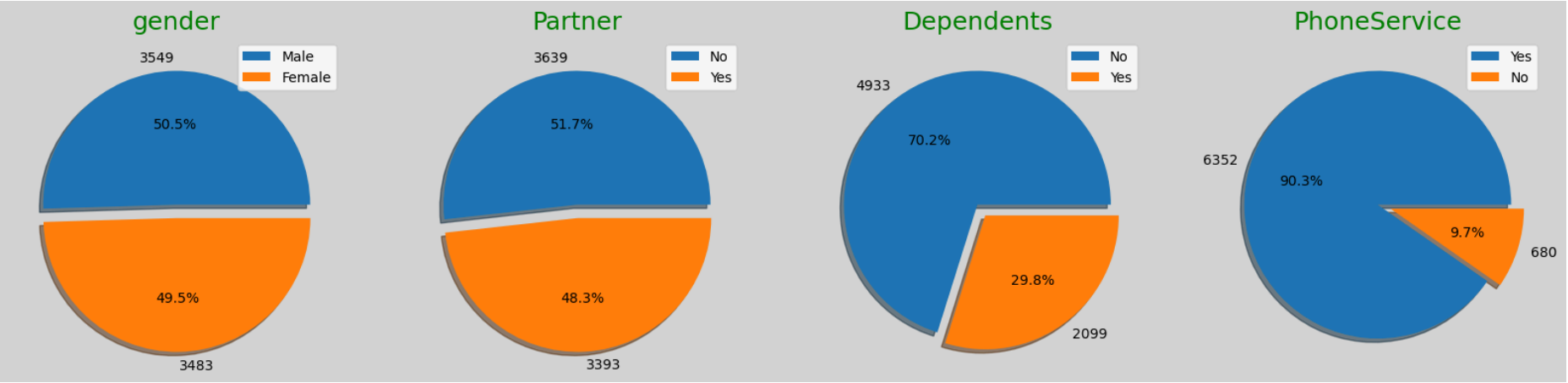
- monthlycharges and totalcharges are float type
- and senior Citizens can not be float type
- rests are has to be object type only in data set

note: for data conversion is done later part of the code rest float coversion for Monthly charges and total charges are done

2 C. Create a function that will accept a DataFrame as input and return pie-charts for all the appropriate Categorical features. Clearly show percentage distribution in the pie-chart. [4 Marks]

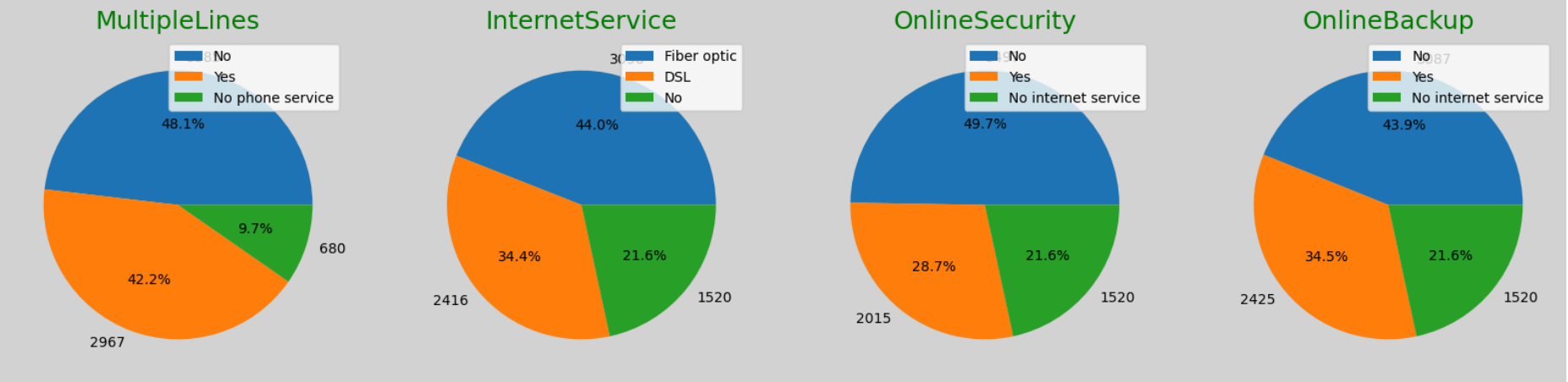
In [458...

```
categ_columns = df.select_dtypes("object").columns[1:-1]

fig, axes = plt.subplots(1, 4, figsize=(20, 12), facecolor="lightgray")

for i, column in enumerate(categ_columns[:4]):
    ax = axes[i]
    data = df[column].value_counts()
    ax.pie(data, labels=data.values,autopct="%1.1f%%",
           explode=[0,0.1],shadow=True)
    ax.set_title(column,color="green",size=18)
    ax.legend(data.index)
```



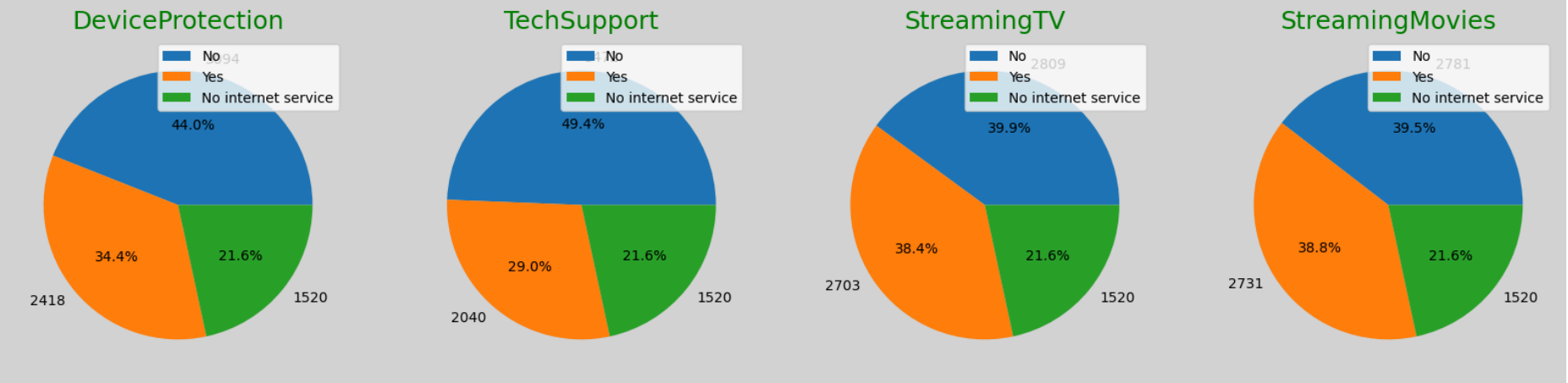
In [459...

```
fig, axes = plt.subplots(1, 4, figsize=(20, 12), facecolor="lightgray")
for i, column in enumerate(categ_columns[4:8]):
    ax = axes[i]
    data = df[column].value_counts()
    ax.pie(data, labels=data.values,autopct="%1.1f%%")
    ax.set_title(column,color="green",size=18)
    ax.legend(data.index,loc="best")
```



In [460…

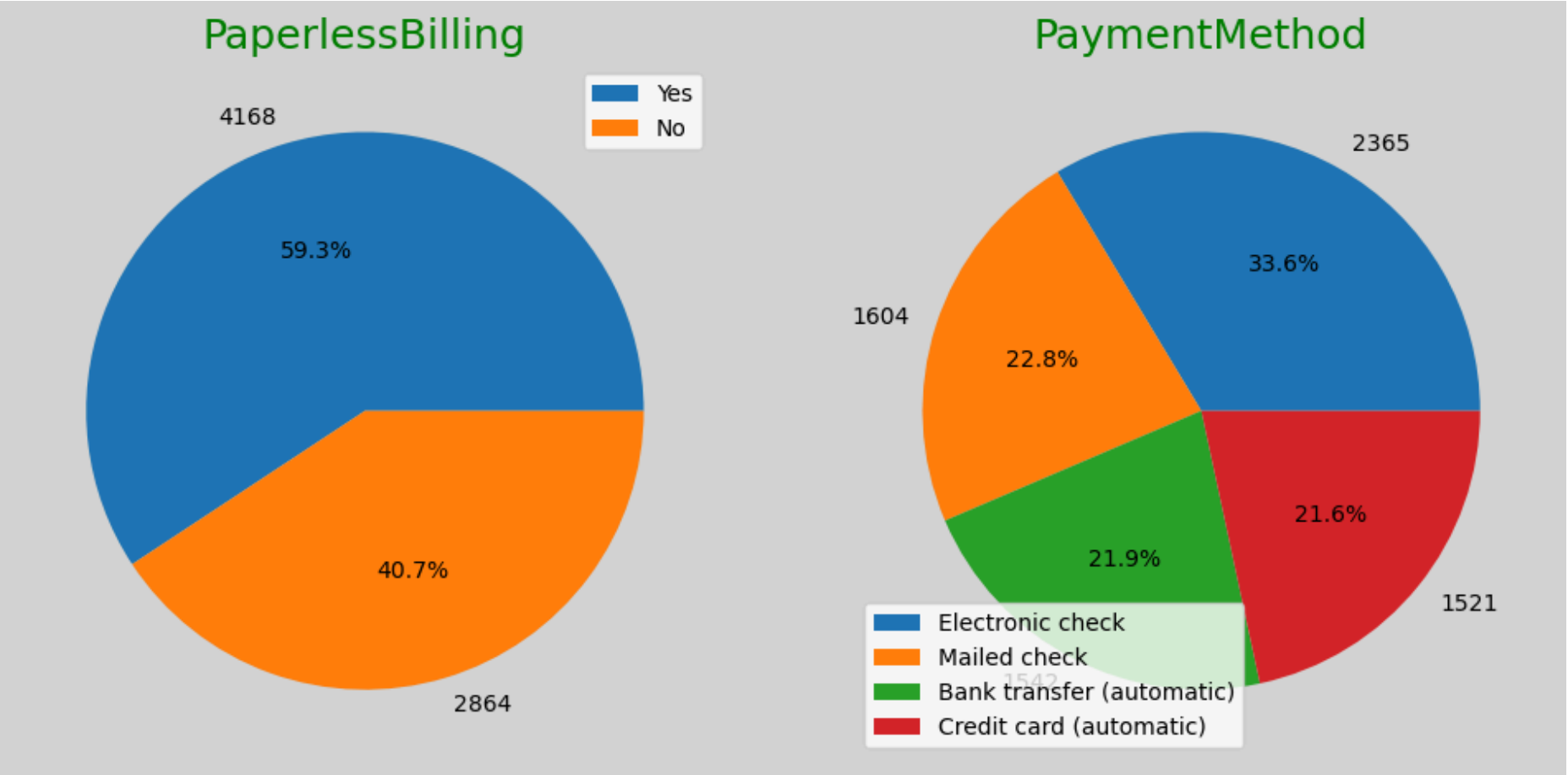
```
fig, axes = plt.subplots(1, 4, figsize=(20, 12), facecolor="lightgray")

for i, column in enumerate(categ_columns[8:12]):
    ax = axes[i]
    data = df[column].value_counts()
    ax.pie(data, labels=data.values,autopct="%1.1f%%")
    ax.set_title(column,color="green",size=18)
    ax.legend(data.index)
```



In [461…

```
fig, axes = plt.subplots(1, 2, figsize=(12, 12), facecolor="lightgray")

for i, column in enumerate(categ_columns[13:]):
    ax = axes[i]
    data = df[column].value_counts()
```

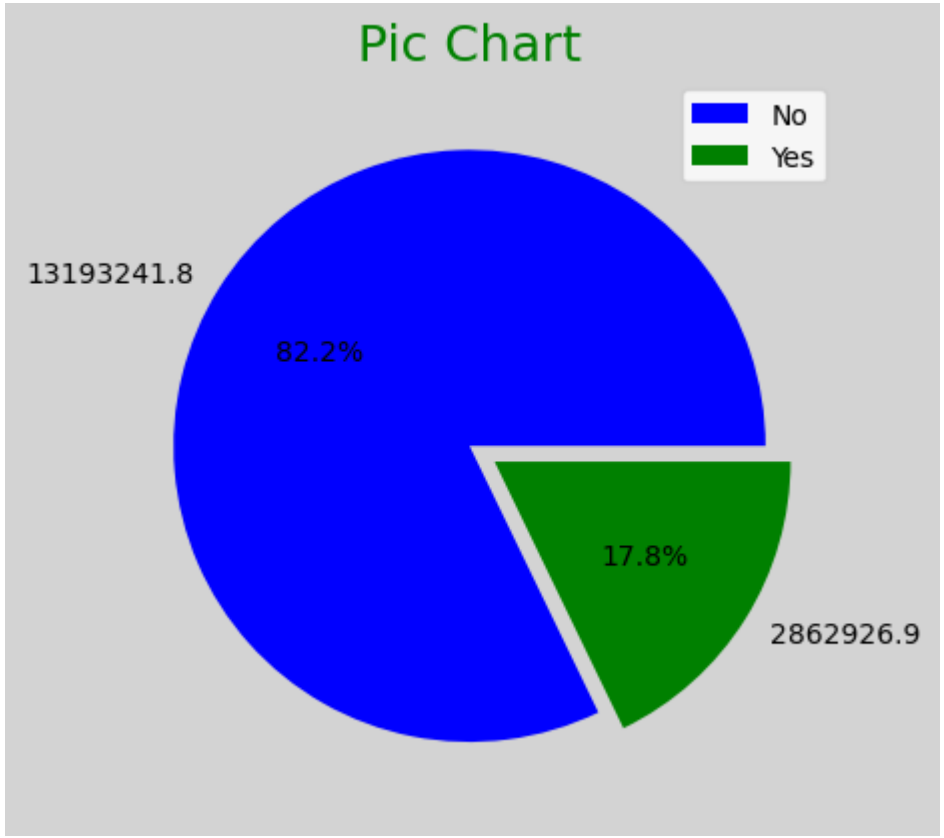
```
    ax.pie(data, labels=data.values,autopct="%1.1f%%")
    ax.set_title(column,color="green",size=18)
    ax.legend(data.index)
```



pie chart of sum Total charges of each classes

In [462…

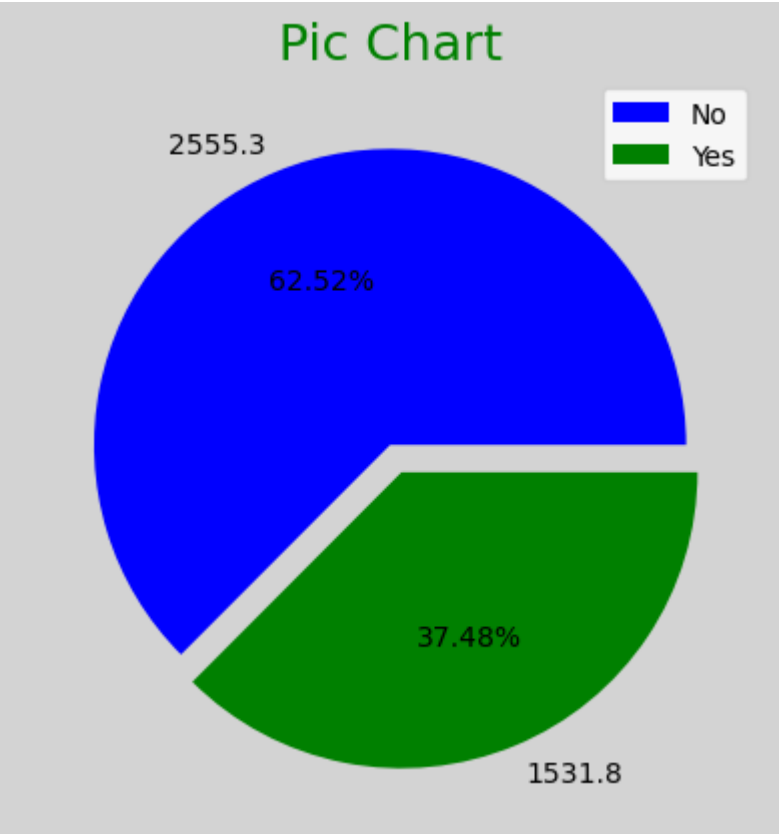
```
fig , ax = plt.subplots(facecolor="lightgray")
data = df.groupby(by="Churn")["TotalCharges"].sum()

ax.pie(data,labels=data.values,explode=[0,0.1],autopct="%1.1f%%",
       colors=["blue","green"])
ax.set_title("Pic Chart",color="green",size=18)
ax.legend(data.index);
```



Pie chart of mean of total charges for each classes

In [463…

```
fig , ax = plt.subplots(facecolor="lightgray")
data = df.groupby(by="Churn")["TotalCharges"].mean()

ax.pie(data,labels=data.values.round(1),explode=[0,0.1],autopct="%1.2f%%",
       colors=["blue","green"])
ax.set_title("Pic Chart",color="green",size=18)
ax.legend(data.index,loc=1);
```



D. Share insights for Q2.c. [2 Marks]

insight

- the are total 3549 males(50.5%) and female are 3483(49.5%)
- 3639 (51.7%) outof total custustomer has no partner and 3393 ie (48.3%) has partners.
- 4933 (70.2%) of total customer (7032) observation has dependents
- 6352 (90.3%) has phose services
- 2967 (42.2%) has multiple lines and rest has no multiple line that includes with no phone services
- 2416 (34.4%) has DSl, 1520(21.6%) has no internet service, rest are using fiber optics for internet
- 2425 (34.5%) uses online back up rest are not inclding those who dont have internet services
- 2418 (34.4%) has Device protection and 44% has no protection plan

- 2703(38.4%) uses services for streaming TV
- 2732(38.8%) uses streaming Movies services
- 2864(40.7%) uses hardpaper billing services and other are paperless billing
- payment are done using electtronic check,mailed check,bank transfer(auto),and c

E. Encode all the appropriate Categorical features with the best suitable approach. [2 Marks]

- I want to print unique values of each columns which are obejct
- we see some of data like (No phone service,No internet service)
- I decided to replace them with NO

In [464…
```
for i in df.columns[1:]:
    if df[i].dtype == 'object':
        print(i,df[i].value_counts().index.values)
```

```
gender ['Male' 'Female']
Partner ['No' 'Yes']
Dependents ['No' 'Yes']
PhoneService ['Yes' 'No']
MultipleLines ['No' 'Yes' 'No phone service']
InternetService ['Fiber optic' 'DSL' 'No']
OnlineSecurity ['No' 'Yes' 'No internet service']
OnlineBackup ['No' 'Yes' 'No internet service']
DeviceProtection ['No' 'Yes' 'No internet service']
TechSupport ['No' 'Yes' 'No internet service']
StreamingTV ['No' 'Yes' 'No internet service']
StreamingMovies ['No' 'Yes' 'No internet service']
Contract ['Month-to-month' 'Two year' 'One year']
PaperlessBilling ['Yes' 'No']
PaymentMethod ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
Churn ['No' 'Yes']
```

In [465…
```
cols= ['MultipleLines','OnlineSecurity','OnlineBackup','DeviceProtection','TechSupport','StreamingTV','StreamingMovies']
```

```
for i in cols:
    df[i].replace('No phone service','No',inplace=True)
    df[i].replace('No internet service','No',inplace=True)
```

In [466…

```
for i in df.columns[1:]:
    if df[i].dtypes == 'object':
        print(i,df[i].value_counts().index.values)
```

```
gender ['Male' 'Female']
Partner ['No' 'Yes']
Dependents ['No' 'Yes']
PhoneService ['Yes' 'No']
MultipleLines ['No' 'Yes']
InternetService ['Fiber optic' 'DSL' 'No']
OnlineSecurity ['No' 'Yes']
OnlineBackup ['No' 'Yes']
DeviceProtection ['No' 'Yes']
TechSupport ['No' 'Yes']
StreamingTV ['No' 'Yes']
StreamingMovies ['No' 'Yes']
Contract ['Month-to-month' 'Two year' 'One year']
PaperlessBilling ['Yes' 'No']
PaymentMethod ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
Churn ['No' 'Yes']
```

- now we dont need column 'customer ID ' so droping that

In [467…

```
df.drop("customerID",axis='columns',inplace= True)
df.head()
```

Out[467]:

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

- so we can see that customerID column is removed

2 E. Encode all the appropriate Categorical features with the best suitable approach. [2 Marks]

ENCODING

In [468…

```
categorical=df.select_dtypes("object")
categorical.head()
```

Out[468]:

	gender	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingT
0	Female	Yes	No	No	No	DSL	No	Yes	No	No	N
1	Male	No	No	Yes	No	DSL	Yes	No	Yes	No	N
2	Male	No	No	Yes	No	DSL	Yes	Yes	No	No	N
3	Male	No	No	No	No	DSL	Yes	No	Yes	Yes	N
4	Female	No	No	Yes	No	Fiber optic	No	No	No	No	N

In [473…

```
from sklearn.preprocessing import StandardScaler,OrdinalEncoder
encoder=OrdinalEncoder().fit(categorical)
encoded=encoder.transform(categorical)
encoder.categories_
```

Out[473]:

```
[array(['Female', 'Male'], dtype=object),
 array(['No', 'Yes'], dtype=object),
 array(['No', 'Yes'], dtype=object),
 array(['No', 'Yes'], dtype=object),
 array(['No', 'Yes'], dtype=object),
 array(['DSL', 'Fiber optic', 'No'], dtype=object),
 array(['No', 'Yes'], dtype=object),
 array(['No', 'Yes'], dtype=object),
 array(['No', 'Yes'], dtype=object),
 array(['No', 'Yes'], dtype=object),
 array(['No', 'Yes'], dtype=object),
 array(['No', 'Yes'], dtype=object),
 array(['Month-to-month', 'One year', 'Two year'], dtype=object),
 array(['No', 'Yes'], dtype=object),
 array(['Bank transfer (automatic)', 'Credit card (automatic)',
        'Electronic check', 'Mailed check'], dtype=object),
 array(['No', 'Yes'], dtype=object)]
```

In [474…
```
number = df.select_dtypes("number").reset_index(drop=True)
```

In [475…
```
cate = pd.DataFrame(encoded.astype("int64"),columns=categorical.columns).reset_index(drop=True)
```

In [476…
```
df_final = pd.concat([number,cate],axis=1)
df_final.head()
```

Out[476]:

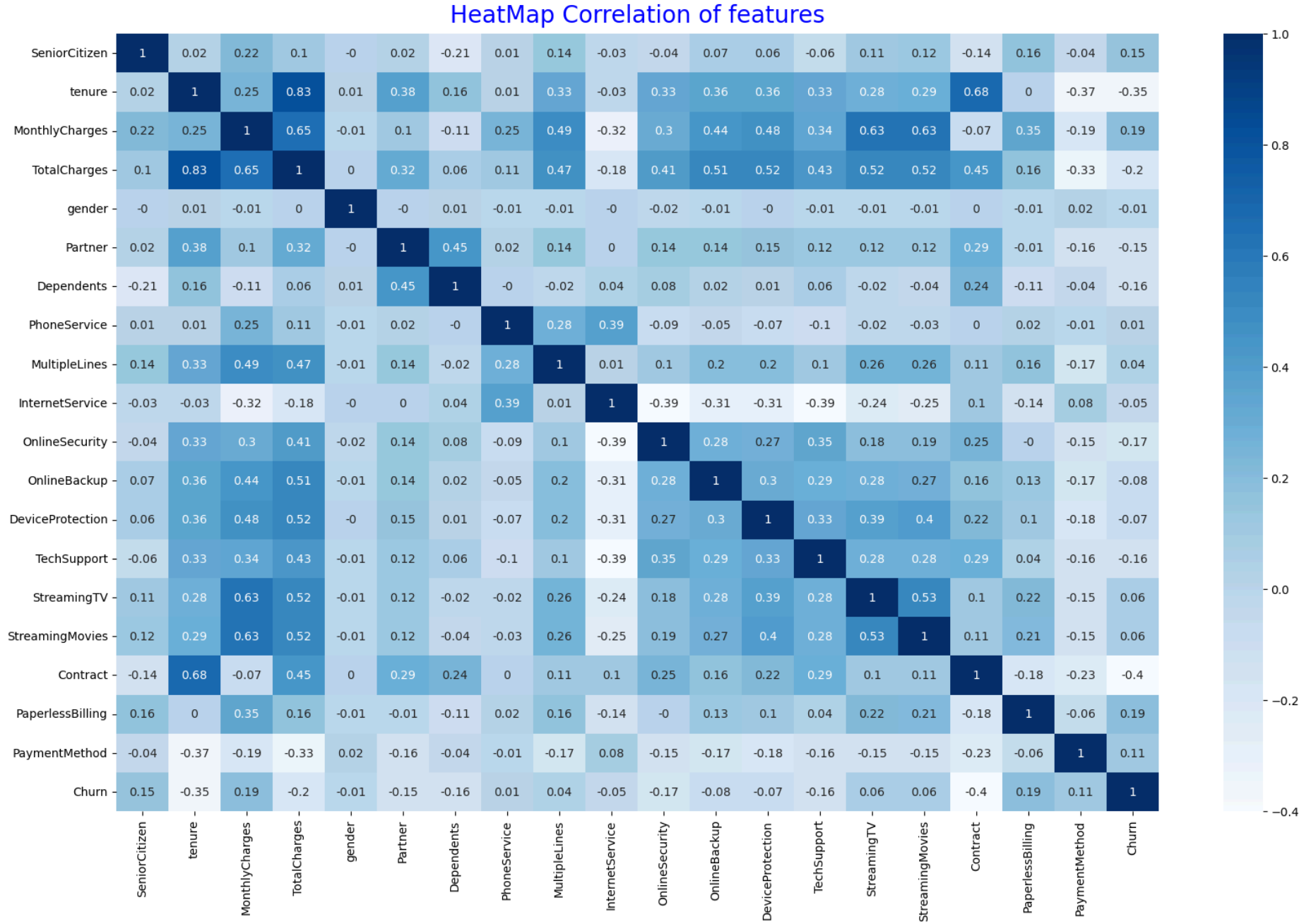
	SeniorCitizen	tenure	MonthlyCharges	TotalCharges	gender	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	Onli
0	0	1	29.85	29.85	0	1	0	0	0	0	0	
1	0	34	56.95	1889.50	1	0	0	1	0	0	1	
2	0	2	53.85	108.15	1	0	0	1	0	0	1	
3	0	45	42.30	1840.75	1	0	0	0	0	0	1	
4	0	2	70.70	151.65	0	0	0	1	0	1	0	

In [477…
```
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7032 entries, 0 to 7031
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   SeniorCitizen     7032 non-null   int64  
 1   tenure            7032 non-null   int64  
 2   MonthlyCharges    7032 non-null   float64
 3   TotalCharges      7032 non-null   float64
 4   gender            7032 non-null   int64  
 5   Partner           7032 non-null   int64  
 6   Dependents        7032 non-null   int64  
 7   PhoneService      7032 non-null   int64  
 8   MultipleLines     7032 non-null   int64  
 9   InternetService   7032 non-null   int64  
 10  OnlineSecurity    7032 non-null   int64  
 11  OnlineBackup      7032 non-null   int64  
 12  DeviceProtection  7032 non-null   int64  
 13  TechSupport       7032 non-null   int64  
 14  StreamingTV       7032 non-null   int64  
 15  StreamingMovies   7032 non-null   int64  
 16  Contract          7032 non-null   int64  
 17  PaperlessBilling  7032 non-null   int64  
 18  PaymentMethod     7032 non-null   int64  
 19  Churn             7032 non-null   int64  
dtypes: float64(2), int64(18)
memory usage: 1.1 MB
```

In [479…

```
fig , ax = plt.subplots(figsize=(20,12))
sns.heatmap(df_final.corr().round(2),annot=True,ax=ax,cmap="Blues")
ax.set_title("HeatMap Correlation of features",size=20,color="Blue",pad=10);
```



2 F. Split the data into 80% train and 20% test. [1 Marks]

Selecting features and target

In [480…
```
scaler = StandardScaler()

df_final[["tenure","MonthlyCharges","TotalCharges"]] = scaler.fit_transform(df_final[["tenure","MonthlyCharges","TotalCharges"]])
```

In [481…
```
X = df_final.drop("Churn",axis="columns")
y = df_final["Churn"]
```

In [482…
```
print("shape of featuers:",X.shape)
print("shape of target",y.shape)
```

```
shape of featuers: (7032, 19)
shape of target (7032,)
```

Split

In [483…
```
from sklearn.model_selection import train_test_split,GridSearchCV,KFold
X_test,X_train,y_test,y_train=train_test_split(X,y,random_state=1,test_size=0.20,stratify=y)
```

In [484…
```
y_test.value_counts()
```

Out[484]:
```
Churn
0    4130
1    1495
Name: count, dtype: int64
```

In [485…
```
df_final['Churn'].value_counts()
```

Out[485]:
```
Churn
0    5163
1    1869
Name: count, dtype: int64
```

G. Normalize/Standardize the data with the best suitable approach. [2 Marks]

In [486…
```
scaler = StandardScaler()

df_final[["tenure","MonthlyCharges","TotalCharges"]] = scaler.fit_transform(df_final[["tenure","MonthlyCharges","TotalCharges"]])
```

3. Model building and performance improvement : [40 Marks]

3 A. Train a model using Decision tree and check the performance of the model on train and test data (4 marks)

Build Decision Tree Model

We will build our model using the DecisionTreeClassifier function. Using default 'gini' criteria to split. Other option include 'entropy'.

- **But since we are building multiple models in this project so making common function**
- **for performance matrics**
- **as well as to print difference perforamnce parameterss such as acuuracy,recall, precision etc**

In [487…

```
def make_confusion_matrix(model,y_actual,labels=[1,0]):
    ''' model : classifier to predict value of X
        y_actual : ground truth
    '''
    y_predict = model.predict(X_test)
    cm=metrics.confusion_matrix(y_actual,y_predict,labels=[0,1])
    df_cm=pd.DataFrame(cm,index=[i for i in["Actual-No","Actual - yes"]],columns=[i for i in ['preedict -No','Predicted -yes']])
    group_counts = ["{0:0.0f}".format(value) for value in cm.flatten()]
    group_percentages=["{0:0.2%}".format(value) for value in cm.flatten()/np.sum(cm)]
    labels = [f"{v1}\n{v2}"for v1,v2 in zip(group_counts,group_percentages)]
    labels = np.asarray(labels).reshape(2,2)
    plt.figure(figsize = (10,7))
    sns.heatmap(df_cm,annot=labels,fmt='')
    plt.ylabel("True label")
    plt.xlabel('Predicted label')
```

In [488…

```
def get_metrics_score(model,flag=True):
    '''model : classifier to predict values of x'''
    ''' defining an empty list to store train and test result'''
    score_list=[]

    '''predict on train and tests'''
    pred_train=model.predict(X_train)
    pred_test=model.predict(X_test)
```

```
    #Accuracy of the model
    train_acc = model.score(X_train,y_train)
    test_acc = model.score(X_test,y_test)

    #recall of the model
    train_recall = metrics.recall_score(y_train,pred_train)
    test_recall = metrics.recall_score(y_test,pred_test)

    # precision of the model
    train_precision = metrics.precision_score(y_train,pred_train)
    test_precision = metrics.precision_score(y_test,pred_test)

    score_list.extend((train_acc,test_acc,train_recall,test_recall,train_precision,test_precision))
    # if the flag is set to true then only the following  print statement will be displayed, the default value is
    if flag == True:
        print("accuracy on training set : ",model.score(X_train,y_train))
        print("accuracy on test set : ",model.score(X_test,y_test))
        print("Recall on training set : ",metrics.recall_score(y_train,pred_train))
        print("Recall on test set : ",metrics.recall_score(y_test,pred_test))
        print("Precision on training set : ",metrics.precision_score(y_train,pred_train))
        print("Precision on test set : ",metrics.precision_score(y_test,pred_test))

    return score_list # returning thr list with train and test scores
```

let us develope a model without setting max_depth parameter of decision tree

In [489…
```
dTree = DecisionTreeClassifier(criterion = 'gini', random_state=1)
dTree.fit(X_train, y_train)
```

Out[489]:
```
DecisionTreeClassifier(random_state=1)
```

In [490…
```
dTree_score=get_metrics_score(dTree)
```

```
accuracy on training set :  1.0
accuracy on test set :  0.7171555555555555
Recall on training set :  1.0
Recall on test set :  0.4916387959866221
Precision on training set :  1.0
Precision on test set :  0.46934865900383144
```

let us develope a model with setting max_depth =3 parameter of decision tree

Reducing over fitting (Regularization)

In [491…
```
dTree_2 = DecisionTreeClassifier(criterion = 'gini', random_state=1,max_depth=3)
dTree_2.fit(X_train, y_train)
```

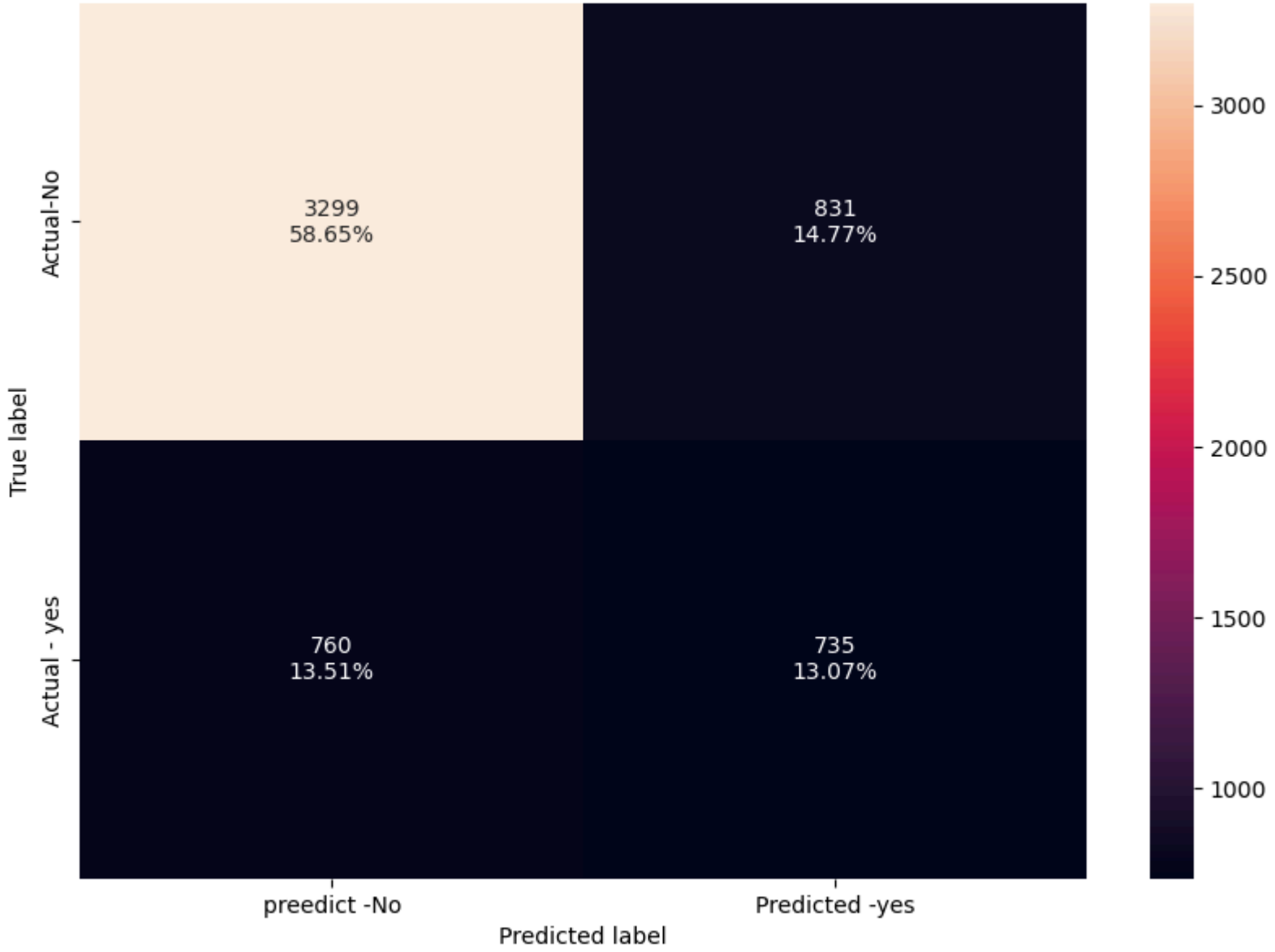
Out[491]:
```
DecisionTreeClassifier(max_depth=3, random_state=1)
```

In [492…
```
dTree_2_score=get_metrics_score(dTree_2)
```

```
accuracy on training set :  0.7803837953091685
accuracy on test set :  0.7662222222222222
Recall on training set :  0.3048128342245989
Recall on test set :  0.28561872909699
Precision on training set :  0.6993865030674846
Precision on test set :  0.6335311572700296
```

In [493…
```
print("the matrix without setting max depth")
print(make_confusion_matrix(dTree,y_test))
```

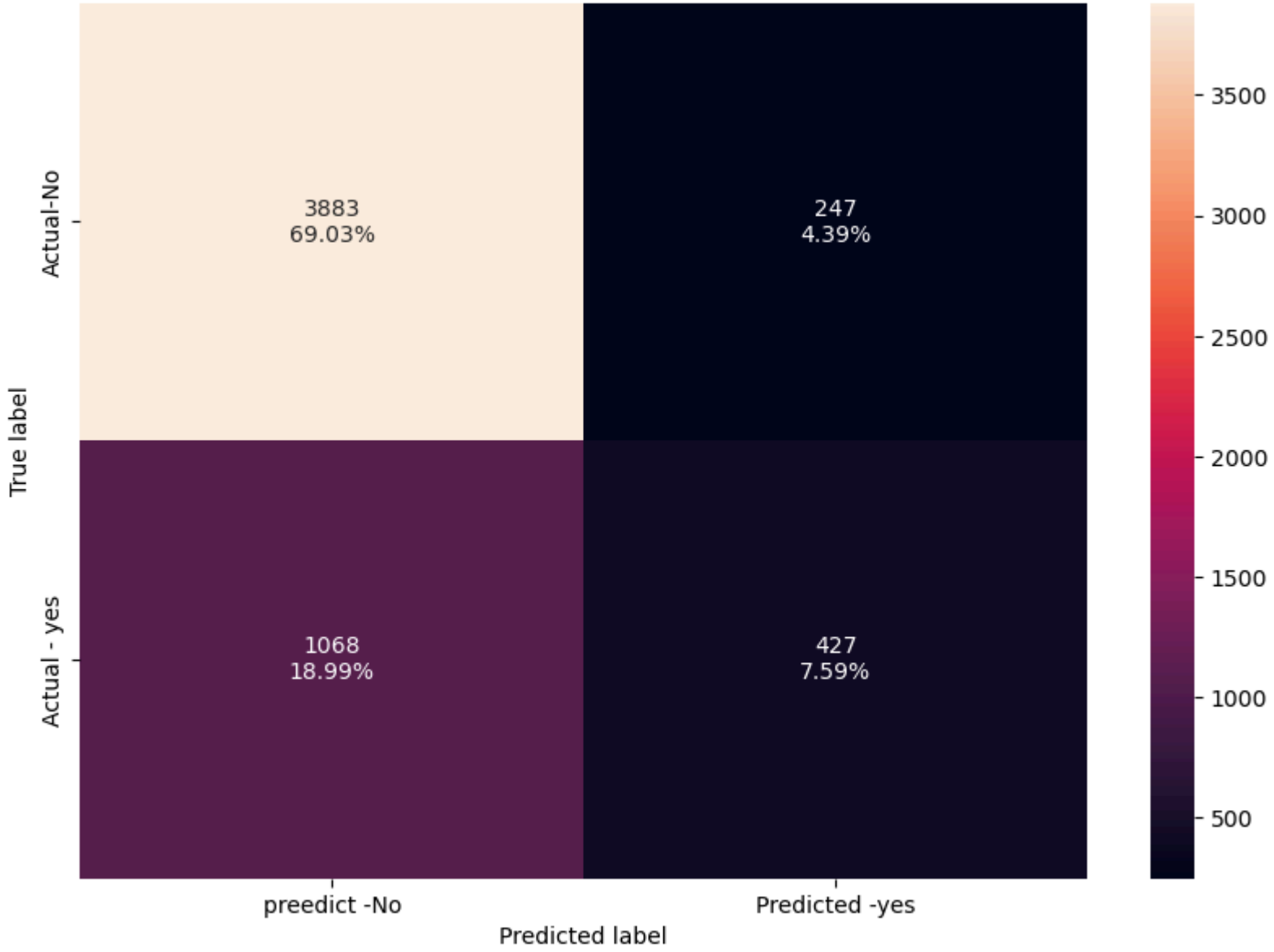
```
the matrix without setting max depth
None
```



In [494…

```
print("the matrix with depth =3 ")
make_confusion_matrix(dTree_2,y_test)
```

```
the matrix with depth =3
```



In [495…

```
print (pd.DataFrame(dTree_2.feature_importances_, columns = ["Imp"], index = X_train.columns))
```

```
                      Imp
SeniorCitizen     0.000000
tenure            0.149120
MonthlyCharges    0.289885
TotalCharges      0.020987
gender            0.000000
Partner           0.000000
Dependents        0.000000
PhoneService      0.000000
MultipleLines     0.000000
InternetService   0.000000
OnlineSecurity    0.000000
OnlineBackup      0.000000
DeviceProtection  0.000000
TechSupport       0.000000
StreamingTV       0.000000
StreamingMovies   0.000000
Contract          0.540008
PaperlessBilling  0.000000
PaymentMethod     0.000000
```

3 B. Use grid search and improve the performance of the Decision tree model , check the performance of the model on train and test data , provide the differences observed in performance in Q3.a and Q3.b (5 marks)

lets perfom Hyperparameter tuning to improve the performance

some of the important hyperparameter available for Decision tree classifier are

- criterion : {"gini", "entropy", "log_loss"}, default="gini" The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the
- splitter : {"best", "random"}, default="best" The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.
- max_depth : int, default=None The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- min_samples_split : int or float, default=2 The minimum number of samples required to split an internal node:

- min_samples_leaf : int or float, default=1 The minimum number of samples required to be at a leaf node.
- min_weight_fraction_leaf : float, default=0.0 The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.
- max_features : int, float or {"auto", "sqrt", "log2"}, default=None The number of features to consider when looking for the best split:
- random_state : int, RandomState instance or None, default=None Controls the randomness of the estimator.
- max_leaf_nodes : int, default=None Grow a tree with `max_leaf_nodes` in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.
- min_impurity_decrease : float, default=0.0 A node will be split if this split induces a decrease of the impurity greater than or equal to this value.
    - class_weight : dict, list of dict or "balanced", default=None Weights associated with classes in the form `{class_label: weight}`. If None, all classes are supposed to have weight one. For multi-output problems, a list of dicts can be provided in the same order as the columns of y.
    - ccp_alpha : non-negative float, default=0.0 Complexity parameter used for Minimal Cost-Complexity Pruning.

Decision Tree classifier grid search for parameters

In [496…

```
#choose the type of classifier
Decision_tree_tuned = DecisionTreeClassifier(random_state=1)

# Grid of parameters to choose from
## add from article

parameters={'max_features' :  [0.7,0.8,0.9,1],
            'max_depth'  :[3,7,20,30,40],
            'ccp_alpha' : [0.014,0.12,0.21,0.50],
            }
# type of scoring used to compare paramets conbinations
acc_scorer = metrics.make_scorer(metrics.recall_score)

# run th grid search
grid_obj = GridSearchCV(Decision_tree_tuned, parameters, scoring=acc_scorer,cv=5)
```

```
grid_obj = grid_obj.fit(X_train, y_train)

#set the clf to the best combination of parameters
Decision_tree_tuned = grid_obj.best_estimator_

#fit the best algorith to the data
Decision_tree_tuned.fit(X_train,y_train)
```

Out[496]:

DecisionTreeClassifier

```
DecisionTreeClassifier(ccp_alpha=0.014, max_depth=3, max_features=0.8,
                       random_state=1)
```

lets check different metrics for Decision tree classifier with best hyperparameters and build a confusion matrix

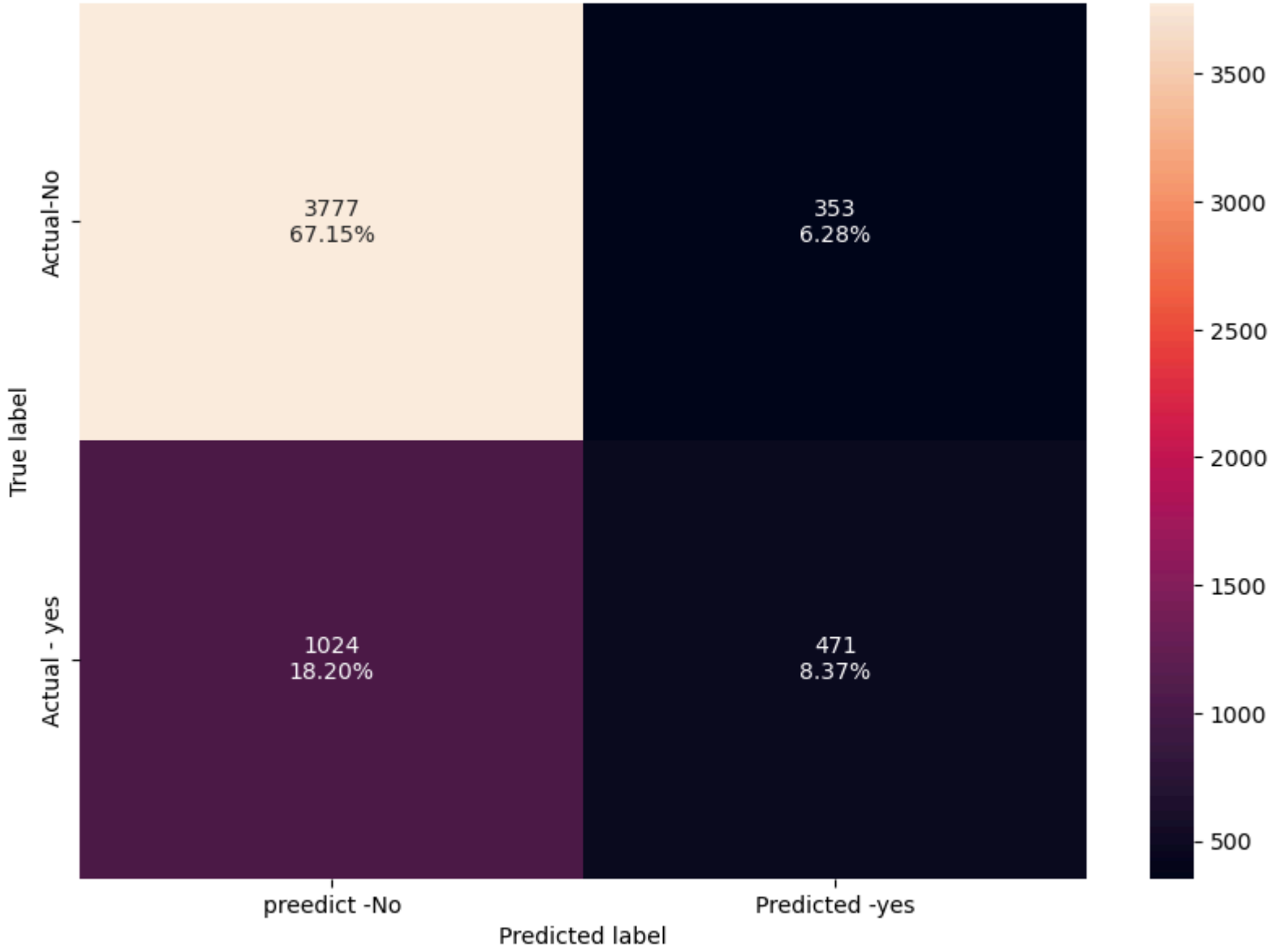
In [497…

```
Decision_tree_tuned_score=get_metrics_score(Decision_tree_tuned)
```

```
accuracy on training set :  0.767590618336887
accuracy on test set :  0.7552
Recall on training set :  0.3315508021390374
Recall on test set :  0.31505016722408025
Precision on training set :  0.6169154228855721
Precision on test set :  0.5716019417475728
```

In [498…

```
make_confusion_matrix(Decision_tree_tuned,y_test)
```



provide the differences observed in performance in Q3.a and Q3.b

** before tuning

- accuracy on training set : 1.0

- accuracy on test set : 0.7171555555555555
- Recall on training set : 1.0
- Recall on test set : 0.4916387959866221
- Precision on training set : 1.0
- Precision on test set : 0.46934865900383144

** after tuning

- accuracy on training set : 0.767590618336887
- accuracy on test set : 0.7552
- Recall on training set : 0.3315508021390374
- Recall on test set : 0.31505016722408025
- Precision on training set : 0.6169154228855721
- Precision on test set : 0.5716019417475728

Observation

- without any tuning we got accuracy for train data is so high compare to test data hence model is completely overfitted and noise is seen in model
- after tunnig we have selected parameters maximum depth,max_fatures and ccp_alpha. accuracy has balance for both tarin and test data but recall did not performed well in both means within total charn we did not identify/pridicted much

3 C. Train a model using Random forest and check the performance of the model on train and test data (4 marks)

In [499…
```
from sklearn.ensemble import RandomForestClassifier
rf_estimator=RandomForestClassifier(random_state=1)
rf_estimator.fit(X_train,y_train)
```

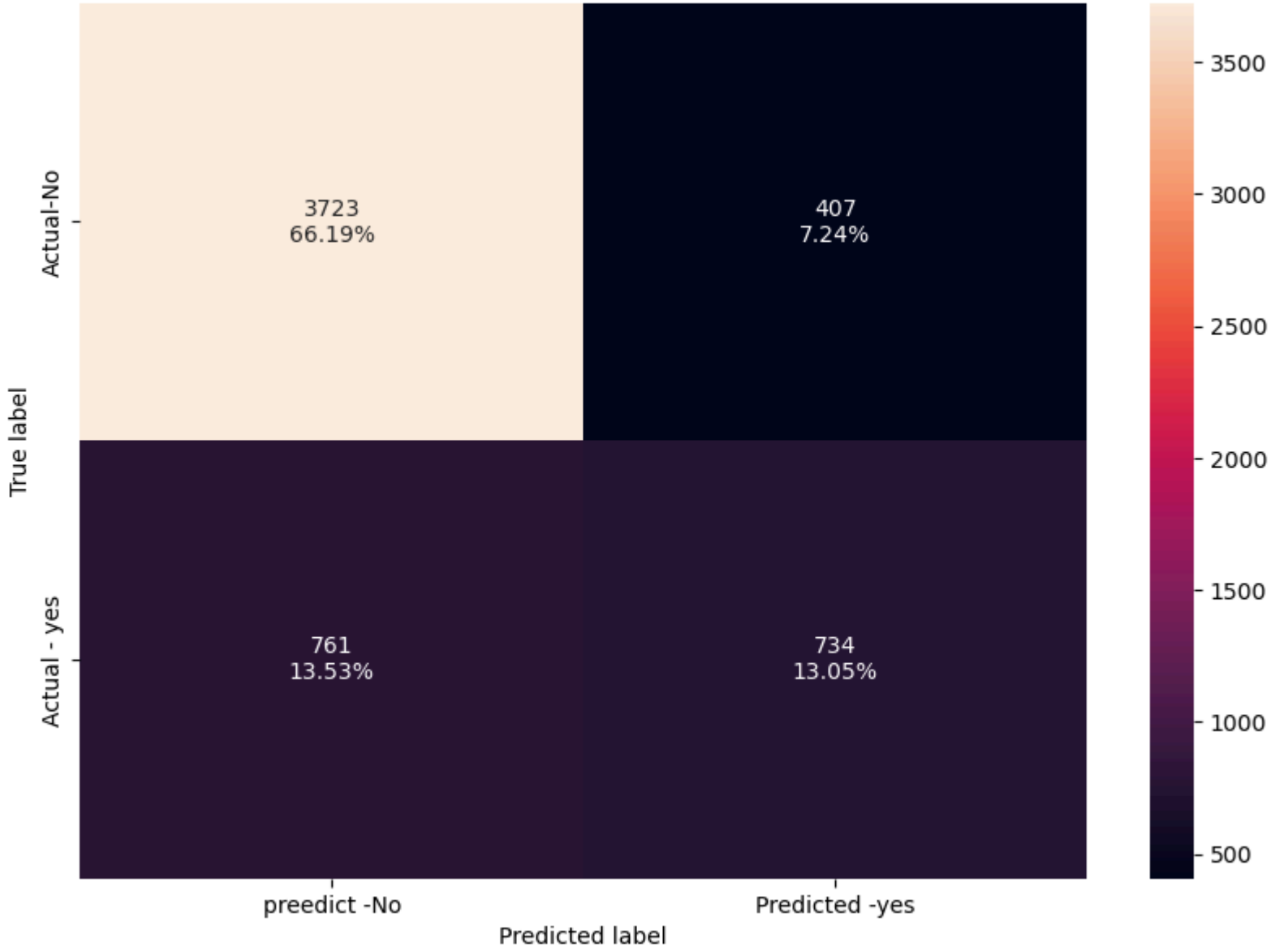
Out[499]:
```
RandomForestClassifier(random_state=1)
```

In [500…
```
rf_estimator_score=get_metrics_score(rf_estimator)
```

```
accuracy on training set :  1.0
accuracy on test set :  0.7923555555555556
Recall on training set :  1.0
Recall on test set :  0.4909698996655518
Precision on training set :  1.0
Precision on test set :  0.6432953549517967
```

In [501…

```
make_confusion_matrix(rf_estimator,y_test)
```



3D. Use grid search and improve the performance of the Random tree model , check the performance of the model on train and test data , provide the differences observed in performance in Q3.c and Q3.d (5 marks)

lets perform hyperparameter tuning performance

some of the important hyperparameter available for Decision tree classifier are

- n_estimators : int, default=100 The number of trees in the forest.
- criterion : {"gini", "entropy", "log_loss"}, default="gini" The function to measure the quality of a split.
- max_depth : int, default=None The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- min_samples_split : int or float, default=2 The minimum number of samples required to split an internal node:
- min_samples_leaf : int or float, default=1 The minimum number of samples required to be at a leaf node.
- min_weight_fraction_leaf : float, default=0.0 The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.
- max_features : {"sqrt", "log2", None}, int or float, default="sqrt" The number of features to consider when looking for the best split:
- max_leaf_nodes : int, default=None Grow trees with `max_leaf_nodes` in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.
- min_impurity_decrease : float, default=0.0 A node will be split if this split induces a decrease of the impurity greater than or equal to this value.
- bootstrap : bool, default=True Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.
- oob_score : bool or callable, default=False Whether to use out-of-bag samples to estimate the generalization score.
- n_jobs : int, default=None The number of jobs to run in parallel.
- random_state : int, RandomState instance or None, default=None Controls both the randomness of the bootstrapping of the samples used when building trees.
- verbose : int, default=0 Controls the verbosity when fitting and predicting.

- warm_start : bool, default=False When set to `True`, reuse the solution of the previous call to fit and add more estimators to the ensemble, otherwise, just fit a whole
- class_weight : {"balanced", "balanced_subsample"}, dict or list of dicts, default=None Weights associated with classes in the form
- ccp_alpha : non-negative float, default=0.0 Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than
- max_samples : int or float, default=None If bootstrap is True, the number of samples to draw from X to train each base estimator.

In [523…

```
#choose the type of classifier
Rforest_tree_tuned = RandomForestClassifier(random_state=1)

# Grid of parameters to choose from
## add from article

parameters={
            'max_features' :  [0.7,0.8,0.9,1],
            'max_depth'  :[3,7,20],
            #'ccp_alpha' : [0.014,0.12,0.21,0.50],
            'max_samples':np.arange(0.3,0.7,0.1),
            #'min_samples_leaf': np.arange(5,10),
            }
# type of scoring used to compare paramets conbinations
acc_scorer = metrics.make_scorer(metrics.recall_score)

# run th grid search
grid_obj = GridSearchCV(Rforest_tree_tuned, parameters, scoring=acc_scorer,cv=5)
grid_obj = grid_obj.fit(X_train, y_train)

#set the clf to the best combination of parameters
Rforest_tree_tuned = grid_obj.best_estimator_

#fit the beswt algorith to the data
Rforest_tree_tuned.fit(X_train,y_train)
```

Out[523]:

RandomForestClassifier

```
RandomForestClassifier(max_depth=7, max_features=0.8,
                       max_samples=0.6000000000000001, random_state=1)
```

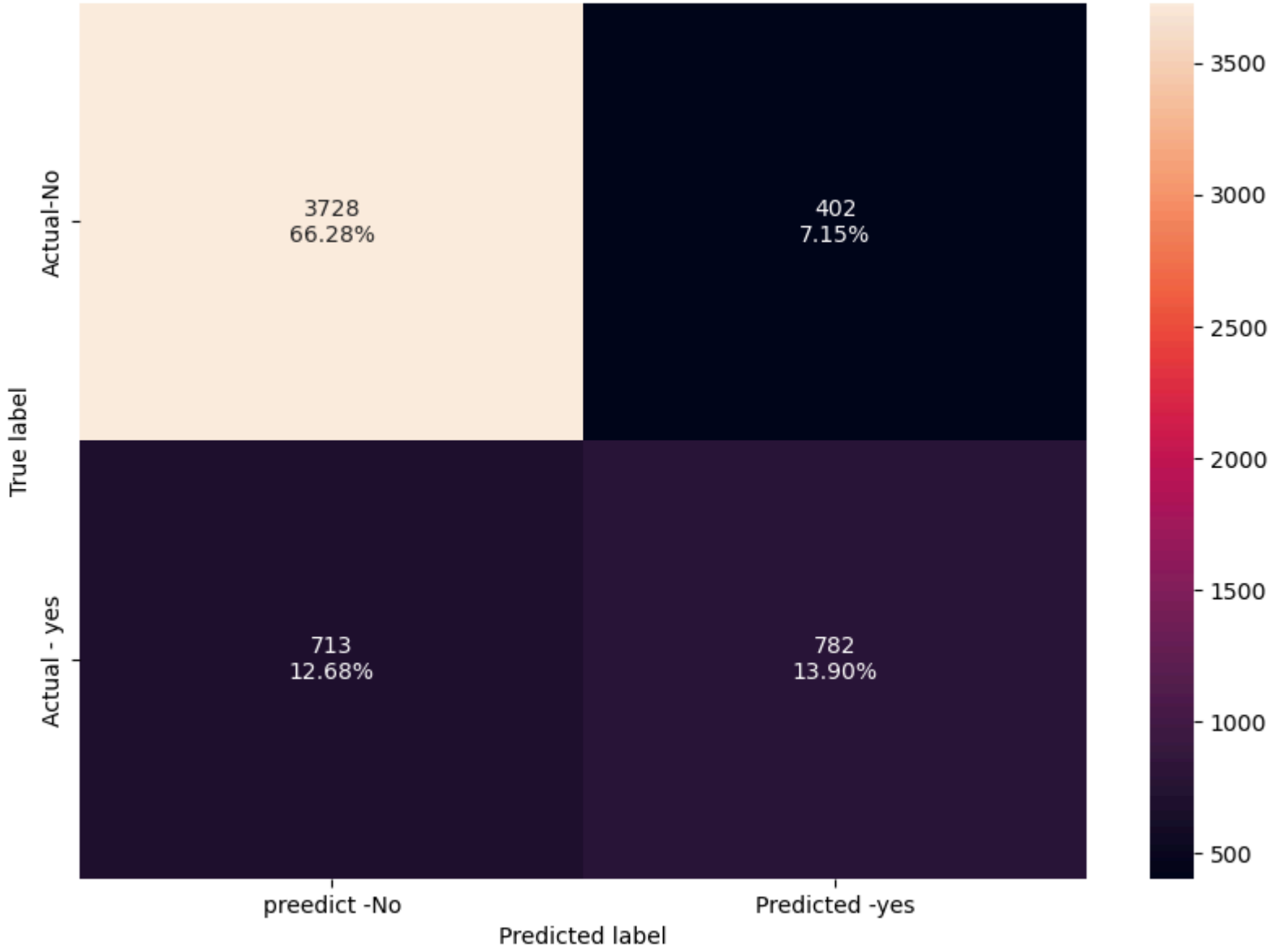
In [524…

```
#using above defined function to get accuracy, recall and precisio on train and test set
Rforest_tree_tuned_score=get_metrics_score(Rforest_tree_tuned)
```

```
accuracy on training set :  0.8734896943852167
accuracy on test set :  0.8017777777777778
Recall on training set :  0.679144385026738
Recall on test set :  0.5230769230769231
Precision on training set :  0.8141025641025641
Precision on test set :  0.660472972972973
```

In [525…

```
make_confusion_matrix(Rforest_tree_tuned,y_test)
```



provide the differences observed in performance in Q3.c and Q3.c

** before tunning

- accuracy on training set : 1.0

- accuracy on test set : 0.7923555555555556
- Recall on training set : 1.0
- Recall on test set : 0.4909698996655518
- Precision on training set : 1.0
- Precision on test set : 0.6432953549517967

** after tunning

- accuracy on training set : 0.8734896943852167
- accuracy on test set : 0.8017777777777778
- Recall on training set : 0.679144385026738
- Recall on test set : 0.5230769230769231
- Precision on training set : 0.8141025641025641
- Precision on test set : 0.660472972972973

Observation

- without any tuning we got accuracy for train data is so high compare to test data hence model is completely overfitted and noise is seen in model
- after tunnig we have selected parameters maximum depth,max_fatures and ccp_alpha. accuracy has balance for both tarin and test data but recall did not performed well in both. means within total charn we did not identify/pridicted much
- recall and precision improved than decision tree

3 E. Train a model using Adaboost and check the performance of the model on train and test data (4 marks)

In [526…
```
adaboost_estimator=AdaBoostClassifier(random_state=1)
adaboost_estimator.fit(X_train,y_train)
```

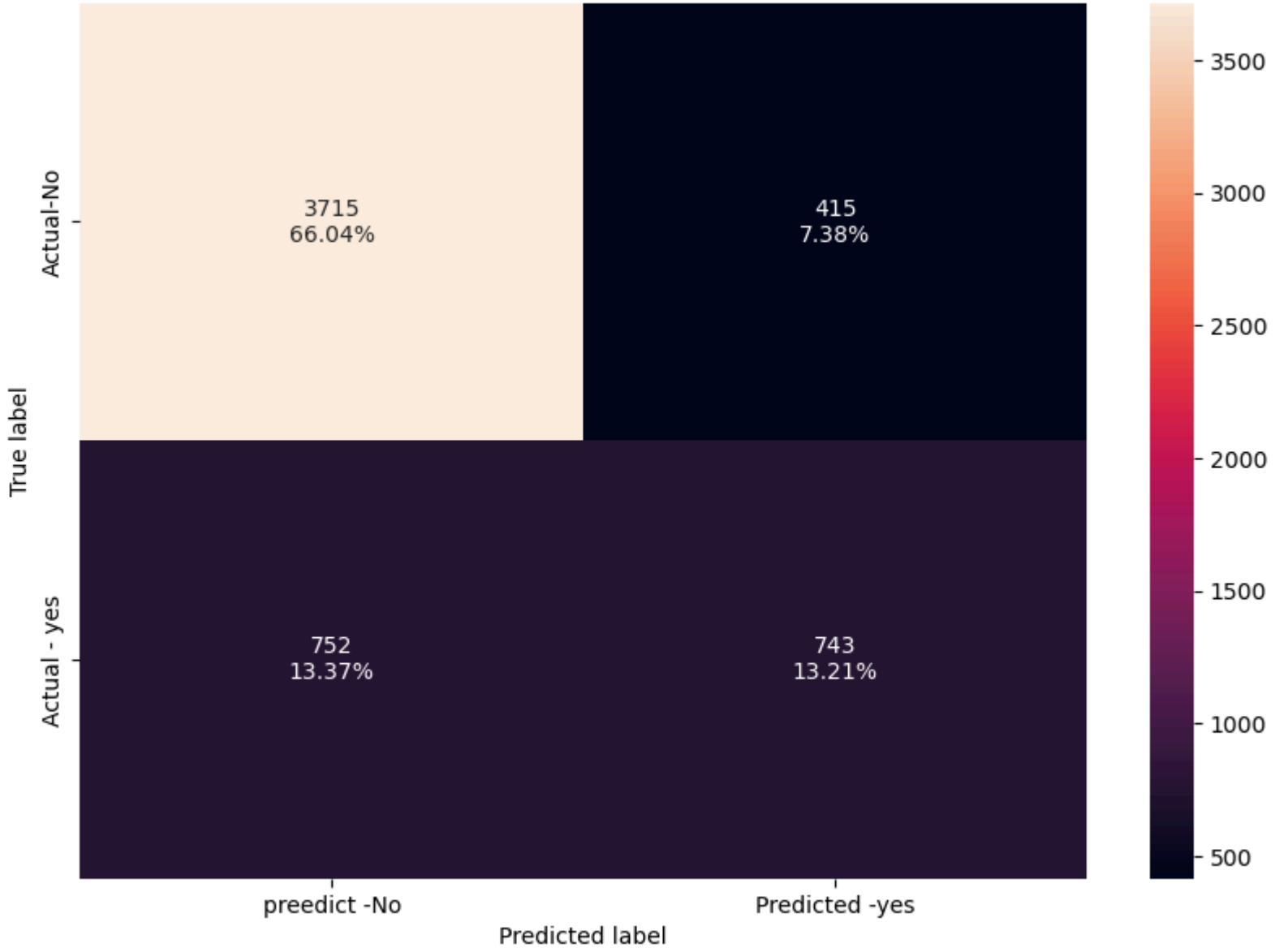
Out[526]:
```
AdaBoostClassifier
AdaBoostClassifier(random_state=1)
```

In [527…
```
adaboost_estimator_score=get_metrics_score(adaboost_estimator)
```

```
accuracy on training set :  0.8187633262260128
accuracy on test set :  0.7925333333333333
Recall on training set :  0.5454545454545454
Recall on test set :  0.49698996655518396
Precision on training set :  0.7058823529411765
Precision on test set :  0.6416234887737479
```

In [528…

```
make_confusion_matrix(adaboost_estimator,y_test)
```



3 F. Use grid search and improve the performance of the Adaboost model , check the performance of the model on train and test data , provide thedifferences observed in performance in Q3.e and Q3.f (5 marks)

let perfom hyperparameter check on ada boost model

some of the important hyperparameter available for Adaboost model are classifier are

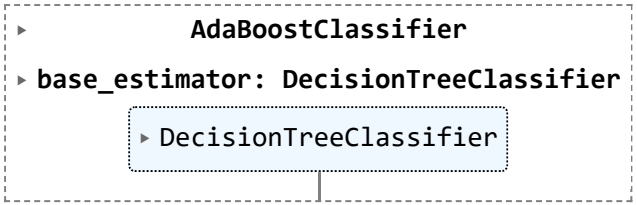
- estimator : object, default=None The base estimator from which the boosted ensemble is built. Support for sample weighting is required, as well as proper
- n_estimators : int, default=50 The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early.
- learning_rate : float, default=1.0 Weight applied to each classifier at each boosting iteration. A higher learning rate increases the contribution of each classifier.
- algorithm : {'SAMME', 'SAMME.R'}, default='SAMME.R' If 'SAMME.R' then use the SAMME.R real boosting algorithm.
- random_state : int, RandomState instance or None, default=None Controls the random seed given at each `estimator` at each boosting iteration.
- base_estimator : object, default=None The base estimator from which the boosted ensemble is built. Support for sample weighting is required, as well as proper

In [529…

```
#choose the type of classifier
Adaboost_tuned = AdaBoostClassifier(random_state=1)

# Grid of parameters to choose from
## add from article

parameters={
             'base_estimator': [dTree],
             'n_estimators': [100,150,200,250],
             'learning_rate':[1,2,3,4,5],
            }
# type of scoring used to compare paramets conbinations
acc_scorer = metrics.make_scorer(metrics.recall_score)

# run th grid search
grid_obj = GridSearchCV(Adaboost_tuned, parameters, scoring=acc_scorer,cv=5)
```

```
grid_obj = grid_obj.fit(X_train, y_train)

#set the clf to the best combination of parameters
Adaboost_tuned = grid_obj.best_estimator_

#fit the beswt algorith to the data
Adaboost_tuned.fit(X_train,y_train)
```

Out[529]:



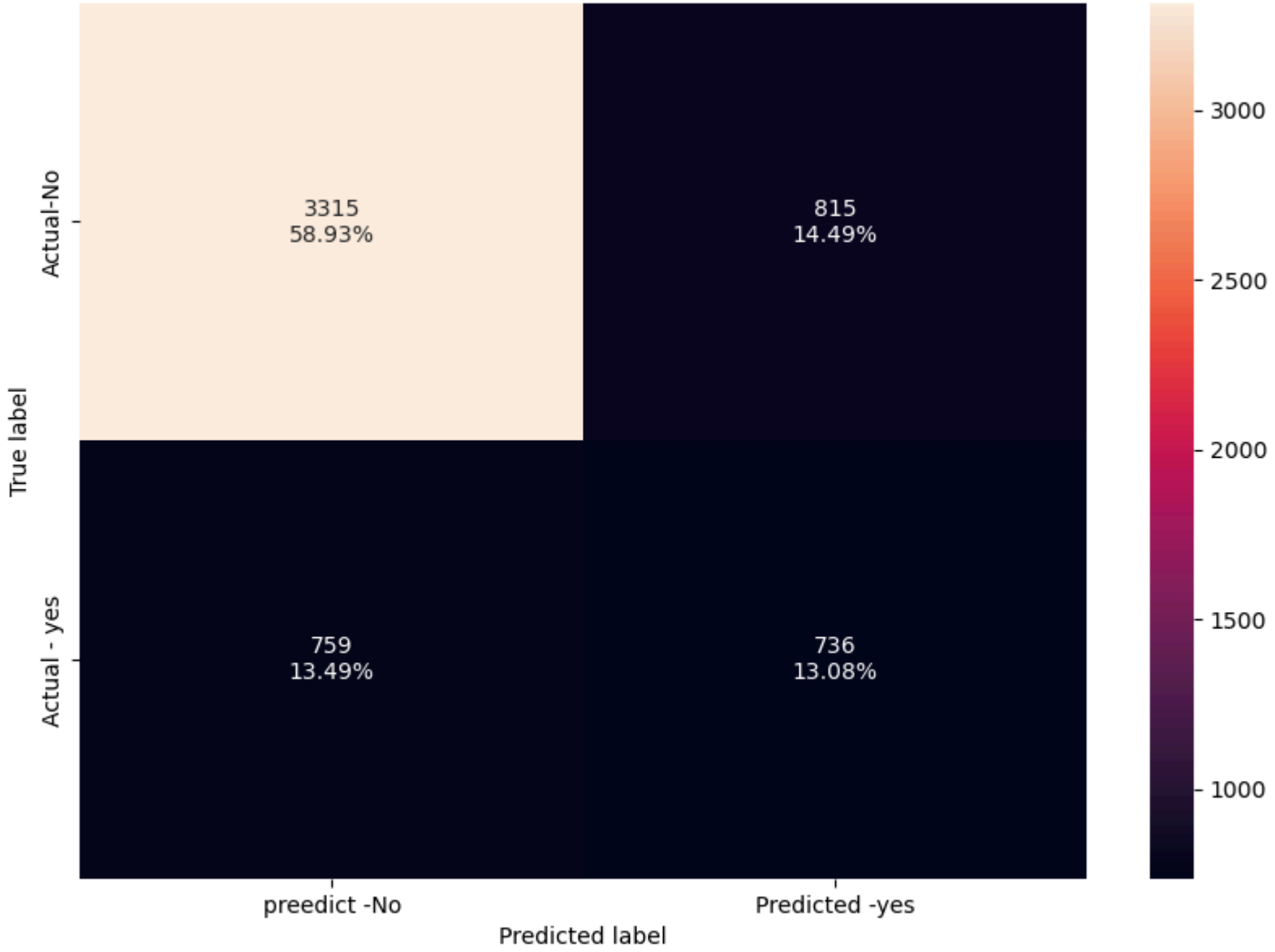
In [530…

```
Adaboost_tuned_score=get_metrics_score(Adaboost_tuned)
```

```
accuracy on training set :  1.0
accuracy on test set :  0.7201777777777778
Recall on training set :  1.0
Recall on test set :  0.49230769230769234
Precision on training set :  1.0
Precision on test set :  0.47453255963894264
```

In [531…

```
make_confusion_matrix(Adaboost_tuned,y_test)
```



provide thedifferences observed in performance in Q3.e and Q3.f

** before tunning

- accuracy on training set : 0.8187633262260128
- accuracy on test set : 0.7925333333333333
- Recall on training set : 0.5454545454545454
- Recall on test set : 0.49698996655518396
- Precision on training set : 0.7058823529411765
- Precision on test set : 0.6416234887737479

** after tunning

- accuracy on training set : 1.0
- accuracy on test set : 0.7201777777777778
- Recall on training set : 1.0
- Recall on test set : 0.49230769230769234
- Precision on training set : 1.0
- Precision on test set : 0.47453255963894264

** observation:

- this model performed well without tunning in accuracy, and recall has improve then previous two model but
- after tunning it was seen it was overfittd due to 'base_estimator'we used decision tree it self,and other parameters like'n_estimators ,'learning_rate' lead this model to overfitted

3 G. Train a model using GradientBoost and check the performance of the model on train and test data (4 marks)

In [532…
```
Gd_boost_estimator=GradientBoostingClassifier(n_estimators=50,random_state=1)
Gd_boost_estimator.fit(X_train,y_train)
```

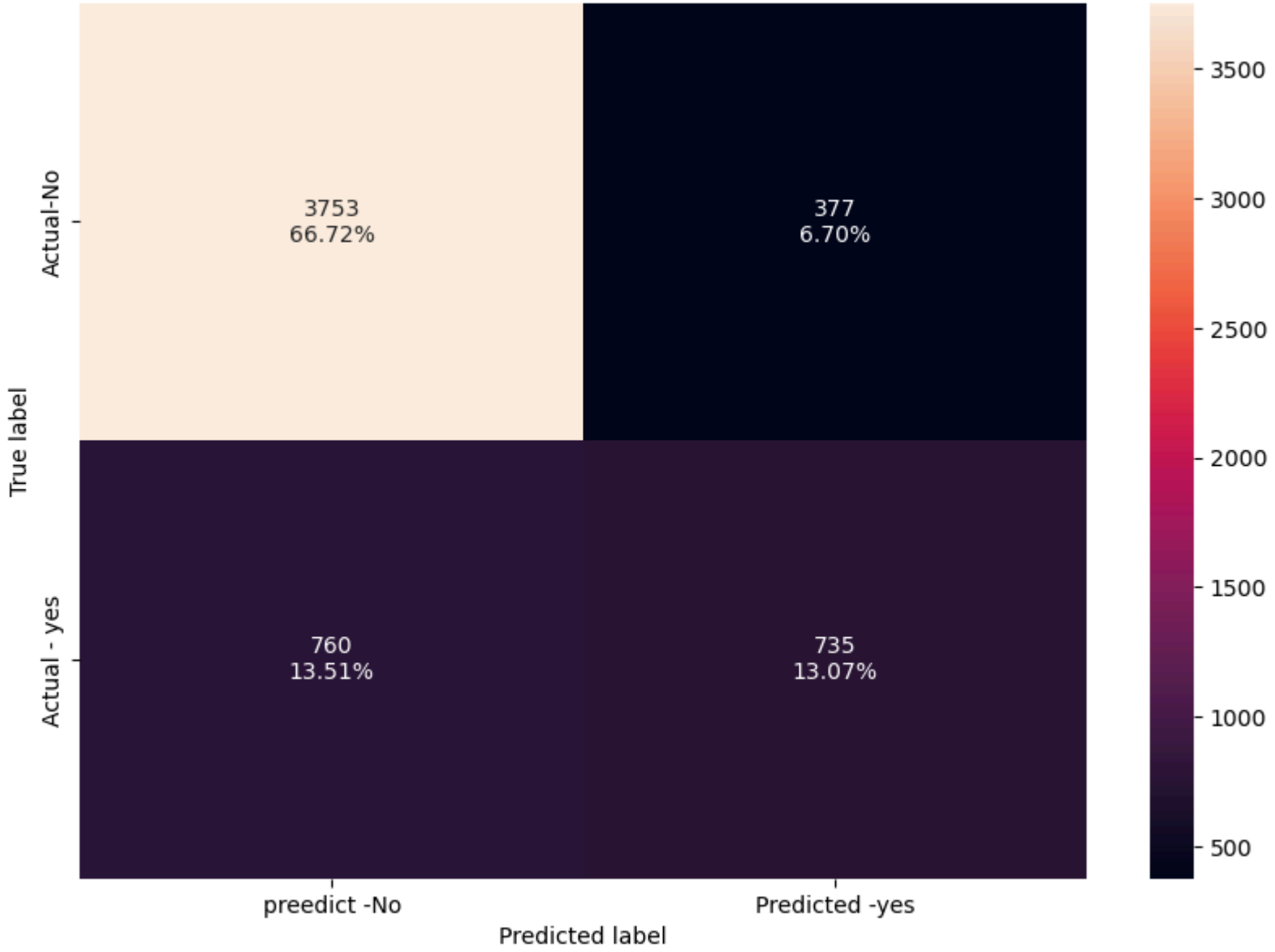
Out[532]:
```
GradientBoostingClassifier(n_estimators=50, random_state=1)
```

In [533…
```
Gd_boost_estimator_score=get_metrics_score(Gd_boost_estimator)
```

```
accuracy on training set :  0.835820895522388
accuracy on test set :  0.7978666666666666
Recall on training set :  0.5721925133689839
Recall on test set :  0.4916387959866221
Precision on training set :  0.7508771929824561
Precision on test set :  0.6609712230215827
```

In [534…

```
make_confusion_matrix(Gd_boost_estimator,y_test)
```



3 H. Use grid search and improve the performance of the GradientBoost model , check the performance of the model on train and test data , provide the differences observed in performance in Q3.g and Q3.h (5 marks)

let perfom hyperparameter check on ada boost model

some of the important hyperparameter available for Gradientboost model are classifier are

- loss : {'log_loss', 'exponential'}, default='log_loss' The loss function to be optimized.
- learning_rate : float, default=0.1 Learning rate shrinks the contribution of each tree by
- n_estimators : int, default=100 The number of boosting stages to perform.
- subsample : float, default=1.0 The fraction of samples to be used for fitting the individual base learners.
- criterion : {'friedman_mse', 'squared_error'}, default='friedman_mse' The function to measure the quality of a split.
- min_samples_split : int or float, default=2 The minimum number of samples required to split an internal node:
- min_samples_leaf : int or float, default=1 The minimum number of samples required to be at a leaf node.
- min_weight_fraction_leaf : float, default=0.0 The minimum weighted fraction of the sum total of weights
- max_depth : int or None, default=3 Maximum depth of the individual regression estimators. The maximum depth limits the number of nodes in the tree.
- min_impurity_decrease : float, default=0.0 A node will be split if this split induces a decrease of the impurity greater than or equal to this value.
- init : estimator or 'zero', default=None An estimator object that is used to compute the initial predictions.
- random_state : int, RandomState instance or None, default=None Controls the random seed given to each Tree estimator at each boosting iteration.
- max_features : {'sqrt', 'log2'}, int or float, default=None The number of features to consider when looking for the best split:
- verbose : int, default=0 Enable verbose output. If 1 then it prints progress and performance once in a while (the more trees the lower the frequency).

- max_leaf_nodes : int, default=None
- warm_start : bool, default=False
- validation_fraction : float, default=0.1 The proportion of training data to set aside as validation set for early stopping.
- n_iter_no_change : int, default=None `n_iter_no_change` is used to decide if early stopping will be used to terminate training when validation score is not improving.
- tol : float, default=1e-4 Tolerance for the early stopping.
- ccp_alpha : non-negative float, default=0.0 Complexity parameter used for Minimal Cost-Complexity Pruning.

In [535…

```
#choose the type of classifier
Gradeboost_tuned = GradientBoostingClassifier(random_state=1)

# Grid of parameters to choose from
## add from article

parameters={
             'n_estimators': [100,150,200,250],
             'learning_rate':[1,2,3,4,5],
             #'max_depth':[2,3,4,5,7]
            }
# type of scoring used to compare paramets conbinations
acc_scorer = metrics.make_scorer(metrics.recall_score)

# run th grid search
grid_obj = GridSearchCV(Gradeboost_tuned, parameters, scoring=acc_scorer,cv=5)
grid_obj = grid_obj.fit(X_train, y_train)

#set the clf to the best combination of parameters
Gradeboost_tuned = grid_obj.best_estimator_

#fit the beswt algorith to the data
Gradeboost_tuned.fit(X_train,y_train)
```

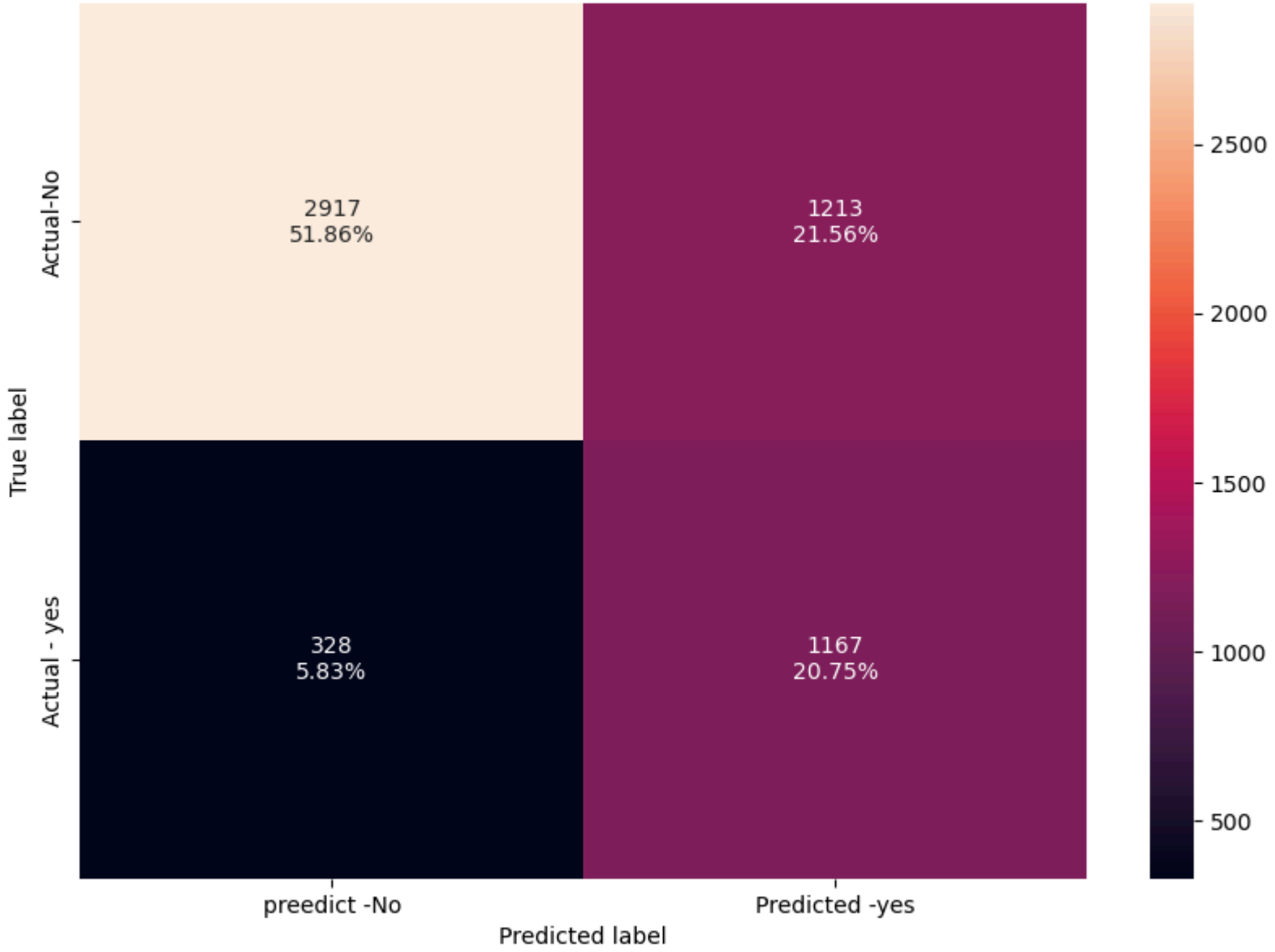
Out[535]:

```
GradientBoostingClassifier(learning_rate=4, random_state=1)
```

In [538…
```
Gradeboost_tuned_score=get_metrics_score(Gradeboost_tuned)
```

```
accuracy on training set :  0.7313432835820896
accuracy on test set :  0.7260444444444445
Recall on training set :  0.7941176470588235
Recall on test set :  0.7806020066889632
Precision on training set :  0.49665551839464883
Precision on test set :  0.4903361344537815
```

In [539…
```
make_confusion_matrix(Gradeboost_tuned,y_test)
```



provide the differences observed in performance in Q3.g and Q3.h

** before tuning

- accuracy on training set : 0.835820895522388

- accuracy on test set : 0.7978666666666666
- Recall on training set : 0.5721925133689839
- Recall on test set : 0.4916387959866221
- Precision on training set : 0.7508771929824561
- Precision on test set : 0.6609712230215827

** after tuning

- accuracy on training set : 0.7313432835820896
- accuracy on test set : 0.7260444444444445
- Recall on training set : 0.7941176470588235
- Recall on test set : 0.7806020066889632
- Precision on training set : 0.49665551839464883
- Precision on test set : 0.4903361344537815

** observation

- before tunning recall was not as per required we did not pridicted well for customer going to chrn out of total actual has churned
- durung tuning we used learing rate and n_etimator and at learning rate 4 it performed well on recall, but accuracy was low after grid search and parameter tuning

3 I. Provide detailed analysis of the below steps (4 marks) :

- (1) Compare the performance of each model in train stage and test stage
- (2) Provide your observation on which model performed the best
- (3) Provide your reasoning on why the model performed best
- (4) Provide your final conclusion on your observation

- **(1) Comparing the performance of each model in train stage and test stage**

In [540…

```
# defining list of models
models = [dTree,Decision_tree_tuned,rf_estimator,Rforest_tree_tuned,adaboost_estimator,Adaboost_tuned,Gd_boost_estimator,Gradeboo

# defining empty list to add train and test results
```

```
acc_train=[]
acc_test=[]
recall_train=[]
recall_test = []
precision_train = []
precision_test = []

#Loop throught all  the models to get the accuracy, precall and precision scores

for model in models:
    j=get_metrics_score(model,False)
    acc_train.append(np.round(j[0],2))
    acc_test.append(np.round(j[1],2))
    recall_train.append(np.round(j[2],2))
    recall_test.append(np.round(j[3],2))
    precision_train.append(np.round(j[4],2))
    precision_test.append(np.round(j[5],2))
```

In [541…

```
comparison_frame = pd.DataFrame({'Model':['Decision Tree with default parameter',
                                          'Decision tree with grid(tuned parameter)search',
                                          'Random forest with default parameters',
                                          'Random forest with grid(tuned parameter)search',
                                          'Ada boosting technique with deafault parameter',
                                          'Ada boosting with grid (tuned parameter)search',
                                          'Gradient boosting technique with deafault parameter',
                                          'Gradient boosting with grid (tuned parameter)search'],

                                 'Train_Accuracy':acc_train,
                                 'Test_Accuracy':acc_test,
                                 'Train_Recall':recall_train,
                                 'Test_Recall':recall_test,
                                 'Train_Precision':precision_train,
                                 'Test_Precision':precision_test})


comparison_frame
```

Out[541]:

	Model	Train_Accuracy	Test_Accuracy	Train_Recall	Test_Recall	Train_Precision	Test_Precision
0	Decision Tree with default parameter	1.00	0.72	1.00	0.49	1.00	0.47
1	Decision tree with grid(tuned parameter)search	0.77	0.76	0.33	0.32	0.62	0.57
2	Random forest with default parameters	1.00	0.79	1.00	0.49	1.00	0.64
3	Random forest with grid(tuned parameter)search	0.87	0.80	0.68	0.52	0.81	0.66
4	Ada boosting technique with deafault parameter	0.82	0.79	0.55	0.50	0.71	0.64
5	Ada boosting with grid (tuned parameter)search	1.00	0.72	1.00	0.49	1.00	0.47
6	Gradient boosting technique with deafault para...	0.84	0.80	0.57	0.49	0.75	0.66
7	Gradient boosting with grid (tuned parameter)s...	0.73	0.73	0.79	0.78	0.50	0.49

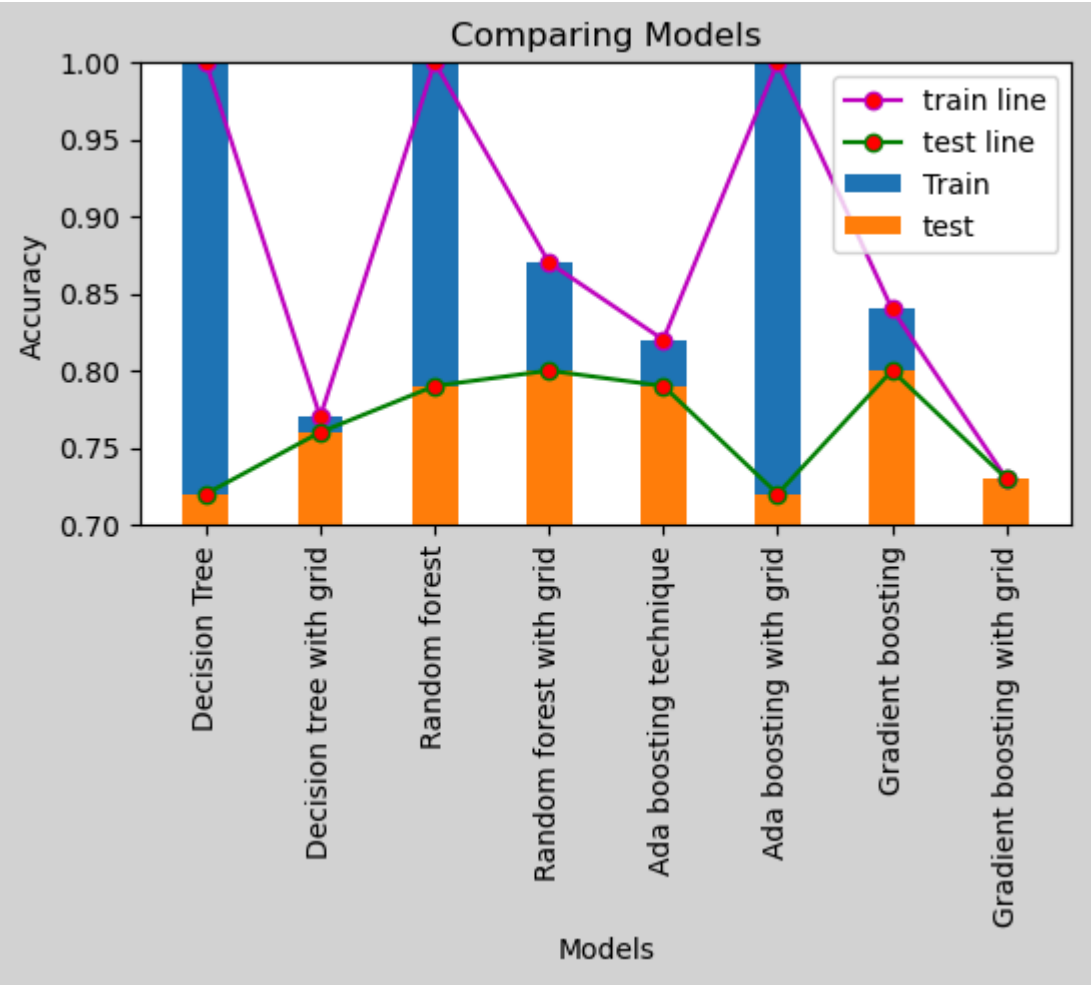
- **(2) Provide your observation on which model performed the best**

In [542...

```
fig , ax = plt.subplots(figsize=(6,3),facecolor="lightgray")
x = np.array(['Decision Tree','Decision tree with grid','Random forest','Random forest with grid','Ada boosting technique','Ada b
ax.plot(x,comparison_frame["Train_Accuracy"],marker="o",mfc="red",color="m",label="train line")
ax.bar(x,comparison_frame["Train_Accuracy"],label="Train",width=0.4)
ax.plot(x,comparison_frame["Test_Accuracy"],marker="o",mfc="red",color="green",label="test line")
ax.bar(x,comparison_frame["Test_Accuracy"],label="test",width=0.4)
ax.set(xticks=x,ylim=[0.7,1],xlabel="Models",ylabel="Accuracy",
      title="Comparing Models")
plt.xticks(rotation=90)
#ax.set_xticklabels(index,rotation=90)
ax.legend()
```

Out[542]:

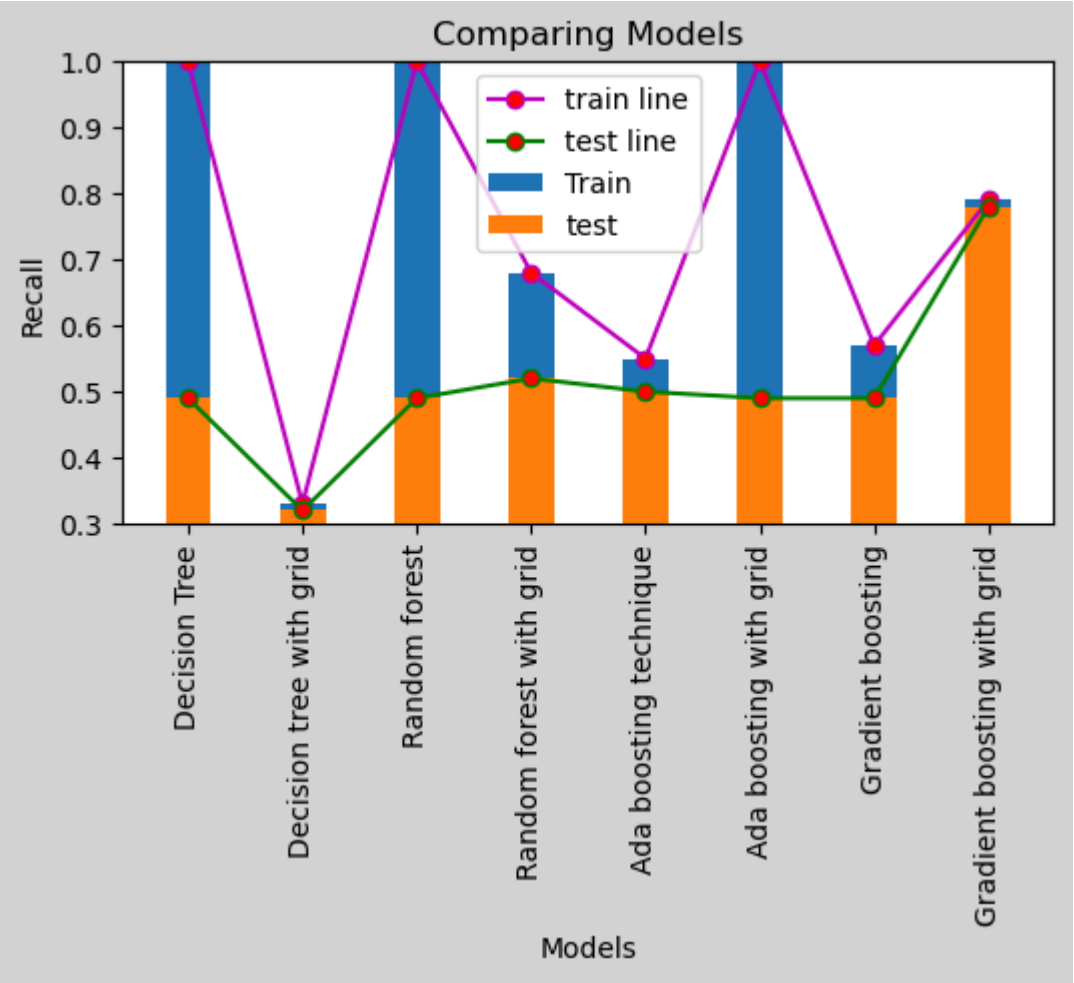
```
<matplotlib.legend.Legend at 0x2db61ee9b10>
```



In [543…

```
fig , ax = plt.subplots(figsize=(6,3),facecolor="lightgray")
x = np.array(['Decision Tree','Decision tree with grid','Random forest','Random forest with grid','Ada boosting technique','Ada b
ax.plot(x,comparison_frame["Train_Recall"],marker="o",mfc="red",color="m",label="train line")
ax.bar(x,comparison_frame["Train_Recall"],label="Train",width=0.4)
ax.plot(x,comparison_frame["Test_Recall"],marker="o",mfc="red",color="green",label="test line")
ax.bar(x,comparison_frame["Test_Recall"],label="test",width=0.4)
ax.set(xticks=x,ylim=[0.3,1],xlabel="Models",ylabel="Recall",
      title="Comparing Models")
plt.xticks(rotation=90)
#ax.set_xticklabels(index,rotation=90)
ax.legend()
```

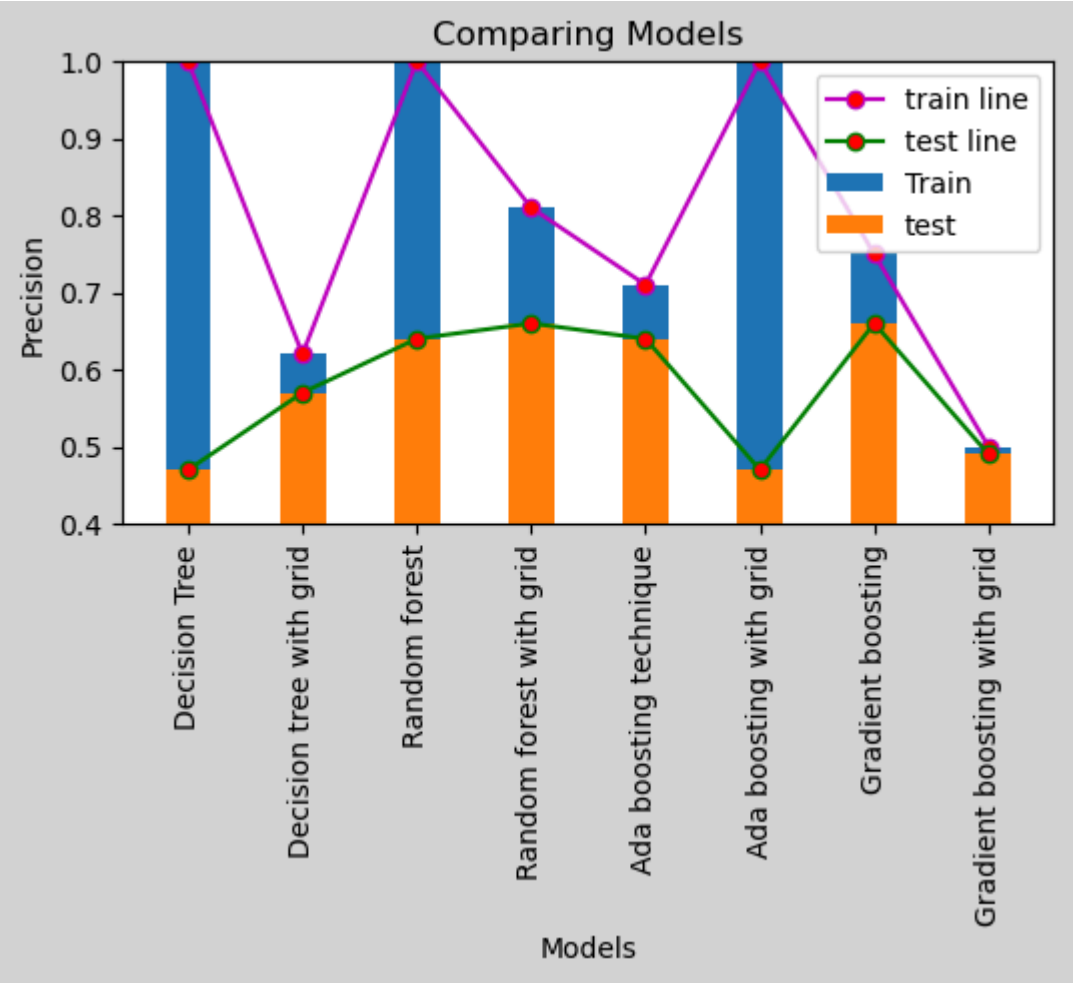
Out[543]: <matplotlib.legend.Legend at 0x2db5ff5dd90>



In [544…

```
fig , ax = plt.subplots(figsize=(6,3),facecolor="lightgray")
x = np.array(['Decision Tree','Decision tree with grid','Random forest','Random forest with grid','Ada boosting technique','Ada b
ax.plot(x,comparison_frame["Train_Precision"],marker="o",mfc="red",color="m",label="train line")
ax.bar(x,comparison_frame["Train_Precision"],label="Train",width=0.4)
ax.plot(x,comparison_frame["Test_Precision"],marker="o",mfc="red",color="green",label="test line")
ax.bar(x,comparison_frame["Test_Precision"],label="test",width=0.4)
ax.set(xticks=x,ylim=[0.4,1],xlabel="Models",ylabel="Precision",
      title="Comparing Models")
plt.xticks(rotation=90)
#ax.set_xticklabels(index,rotation=90)
ax.legend()
```

Out[544]: <matplotlib.legend.Legend at 0x2db601dafd0>



from obsevation Gradient boosting technique perform well on accuracy or in recall perforamnce parameter

(3) Provide your reasoning on why the model performed best

- the best model performed was gradient boosting because depth of tree is less hence in less complex data or tree it performs well.
- acuracy was quite good and recall was improved compare to other model

- Hyperparameter tunning is tricky and exhaustive in the sense that there is no direct way to claculate how a change in the hyperparameter value will reduce the loss of your model until you try those hyperparameers
- the fial result depend on parameter used/checked using GridsearchCV.
- there may be yet better parameter which may result in a better accuracy and recall
- if we dont define max_depth then random forest works extremly well and accuracy anf recall both gets improve.

(4) Provide your final conclusion on your observation

- Hyperparameter tunning is tricky and exhaustive in the sense that there is no direct way to claculate how a change in the hyperparameter value will reduce the loss of your model until you try those hyperparameers
- the fial result depend on parameter used/checked using GridsearchCV.
- there may be yet better parameter which may result in a better accuracy and recall
- if we dont define max_depth then random forest works extremly well and accuracy anf recall both gets improve.
- random solves the problem of overfitting as output is based on majority voting or averaging. It performs well even if the data contains null/missing values. Each decision tree created is independent of the other; thus, it shows the property of parallelization.
- here we finalise gradient boosting technique it accuracy was as per requirement and with parameter tuning we indentifed probabilty of customer to opt for service and actully have opt is also more.
- This algorithm not only supports numerical datasets but is also efficient for categorical data handling.
- Gradient boosting algorithm in machine learning is a resilient method that checks over-fitting training datasets quite easily.
- Gradient boosting models can be computationally expensive and can take a longer time to train the complete model on CPUs.

==========END OF PROJECT==========

In [ ]: