# Telecommunication

November 22, 2024

```
[ ]: import pandas as pd                             #### To read the data
     import numpy as np                              ### To Perform some Numerical␣
      ↪Calculations
     import matplotlib.pyplot as plt                 ### To Perform some␣
      ↪visualizations for Understanding
     import seaborn as sns                           ### To Perform some␣
      ↪visualizations for calaculations
     import scipy.stats as stats                     ### To check the calculations
     import warnings
     warnings.filterwarnings('ignore')               ### To remove Warnings
```

```
[ ]: df=pd.read_csv("churn.csv")                       #### Reading the Data
```

```
[ ]: df
```

```
[ ]:       Unnamed: 0 state      area.code  account.length voice.plan  \
     0              1    KS  area_code_415             128        yes
     1              2    OH  area_code_415             107        yes
     2              3    NJ  area_code_415             137         no
     3              4    OH  area_code_408              84         no
     4              5    OK  area_code_415              75         no
     …            …   …             …          …         …
     4995        4996    HI  area_code_408              50        yes
     4996        4997    WV  area_code_415             152         no
     4997        4998    DC  area_code_415              61         no
     4998        4999    DC  area_code_510             109         no
     4999        5000    VT  area_code_415              86        yes

           voice.messages intl.plan  intl.mins  intl.calls  intl.charge  …  \
     0                 25        no       10.0           3         2.70  …
     1                 26        no       13.7           3         3.70  …
     2                  0        no       12.2           5         3.29  …
     3                  0       yes        6.6           7         1.78  …
     4                  0       yes       10.1           3         2.73  …
     …              …       …        …          …          …   …
     4995              40        no        9.9           5         2.67  …
     4996               0        no       14.7           2         3.97  …
```

```
4997               0        no      13.6           4         3.67  …
4998               0        no       8.5           6         2.30  …
4999              34        no       9.3          16         2.51  …

      day.calls  day.charge eve.mins eve.calls  eve.charge  night.mins  \
0           110       45.07    197.4        99       16.78       244.7
1           123       27.47    195.5       103       16.62       254.4
2           114       41.38    121.2       110       10.30       162.6
3            71        50.9     61.9        88        5.26       196.9
4           113       28.34    148.3       122       12.61       186.9
…         …         …      …       …         …         …
4995        127       40.07      223       126       18.96       297.5
4996         90       31.31    256.8        73       21.83       213.6
4997         89        23.9    172.8       128       14.69       212.4
4998         67        32.1    171.7        92       14.59       224.4
4999        102          22    267.1       104       22.70       154.8

      night.calls  night.charge  customer.calls  churn
0              91         11.01               1     no
1             103         11.45               1     no
2             104          7.32               0     no
3              89          8.86               2     no
4             121          8.41               3     no
…           …           …             …    …
4995          116         13.39               2     no
4996          113          9.61               3    yes
4997           97          9.56               1     no
4998           89         10.10               0     no
4999          100          6.97               0     no

[5000 rows x 21 columns]
```

### 0.0.1 Exploratory Data Analysis

[ ]:
```
## Changing the data type column
df['day.charge']=df['day.charge'].astype('float')
df['eve.mins']=df['eve.mins'].astype('float')
```

[ ]:
```
### Dividing the target variable and features
features=df.drop(columns=['churn','Unnamed: 0','state'])
target=df['churn']
```

[ ]:
```
features
```

[ ]:
```
          area.code  account.length voice.plan  voice.messages intl.plan  \
0     area_code_415             128        yes              25        no
1     area_code_415             107        yes              26        no
```

```
2     area_code_415             137         no               0        no
3     area_code_408              84         no               0       yes
4     area_code_415              75         no               0       yes
…             …             …        …             …       …
4995  area_code_408              50        yes              40        no
4996  area_code_415             152         no               0        no
4997  area_code_415              61         no               0        no
4998  area_code_510             109         no               0        no
4999  area_code_415              86        yes              34        no

      intl.mins  intl.calls  intl.charge  day.mins  day.calls  day.charge  \
0          10.0           3         2.70     265.1        110       45.07
1          13.7           3         3.70     161.6        123       27.47
2          12.2           5         3.29     243.4        114       41.38
3           6.6           7         1.78     299.4         71       50.90
4          10.1           3         2.73     166.7        113       28.34
…         …         …          …       …        …         …
4995        9.9           5         2.67     235.7        127       40.07
4996       14.7           2         3.97     184.2         90       31.31
4997       13.6           4         3.67     140.6         89       23.90
4998        8.5           6         2.30     188.8         67       32.10
4999        9.3          16         2.51     129.4        102       22.00

      eve.mins  eve.calls  eve.charge  night.mins  night.calls  night.charge  \
0        197.4         99       16.78       244.7           91         11.01
1        195.5        103       16.62       254.4          103         11.45
2        121.2        110       10.30       162.6          104          7.32
3         61.9         88        5.26       196.9           89          8.86
4        148.3        122       12.61       186.9          121          8.41
…        …        …         …         …          …           …
4995     223.0        126       18.96       297.5          116         13.39
4996     256.8         73       21.83       213.6          113          9.61
4997     172.8        128       14.69       212.4           97          9.56
4998     171.7         92       14.59       224.4           89         10.10
4999     267.1        104       22.70       154.8          100          6.97

      customer.calls
0                  1
1                  1
2                  0
3                  2
4                  3
…              …
4995               2
4996               3
4997               1
4998               0
```

```
4999               0

[5000 rows x 18 columns]
```

[ ]:

```
features.info()                          #### Showing the Information about␣
 ↪each column
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   area.code        5000 non-null   object
 1   account.length   5000 non-null   int64
 2   voice.plan       5000 non-null   object
 3   voice.messages   5000 non-null   int64
 4   intl.plan        5000 non-null   object
 5   intl.mins        5000 non-null   float64
 6   intl.calls       5000 non-null   int64
 7   intl.charge      5000 non-null   float64
 8   day.mins         5000 non-null   float64
 9   day.calls        5000 non-null   int64
 10  day.charge       4993 non-null   float64
 11  eve.mins         4976 non-null   float64
 12  eve.calls        5000 non-null   int64
 13  eve.charge       5000 non-null   float64
 14  night.mins       5000 non-null   float64
 15  night.calls      5000 non-null   int64
 16  night.charge     5000 non-null   float64
 17  customer.calls   5000 non-null   int64
dtypes: float64(8), int64(7), object(3)
memory usage: 703.2+ KB
```

[ ]:

```
features.describe()       #### Showing the statistical measures for each␣
 ↪numerical columns
```

[ ]:

```
       account.length  voice.messages    intl.mins   intl.calls  intl.charge  \
count      5000.00000     5000.000000  5000.000000  5000.000000  5000.000000
mean        100.25860        7.755200    10.261780     4.435200     2.771196
std          39.69456       13.546393     2.761396     2.456788     0.745514
min           1.00000        0.000000     0.000000     0.000000     0.000000
25%          73.00000        0.000000     8.500000     3.000000     2.300000
50%         100.00000        0.000000    10.300000     4.000000     2.780000
75%         127.00000       17.000000    12.000000     6.000000     3.240000
max         243.00000       52.000000    20.000000    20.000000     5.400000

          day.mins    day.calls   day.charge     eve.mins    eve.calls  \
count  5000.000000  5000.000000  4993.000000  4976.000000  5000.000000
```

```
mean     180.288900   100.029400    30.653501   200.580326   100.191000
std       53.894699    19.831197     9.166356    50.554637    19.826496
min        0.000000     0.000000     0.000000     0.000000     0.000000
25%      143.700000    87.000000    24.430000   166.275000    87.000000
50%      180.100000   100.000000    30.620000   201.000000   100.000000
75%      216.200000   113.000000    36.750000   234.100000   114.000000
max      351.500000   165.000000    59.760000   363.700000   170.000000

         eve.charge   night.mins  night.calls  night.charge  customer.calls
count   5000.000000  5000.000000  5000.000000   5000.000000     5000.000000
mean      17.054322   200.391620    99.919200      9.017732        1.570400
std        4.296843    50.527789    19.958686      2.273763        1.306363
min        0.000000     0.000000     0.000000      0.000000        0.000000
25%       14.140000   166.900000    87.000000      7.510000        1.000000
50%       17.090000   200.400000   100.000000      9.020000        1.000000
75%       19.900000   234.700000   113.000000     10.560000        2.000000
max       30.910000   395.000000   175.000000     17.770000        9.000000
```

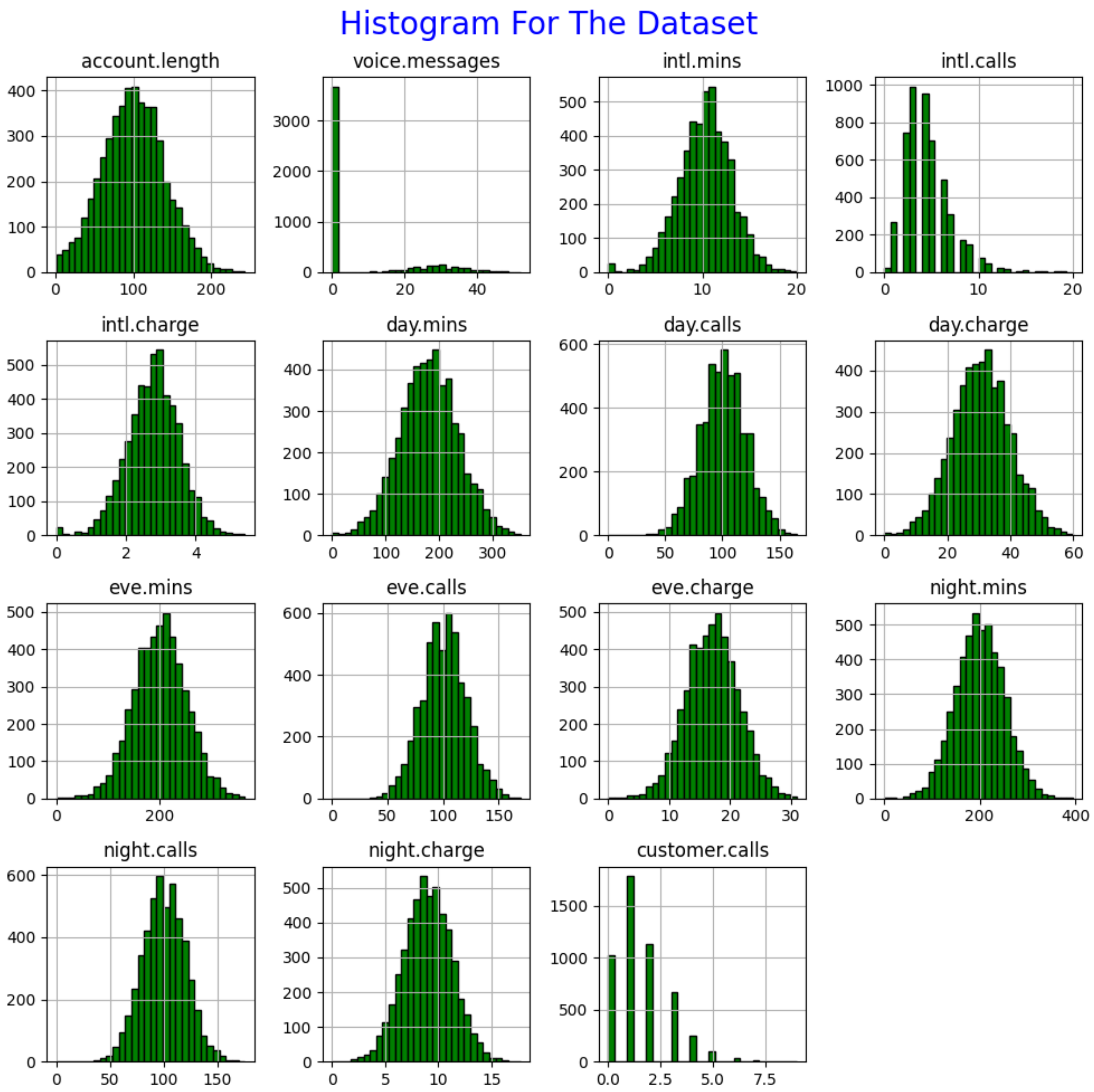
[ ]:

```

```

### 0.0.2 Visualizations For understanding the data

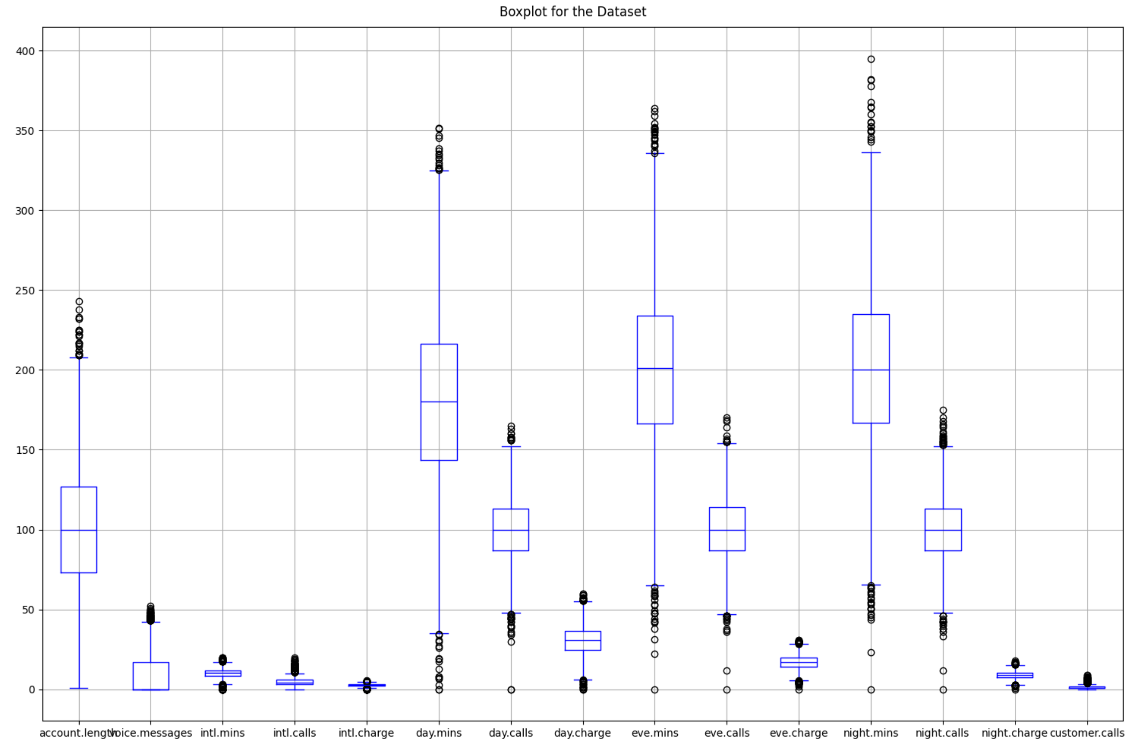
[ ]:

```python
### Histogram for the Dataset
### Only for Numerical Columns from the datset
features.hist(bins=30,figsize=(10,10),color='green',edgecolor='black') ### Showing the Histogram
plt.suptitle('Histogram For The Dataset',fontsize=20,color='blue') ### Showing the Title for the graph
plt.tight_layout() ### Removes unnecessary congestion
plt.show()
```



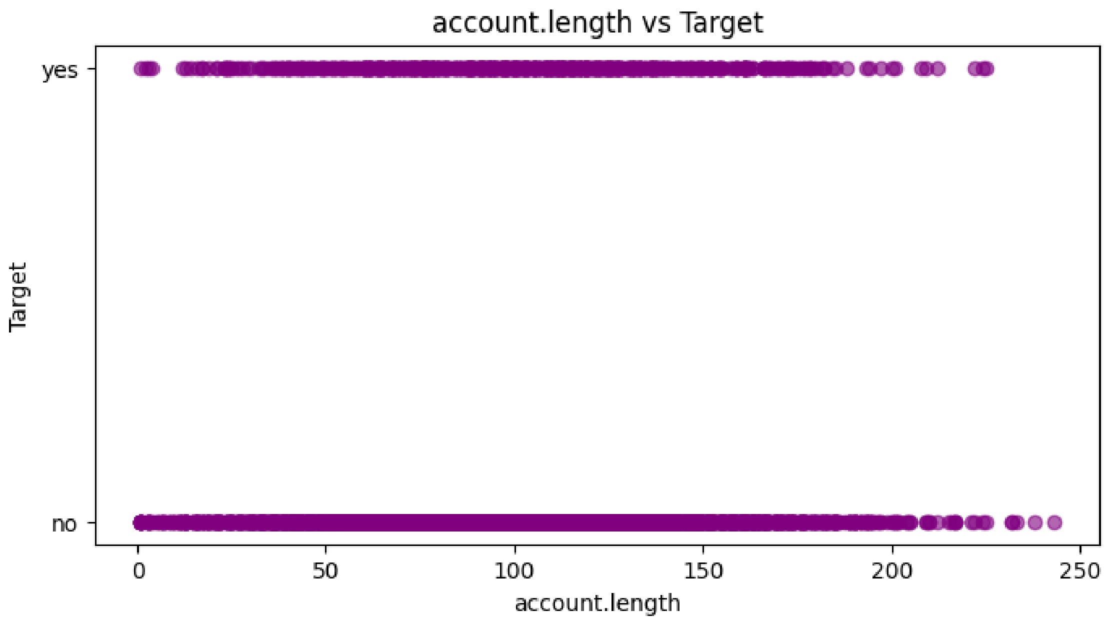
[ ]:

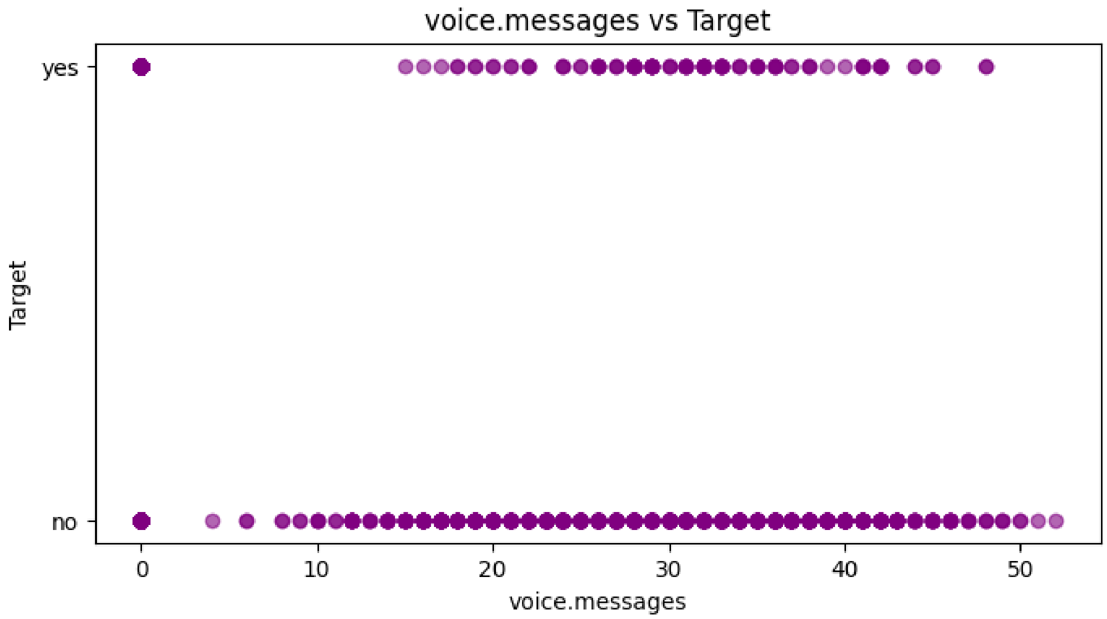
```
#### Boxplot for the Dataset
### Showing the Outliers in the dataset , but only for numerical columns
features.boxplot(figsize=(15,10),color='blue')
plt.suptitle("Boxplot for the Dataset")
plt.tight_layout()
plt.show()
```

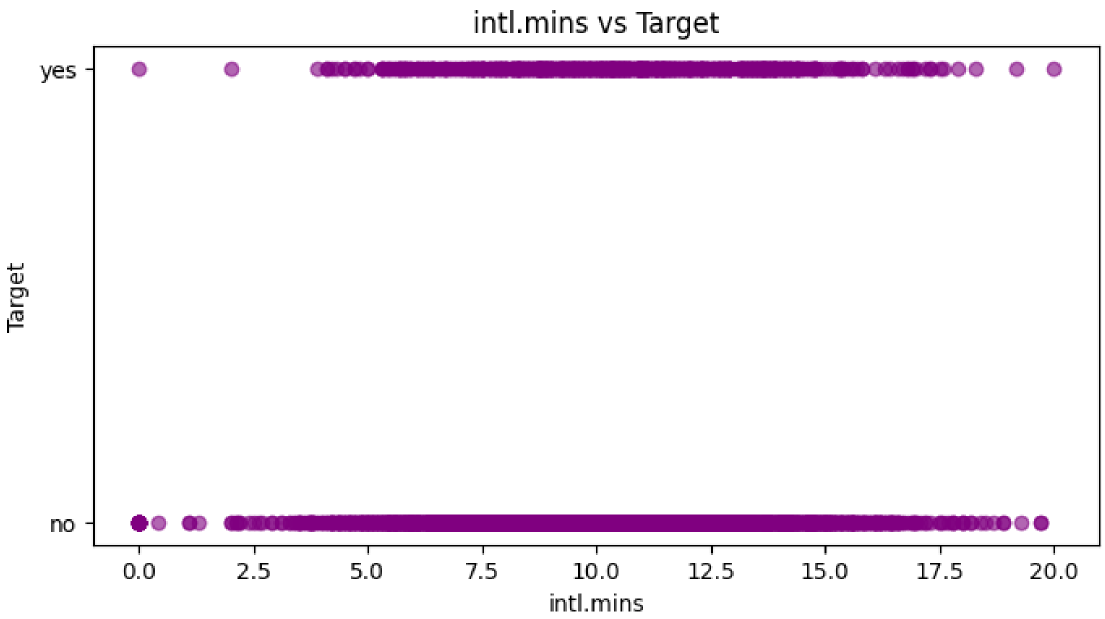


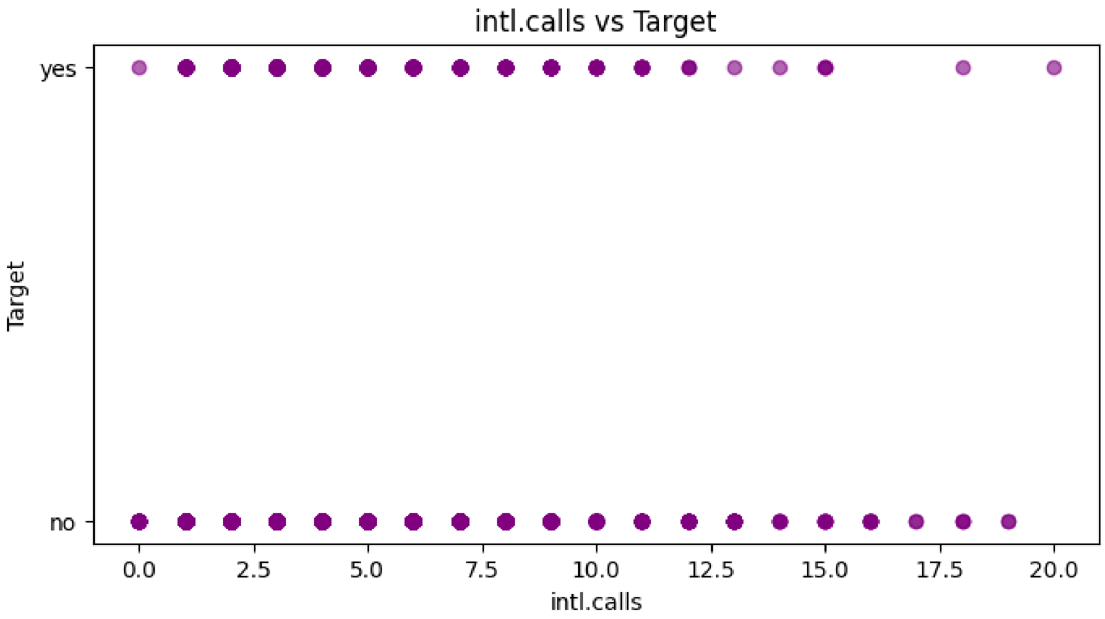
[ ]:

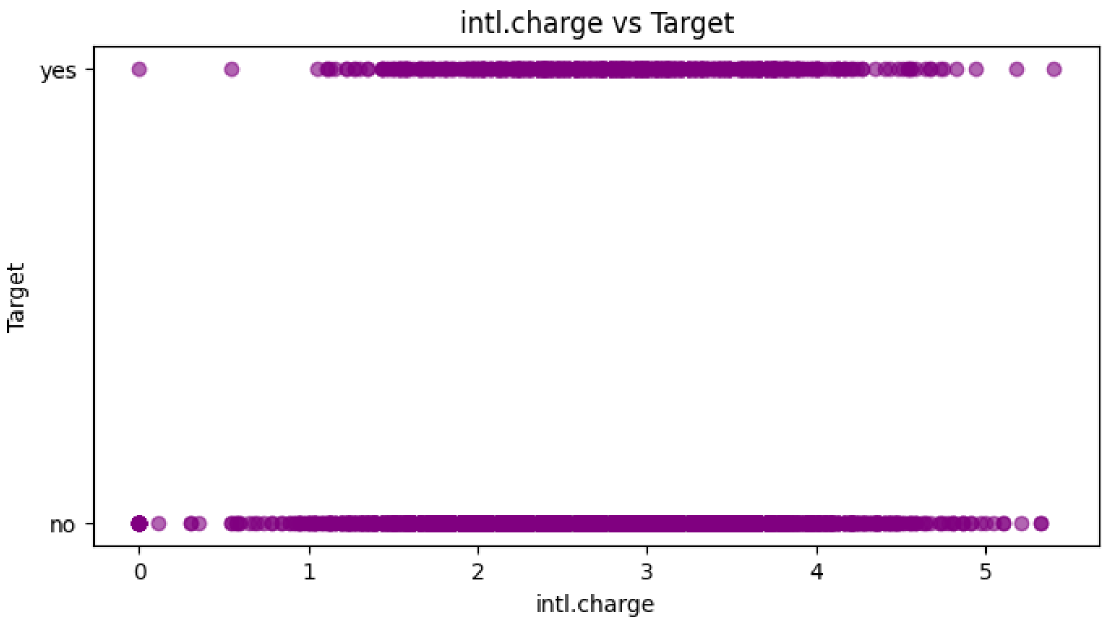
```
### Scatter plot shows relationshops with target
for column in features.select_dtypes(include=['float64', 'int64']).columns:
    plt.figure(figsize=(8, 4))
    plt.scatter(features[column], df['churn'], alpha=0.6, color='purple')
    plt.title(f"{column} vs Target")
    plt.xlabel(column)
    plt.ylabel('Target')
    plt.show()
```

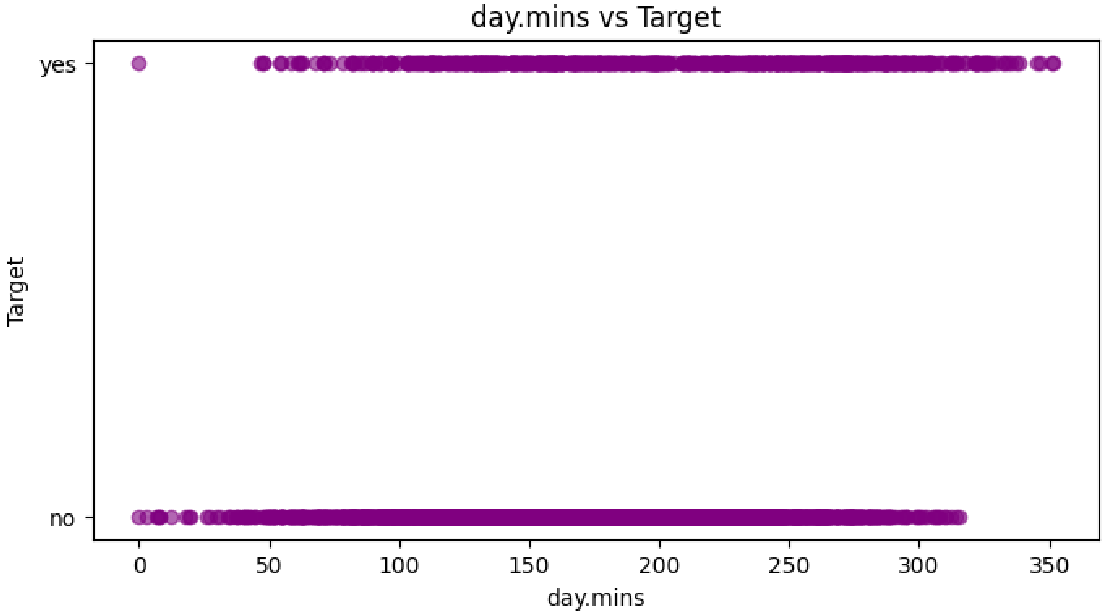

account.length vs Target
yes
no
Target
0
50
100
150
200
250
account.length

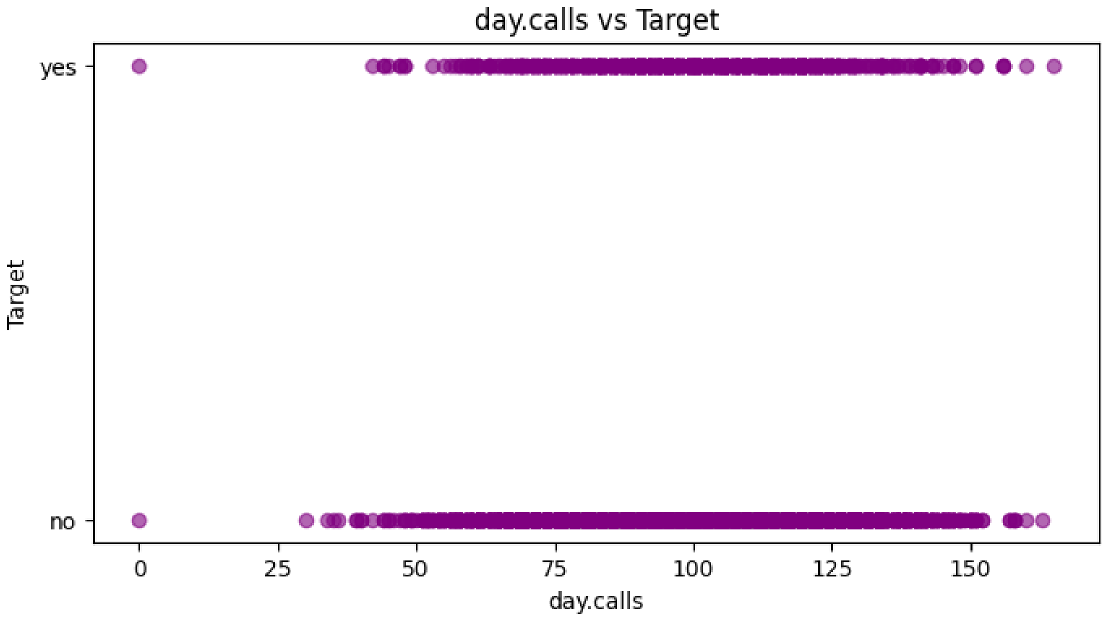

voice.messages vs Target
yes
no
Target
0
10
20
30
40
50
voice.messages

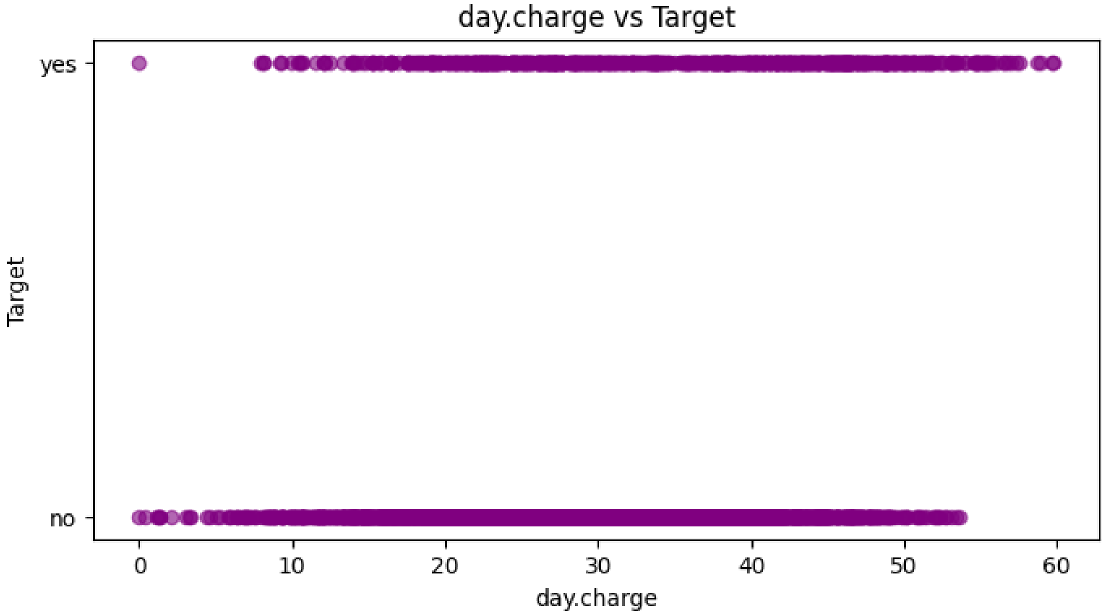

intl.mins vs Target
yes
no
Target
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
intl.mins

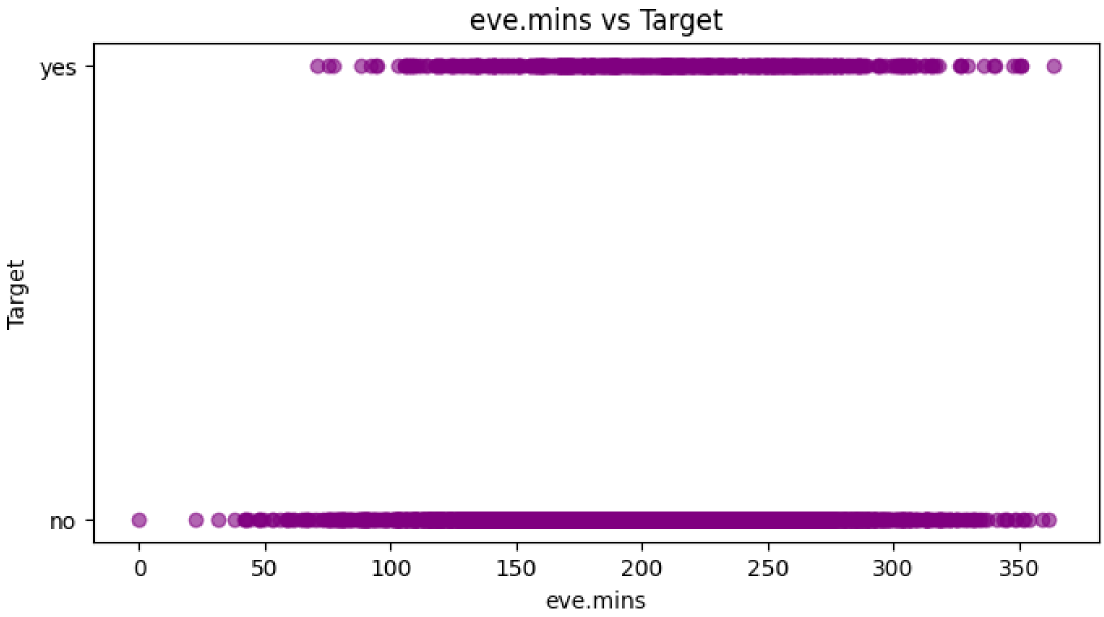

intl.calls vs Target
yes
no
Target
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
intl.calls

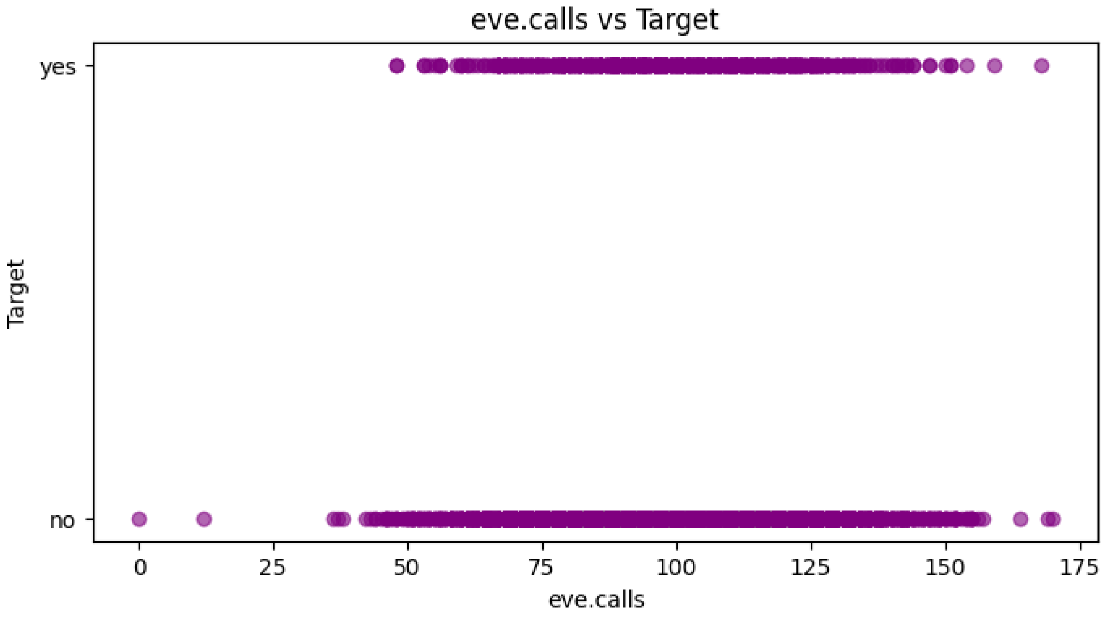

intl.charge vs Target
yes
no
Target
0
1
2
3
4
5
intl.charge

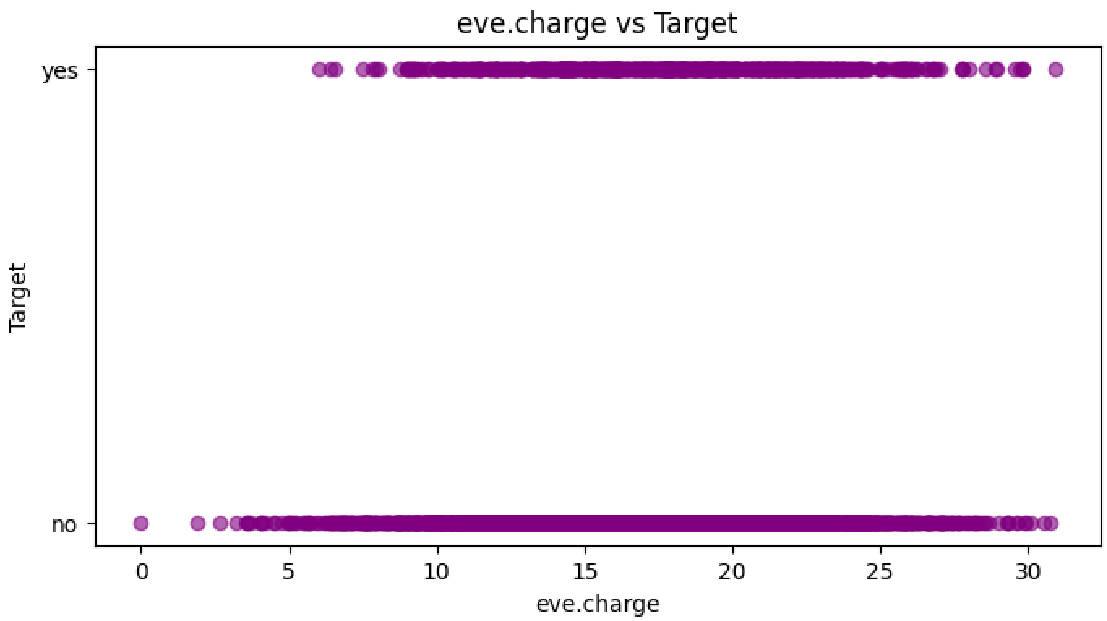

day.mins vs Target
yes
no
Target
0
50
100
150
200
250
300
350
day.mins

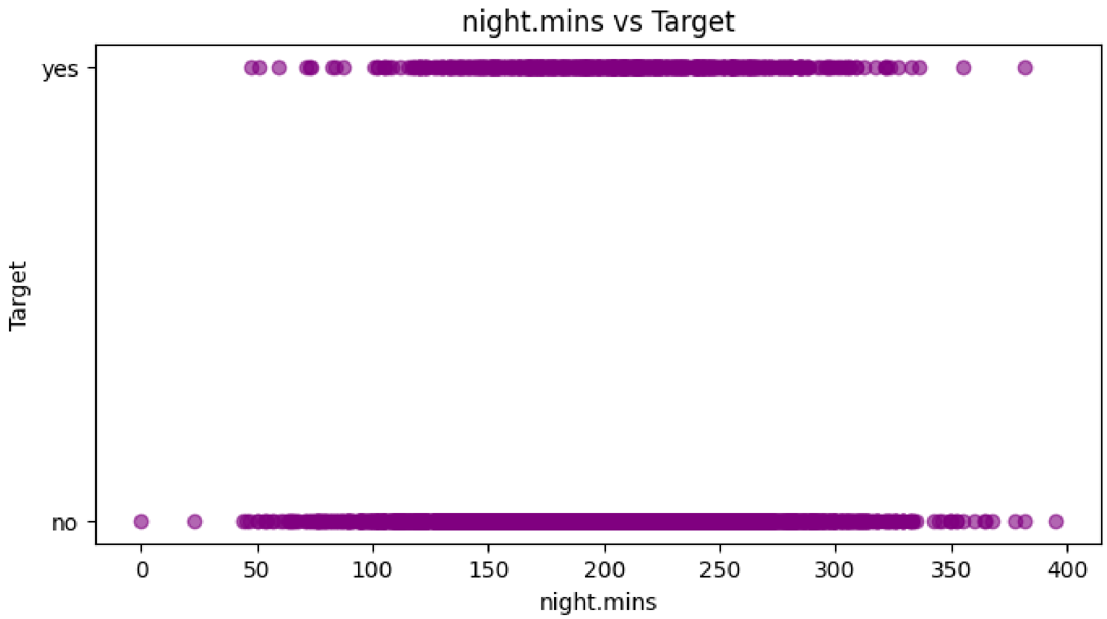

day.calls vs Target
yes
no
Target
0
25
50
75
100
125
150
day.calls

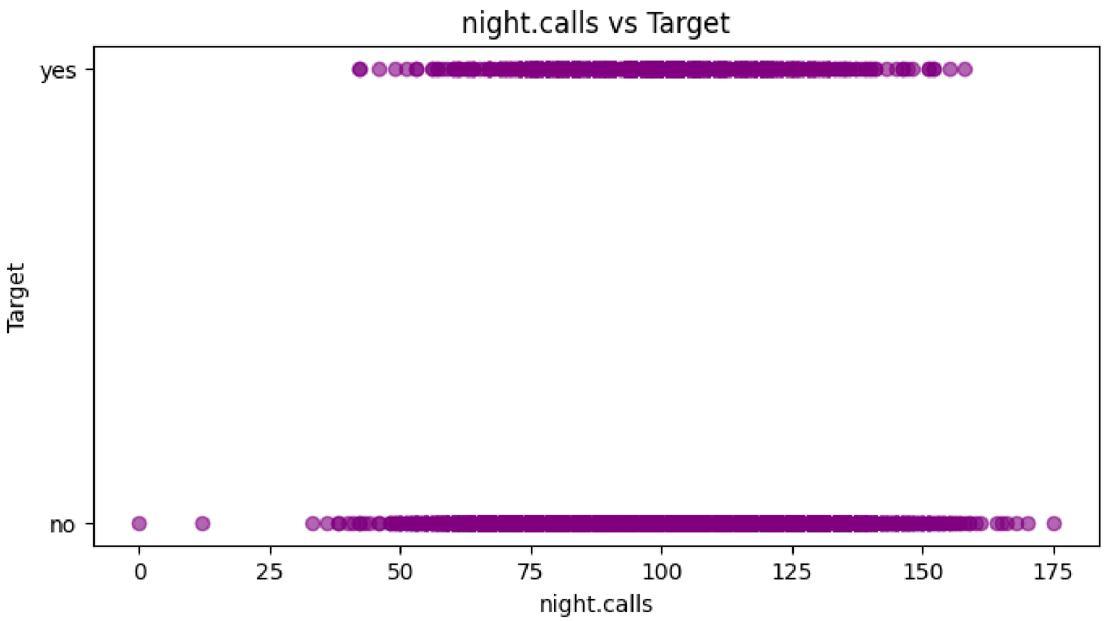

day.charge vs Target
yes
no
Target
0
10
20
30
40
50
60
day.charge

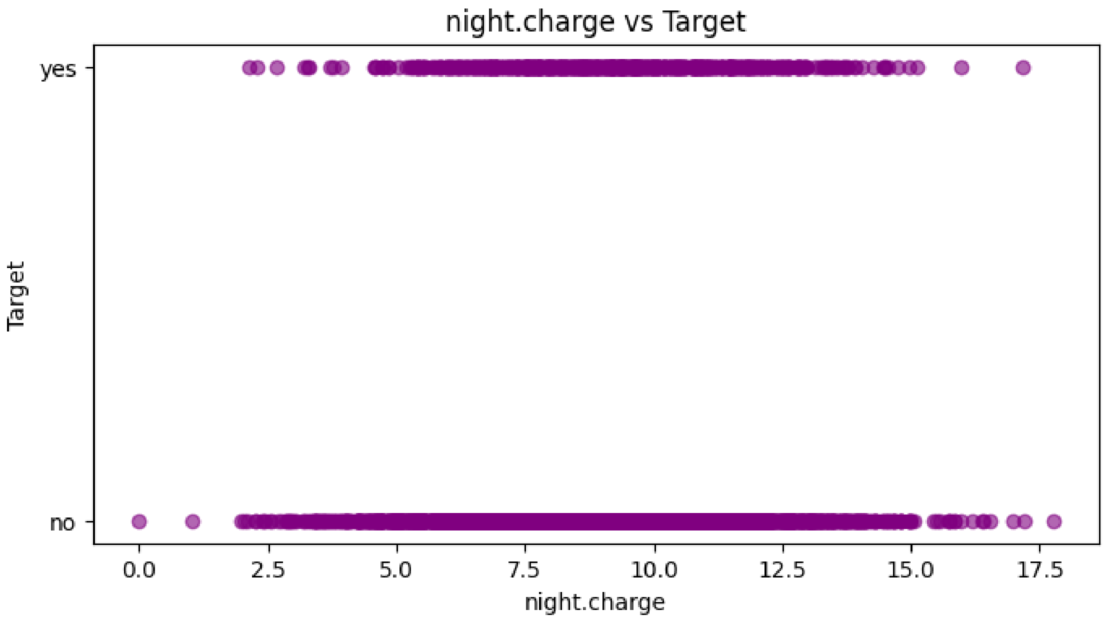

eve.mins vs Target
Target
yes
no
0
50
100
150
200
250
300
350
eve.mins

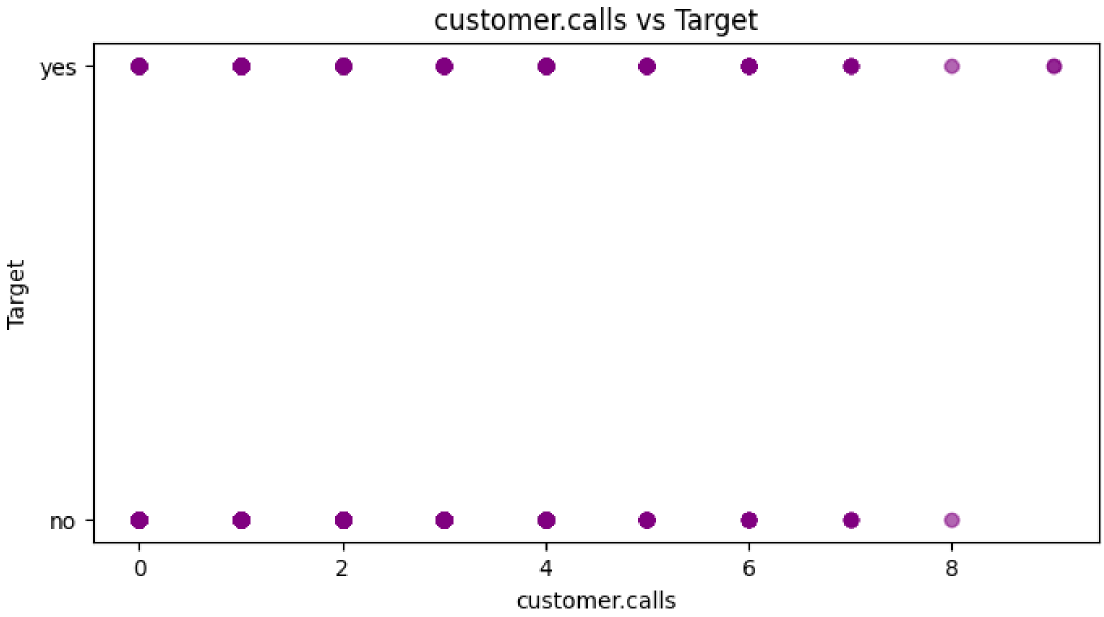

eve.calls vs Target
Target
yes
no
0
25
50
75
100
125
150
175
eve.calls


eve.charge vs Target
yes
no
Target
0
5
10
15
20
25
30
eve.charge

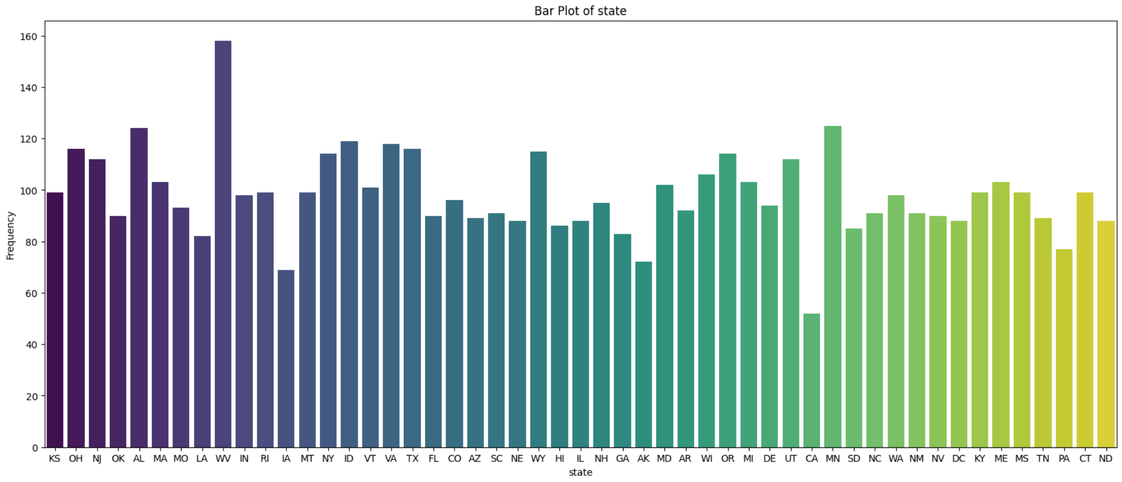

night.mins vs Target
yes
no
Target
0
50
100
150
200
250
300
350
400
night.mins

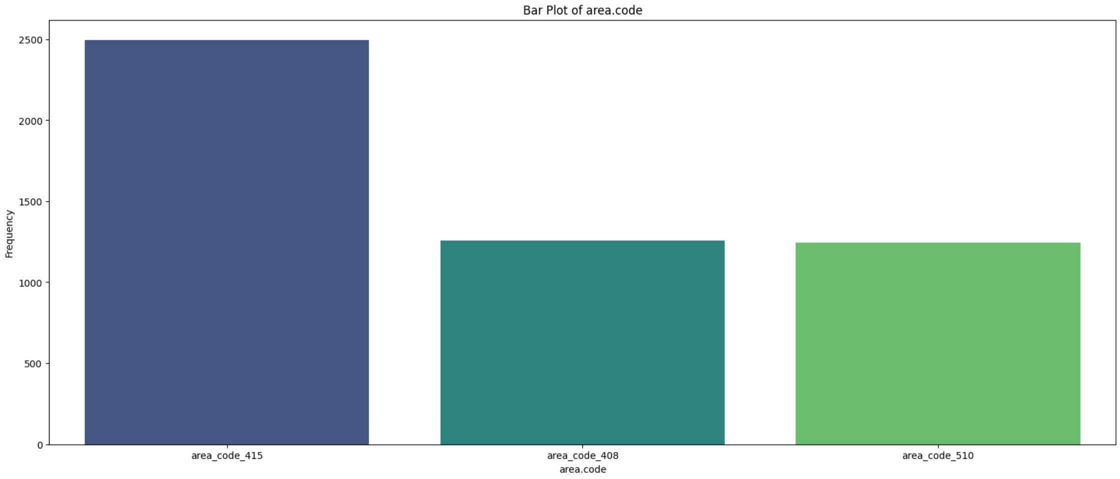

night.calls vs Target
Target
yes
no
0
25
50
75
100
125
150
175
night.calls

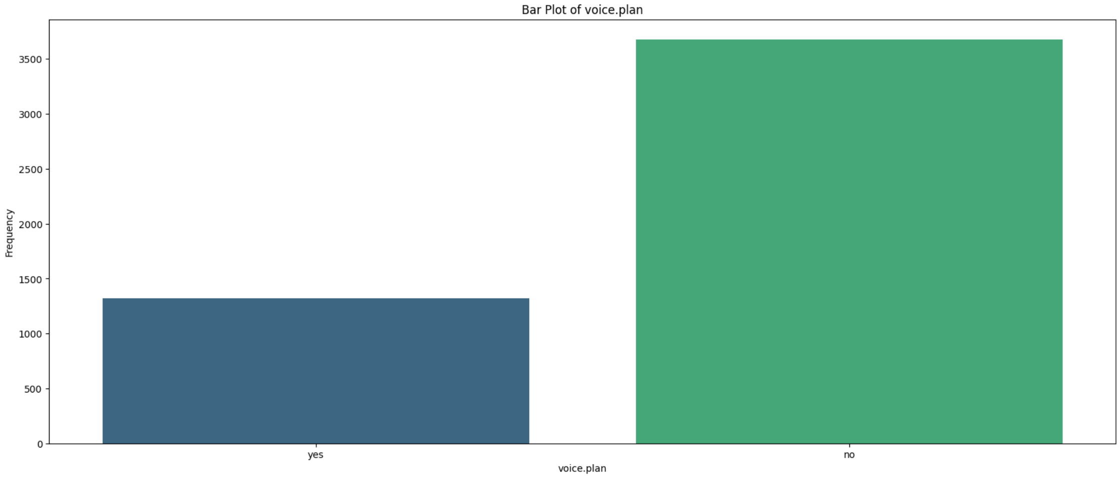

night.charge vs Target
Target
yes
no
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
night.charge

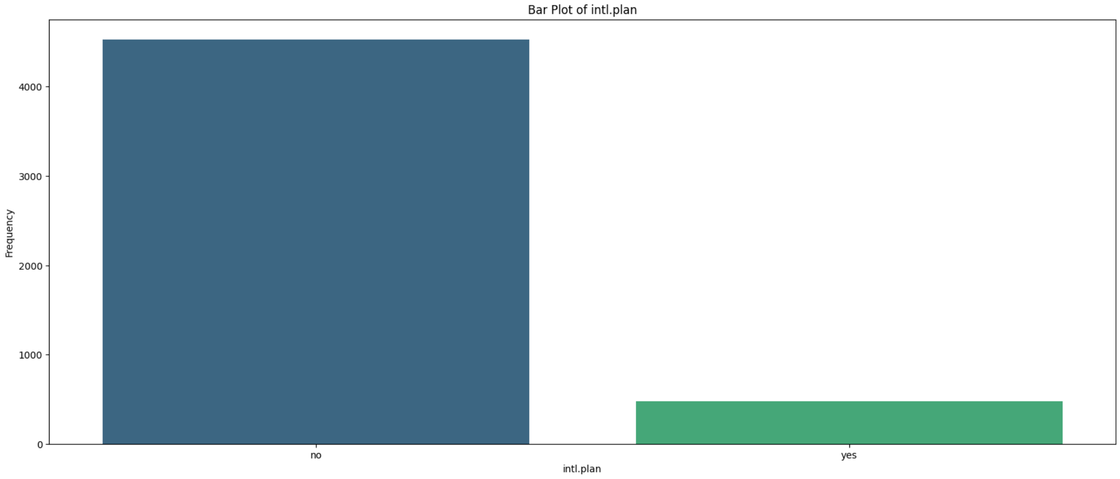


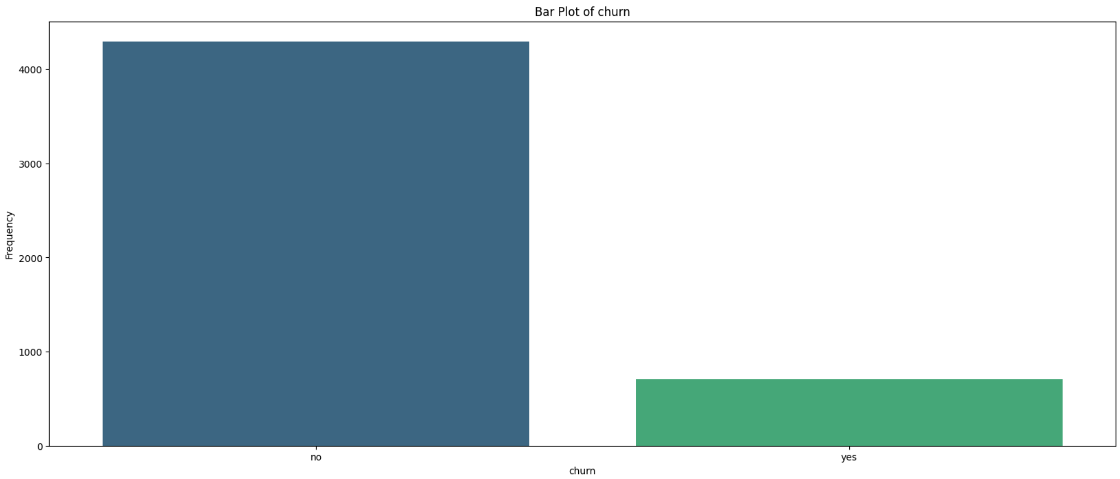
```
[ ]: ### Bar Plot to check the Frequency

     for column in df.select_dtypes(include=['category', 'object']).columns:
         plt.figure(figsize=(20,8))
         sns.countplot(x=df[column], palette='viridis')
         plt.title(f'Bar Plot of {column}')
         plt.xlabel(column)
         plt.ylabel('Frequency')
         plt.show()
```




Bar Plot of area.code
Frequency
0
500
1000
1500
2000
2500
area_code_415
area_code_408
area_code_510
area.code

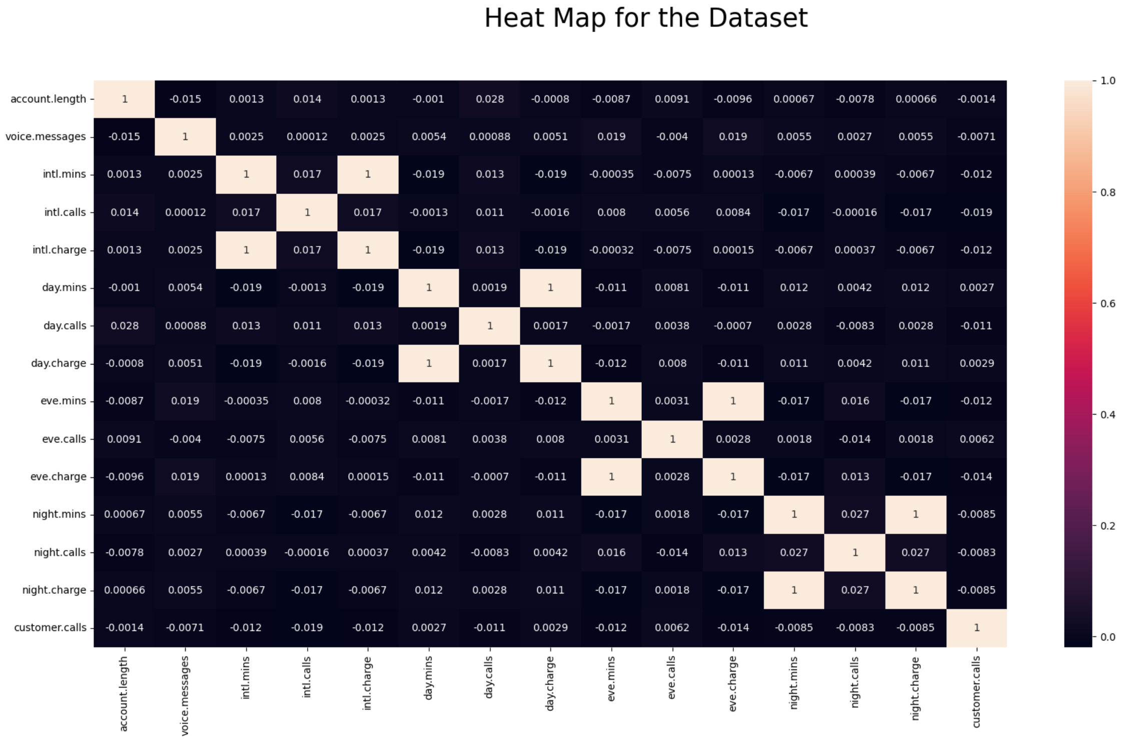

Bar Plot of voice.plan
Frequency
0
500
1000
1500
2000
2500
3000
3500
yes
no
voice.plan


Bar Plot of intl.plan
Frequency
0
1000
2000
3000
4000
no
yes
intl.plan



```
[ ]: num_col1=[i for i in features.columns if features[i].dtype!='O']
     plt.figure(figsize=(20,10))
     sns.heatmap(features[num_col1].corr(),annot=True)
     plt.suptitle("Heat Map for the Dataset",size=25)
```

```
[ ]: Text(0.5, 0.98, 'Heat Map for the Dataset')
```



| | account.length | voice.messages | intl.mins | intl.calls | intl.charge | day.mins | day.calls | day.charge | eve.mins | eve.calls | eve.charge | night.mins | night.calls | night.charge | customer.calls |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| account.length | 1 | -0.015 | 0.0013 | 0.014 | 0.0013 | -0.001 | 0.028 | -0.0008 | -0.0087 | 0.0091 | -0.0096 | 0.00067 | -0.0078 | 0.00066 | -0.0014 |
| voice.messages | -0.015 | 1 | 0.0025 | 0.00012 | 0.0025 | 0.0054 | 0.00088 | 0.0051 | 0.019 | -0.004 | 0.019 | 0.0055 | 0.0027 | 0.0055 | -0.0071 |
| intl.mins | 0.0013 | 0.0025 | 1 | 0.017 | 1 | -0.019 | 0.013 | -0.019 | -0.00035 | -0.0075 | 0.00013 | -0.0067 | 0.00039 | -0.0067 | -0.012 |
| intl.calls | 0.014 | 0.00012 | 0.017 | 1 | 0.017 | -0.0013 | 0.011 | -0.0016 | 0.008 | 0.0056 | 0.0084 | -0.017 | -0.00016 | -0.017 | -0.019 |
| intl.charge | 0.0013 | 0.0025 | 1 | 0.017 | 1 | -0.019 | 0.013 | -0.019 | -0.00032 | -0.0075 | 0.00015 | -0.0067 | 0.00037 | -0.0067 | -0.012 |
| day.mins | -0.001 | 0.0054 | -0.019 | -0.0013 | -0.019 | 1 | 0.0019 | 1 | -0.011 | 0.0081 | -0.011 | 0.012 | 0.0042 | 0.012 | 0.0027 |
| day.calls | 0.028 | 0.00088 | 0.013 | 0.011 | 0.013 | 0.0019 | 1 | 0.0017 | -0.0017 | 0.0038 | -0.0007 | 0.0028 | -0.0083 | 0.0028 | -0.011 |
| day.charge | -0.0008 | 0.0051 | -0.019 | -0.0016 | -0.019 | 1 | 0.0017 | 1 | -0.012 | 0.008 | -0.011 | 0.011 | 0.0042 | 0.011 | 0.0029 |
| eve.mins | -0.0087 | 0.019 | -0.00035 | 0.008 | -0.00032 | -0.011 | -0.0017 | -0.012 | 1 | 0.0031 | 1 | -0.017 | 0.016 | -0.017 | -0.012 |
| eve.calls | 0.0091 | -0.004 | -0.0075 | 0.0056 | -0.0075 | 0.0081 | 0.0038 | 0.008 | 0.0031 | 1 | 0.0028 | 0.0018 | -0.014 | 0.0018 | 0.0062 |
| eve.charge | -0.0096 | 0.019 | 0.00013 | 0.0084 | 0.00015 | -0.011 | -0.0007 | -0.011 | 1 | 0.0028 | 1 | -0.017 | 0.013 | -0.017 | -0.014 |
| night.mins | 0.00067 | 0.0055 | -0.0067 | -0.017 | -0.0067 | 0.012 | 0.0028 | 0.011 | -0.017 | 0.0018 | -0.017 | 1 | 0.027 | 1 | -0.0085 |
| night.calls | -0.0078 | 0.0027 | 0.00039 | -0.00016 | 0.00037 | 0.0042 | -0.0083 | 0.0042 | 0.016 | -0.014 | 0.013 | 0.027 | 1 | 0.027 | -0.0083 |
| night.charge | 0.00066 | 0.0055 | -0.0067 | -0.017 | -0.0067 | 0.012 | 0.0028 | 0.011 | -0.017 | 0.0018 | -0.017 | 1 | 0.027 | 1 | -0.0085 |
| customer.calls | -0.0014 | -0.0071 | -0.012 | -0.019 | -0.012 | 0.0027 | -0.011 | 0.0029 | -0.012 | 0.0062 | -0.014 | -0.0085 | -0.0083 | -0.0085 | 1 |

```
[ ]: features.info()          ### Checking the Information
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 18 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   area.code       5000 non-null   object
 1   account.length  5000 non-null   int64
 2   voice.plan      5000 non-null   object
 3   voice.messages  5000 non-null   int64
 4   intl.plan       5000 non-null   object
 5   intl.mins       5000 non-null   float64
 6   intl.calls      5000 non-null   int64
 7   intl.charge     5000 non-null   float64
 8   day.mins        5000 non-null   float64
 9   day.calls       5000 non-null   int64
 10  day.charge      4993 non-null   float64
 11  eve.mins        4976 non-null   float64
 12  eve.calls       5000 non-null   int64
 13  eve.charge      5000 non-null   float64
 14  night.mins      5000 non-null   float64
 15  night.calls     5000 non-null   int64
 16  night.charge    5000 non-null   float64
 17  customer.calls  5000 non-null   int64
dtypes: float64(8), int64(7), object(3)
memory usage: 703.2+ KB
```

```
[ ]: ### To get the unique values for each column
     for column in features.select_dtypes(include=['int64','float64','object']).
      ↪columns:
         k=features[column].nunique()
         print(f'{column}:{k}')
```

```
area.code:3
account.length:218
voice.plan:2
voice.messages:48
intl.plan:2
intl.mins:170
intl.calls:21
intl.charge:170
day.mins:1961
day.calls:123
day.charge:1961
eve.mins:1876
eve.calls:126
eve.charge:1659
night.mins:1853
night.calls:131
night.charge:1028
customer.calls:10
```

### 0.0.3 Feature Engineering

```
[ ]: from sklearn.impute import SimpleImputer     ### To handle the Missing Values
     from sklearn.compose import ColumnTransformer       ### To handle different␣
      ↪data types and handling the pipline
     from sklearn.pipeline import make_pipeline     ### we can use different encoding␣
      ↪methods
     from sklearn.preprocessing import OrdinalEncoder     ### we can change the␣
      ↪categorical data to Numerical data using this
```

```
[ ]: features.isnull().sum()         ## To Check the Null Values
```

```
[ ]: area.code          0
     account.length     0
     voice.plan         0
     voice.messages     0
     intl.plan          0
     intl.mins          0
     intl.calls         0
     intl.charge        0
     day.mins           0
     day.calls          0
     day.charge         7
     eve.mins          24
     eve.calls          0
     eve.charge         0
     night.mins         0
     night.calls        0
     night.charge       0
     customer.calls     0
     dtype: int64
```

```
[ ]: features.isna().any().any()     ### To check the NAN values
```

```
[ ]: True
```

```
[ ]: features.duplicated().sum()     ### Checking the Duplicated Values
     ### There are no duplicated values
```

```
[ ]: 0
[ ]: ### Dividing the categorical columns and numerical columns separately
     cat_col=[i for i in features.columns if features[i].dtype=='O']
     num_col=[i for i in features.columns if features[i].dtype!='O']
```

```
[ ]: ### Encoding the categorical data to Numerical data
     ord=make_pipeline(OrdinalEncoder())
     trans=ColumnTransformer([('ord',ord,cat_col)])
     df_cat=pd.DataFrame(trans.fit_transform(features[cat_col]),columns=trans.
      ↪get_feature_names_out())
```

```
[ ]: features[num_col]
```

```
[ ]:       account.length  voice.messages  intl.mins  intl.calls  intl.charge  \
     0                128              25       10.0           3         2.70
     1                107              26       13.7           3         3.70
     2                137               0       12.2           5         3.29
     3                 84               0        6.6           7         1.78
     4                 75               0       10.1           3         2.73
     …                  …               …          …           …            …
     4995              50              40        9.9           5         2.67
     4996             152               0       14.7           2         3.97
     4997              61               0       13.6           4         3.67
     4998             109               0        8.5           6         2.30
     4999              86              34        9.3          16         2.51

           day.mins  day.calls  day.charge  eve.mins  eve.calls  eve.charge  \
     0        265.1        110       45.07     197.4         99       16.78
     1        161.6        123       27.47     195.5        103       16.62
     2        243.4        114       41.38     121.2        110       10.30
     3        299.4         71       50.90      61.9         88        5.26
     4        166.7        113       28.34     148.3        122       12.61
     …            …          …           …         …          …           …
     4995     235.7        127       40.07     223.0        126       18.96
     4996     184.2         90       31.31     256.8         73       21.83
     4997     140.6         89       23.90     172.8        128       14.69
     4998     188.8         67       32.10     171.7         92       14.59
     4999     129.4        102       22.00     267.1        104       22.70

           night.mins  night.calls  night.charge  customer.calls
     0          244.7           91         11.01               1
     1          254.4          103         11.45               1
     2          162.6          104          7.32               0
     3          196.9           89          8.86               2
     4          186.9          121          8.41               3
     …              …            …             …               …
     4995       297.5          116         13.39               2
     4996       213.6          113          9.61               3
```

```
4997       212.4         97         9.56               1
4998       224.4         89        10.10               0
4999       154.8        100         6.97               0

[5000 rows x 15 columns]
```

```
[ ]: ### Replacing the NAN or missing values in the dataset using median
     simple=SimpleImputer(missing_values=np.nan,strategy='median')
     df_num=pd.DataFrame(simple.
      ↪fit_transform(features[num_col]),columns=features[num_col].columns)
```

```
[ ]: df_cat       ### converted categorical columns
```

```
[ ]:       ord__area.code  ord__voice.plan  ord__intl.plan
     0                1.0              1.0             0.0
     1                1.0              1.0             0.0
     2                1.0              0.0             0.0
     3                0.0              0.0             1.0
     4                1.0              0.0             1.0
     …                  …                …               …
     4995             0.0              1.0             0.0
     4996             1.0              0.0             0.0
     4997             1.0              0.0             0.0
     4998             2.0              0.0             0.0
     4999             1.0              1.0             0.0

     [5000 rows x 3 columns]
```

```
[ ]: df_num
```

```
[ ]:       account.length  voice.messages  intl.mins  intl.calls  intl.charge  \
     0              128.0            25.0       10.0         3.0         2.70
     1              107.0            26.0       13.7         3.0         3.70
     2              137.0             0.0       12.2         5.0         3.29
     3               84.0             0.0        6.6         7.0         1.78
     4               75.0             0.0       10.1         3.0         2.73
     …                  …               …          …           …            …
     4995            50.0            40.0        9.9         5.0         2.67
     4996           152.0             0.0       14.7         2.0         3.97
     4997            61.0             0.0       13.6         4.0         3.67
     4998           109.0             0.0        8.5         6.0         2.30
     4999            86.0            34.0        9.3        16.0         2.51

           day.mins  day.calls  day.charge  eve.mins  eve.calls  eve.charge  \
     0        265.1      110.0       45.07     197.4       99.0       16.78
     1        161.6      123.0       27.47     195.5      103.0       16.62
     2        243.4      114.0       41.38     121.2      110.0       10.30
```

```
3        299.4       71.0       50.90      61.9       88.0        5.26
4        166.7      113.0       28.34     148.3      122.0       12.61
…        …        …          …       …        …         …
4995     235.7      127.0       40.07     223.0      126.0       18.96
4996     184.2       90.0       31.31     256.8       73.0       21.83
4997     140.6       89.0       23.90     172.8      128.0       14.69
4998     188.8       67.0       32.10     171.7       92.0       14.59
4999     129.4      102.0       22.00     267.1      104.0       22.70

      night.mins  night.calls  night.charge  customer.calls
0          244.7         91.0        11.01             1.0
1          254.4        103.0        11.45             1.0
2          162.6        104.0         7.32             0.0
3          196.9         89.0         8.86             2.0
4          186.9        121.0         8.41             3.0
…          …          …           …             …
4995       297.5        116.0        13.39             2.0
4996       213.6        113.0         9.61             3.0
4997       212.4         97.0         9.56             1.0
4998       224.4         89.0        10.10             0.0
4999       154.8        100.0         6.97             0.0

[5000 rows x 15 columns]
```

```
[ ]: df=pd.concat([df_num,df_cat],axis=1)
```

```
[ ]: df
```

```
[ ]:       account.length  voice.messages  intl.mins  intl.calls  intl.charge  \
     0              128.0            25.0       10.0        3.0         2.70
     1              107.0            26.0       13.7        3.0         3.70
     2              137.0             0.0       12.2        5.0         3.29
     3               84.0             0.0        6.6        7.0         1.78
     4               75.0             0.0       10.1        3.0         2.73
     …              …             …        …          …          …
     4995            50.0            40.0        9.9        5.0         2.67
     4996           152.0             0.0       14.7        2.0         3.97
     4997            61.0             0.0       13.6        4.0         3.67
     4998           109.0             0.0        8.5        6.0         2.30
     4999            86.0            34.0        9.3       16.0         2.51

           day.mins  day.calls  day.charge  eve.mins  eve.calls  eve.charge  \
     0        265.1      110.0       45.07     197.4       99.0       16.78
     1        161.6      123.0       27.47     195.5      103.0       16.62
     2        243.4      114.0       41.38     121.2      110.0       10.30
     3        299.4       71.0       50.90      61.9       88.0        5.26
     4        166.7      113.0       28.34     148.3      122.0       12.61
```

```
…        …        …         …       …        …         …
4995     235.7    127.0     40.07   223.0    126.0     18.96
4996     184.2     90.0     31.31   256.8     73.0     21.83
4997     140.6     89.0     23.90   172.8    128.0     14.69
4998     188.8     67.0     32.10   171.7     92.0     14.59
4999     129.4    102.0     22.00   267.1    104.0     22.70

      night.mins  night.calls  night.charge  customer.calls  ord__area.code  \
0          244.7         91.0         11.01             1.0             1.0
1          254.4        103.0         11.45             1.0             1.0
2          162.6        104.0          7.32             0.0             1.0
3          196.9         89.0          8.86             2.0             0.0
4          186.9        121.0          8.41             3.0             1.0
…          …          …            …             …              …
4995       297.5        116.0         13.39             2.0             0.0
4996       213.6        113.0          9.61             3.0             1.0
4997       212.4         97.0          9.56             1.0             1.0
4998       224.4         89.0         10.10             0.0             2.0
4999       154.8        100.0          6.97             0.0             1.0

      ord__voice.plan  ord__intl.plan
0                 1.0             0.0
1                 1.0             0.0
2                 0.0             0.0
3                 0.0             1.0
4                 0.0             1.0
…                …             …
4995              1.0             0.0
4996              0.0             0.0
4997              0.0             0.0
4998              0.0             0.0
4999              1.0             0.0

[5000 rows x 18 columns]
```

[ ]:
```
df.duplicated().sum()  ### There are no duplicated values
```

[ ]:
```
0
```

[ ]:
```
df.isnull().sum()         ### There are no missing values
```

[ ]:
```
account.length     0
voice.messages     0
intl.mins          0
intl.calls         0
intl.charge        0
day.mins           0
day.calls          0
day.charge         0
eve.mins           0
eve.calls          0
eve.charge         0
night.mins         0
night.calls        0
night.charge       0
customer.calls     0
ord__area.code     0
ord__voice.plan    0
ord__intl.plan     0
dtype: int64
```

```
[ ]: df.isna().any()   #### There are no NAN values
```

```
[ ]: account.length     False
     voice.messages     False
     intl.mins          False
     intl.calls         False
     intl.charge        False
     day.mins           False
     day.calls          False
     day.charge         False
     eve.mins           False
     eve.calls          False
     eve.charge         False
     night.mins         False
     night.calls        False
     night.charge       False
     customer.calls     False
     ord__area.code     False
     ord__voice.plan    False
     ord__intl.plan     False
     dtype: bool
```

```
[ ]: ## Removing the Outliers from the dataset
     for i in range(2):
             initial_size = len(df)

             # Calculate IQR for each numerical column
             for col in df.select_dtypes(include=np.number).columns:
                 Q1 = df[col].quantile(0.25)
                 Q3 = df[col].quantile(0.75)
                 IQR = Q3 - Q1
                 lower_bound = Q1 - 1.5 * IQR
                 upper_bound = Q3 + 1.5 * IQR
```

```
            df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
```

```
[ ]: df
```

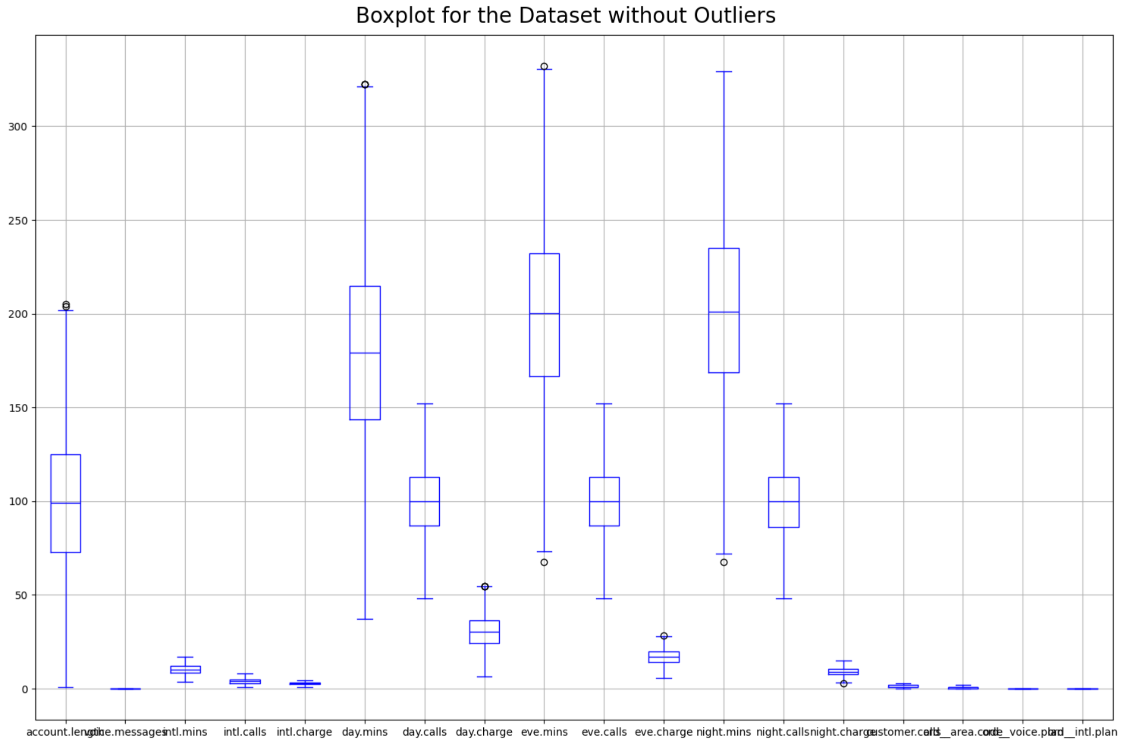
```
[ ]:       account.length  voice.messages  intl.mins  intl.calls  intl.charge  \
     2              137.0             0.0       12.2         5.0         3.29
     11              74.0             0.0        9.1         5.0         2.46
     12             168.0             0.0       11.2         2.0         3.02
     13              95.0             0.0       12.3         5.0         3.32
     17              93.0             0.0        8.1         3.0         2.19
     …                 …               …         …           …           …
     4993            73.0             0.0       11.5         6.0         3.11
     4994            75.0             0.0        6.9         7.0         1.86
     4996           152.0             0.0       14.7         2.0         3.97
     4997            61.0             0.0       13.6         4.0         3.67
     4998           109.0             0.0        8.5         6.0         2.30

           day.mins  day.calls  day.charge  eve.mins  eve.calls  eve.charge  \
     2        243.4      114.0       41.38     121.2      110.0       10.30
     11       187.7      127.0       31.91     163.4      148.0       13.89
     12       128.8       96.0       30.62     104.9       71.0        8.92
     13       156.6       88.0       26.62     247.6       75.0       21.05
     17       190.7      114.0       32.42     218.2      111.0       18.55
     …          …         …           …         …          …          …
     4993     177.9       89.0       30.24     131.2       82.0       11.15
     4994     170.7      101.0       29.02     193.1      126.0       16.41
     4996     184.2       90.0       31.31     256.8       73.0       21.83
     4997     140.6       89.0       23.90     172.8      128.0       14.69
     4998     188.8       67.0       32.10     171.7       92.0       14.59

           night.mins  night.calls  night.charge  customer.calls  ord__area.code  \
     2          162.6        104.0          7.32             0.0             1.0
     11         196.0         94.0          8.82             0.0             1.0
     12         141.1        128.0          6.35             1.0             0.0
     13         192.3        115.0          8.65             3.0             2.0
     17         129.6        121.0          5.83             3.0             2.0
     …            …            …             …              …               …
     4993       186.2         89.0          8.38             3.0             0.0
     4994       129.1        104.0          5.81             1.0             0.0
     4996       213.6        113.0          9.61             3.0             1.0
     4997       212.4         97.0          9.56             1.0             1.0
     4998       224.4         89.0         10.10             0.0             2.0

           ord__voice.plan  ord__intl.plan
     2                 0.0             0.0
     11                0.0             0.0
     12                0.0             0.0
```

```
13                  0.0             0.0
17                  0.0             0.0
…                …             …
4993                0.0             0.0
4994                0.0             0.0
4996                0.0             0.0
4997                0.0             0.0
4998                0.0             0.0

[2653 rows x 18 columns]
```

```
[ ]: #def box_plot(df,columns):
        # threshold=2
         #z_score=stats.zscore(df[columns])
       #  return df[(abs(z_score)<=threshold).all(axis=1)]
```

```
[ ]: ##box_plot(df,df.columns)
```

```
[ ]: df.boxplot(figsize=(15,10),color='blue')
     plt.suptitle("Boxplot for the Dataset without Outliers",size=20,color='black')
     plt.tight_layout()
     plt.show()
```



```
[ ]: ## Standardizing the data from original dataset
     from sklearn.preprocessing import StandardScaler
     std=StandardScaler()
     df1=pd.DataFrame(std.fit_transform(df),columns=df.columns)     ## by␣
      ↪Standardizing the data the mean will be zero and standard deviation will␣
      ↪becomes to 1
```

```
[ ]: df
```

```
[ ]:       account.length  voice.messages  intl.mins  intl.calls  intl.charge  \
     2              137.0             0.0       12.2         5.0         3.29
     11              74.0             0.0        9.1         5.0         2.46
     12             168.0             0.0       11.2         2.0         3.02
     13              95.0             0.0       12.3         5.0         3.32
     17              93.0             0.0        8.1         3.0         2.19
     …                  …               …          …           …            …
     4993            73.0             0.0       11.5         6.0         3.11
     4994            75.0             0.0        6.9         7.0         1.86
     4996           152.0             0.0       14.7         2.0         3.97
     4997            61.0             0.0       13.6         4.0         3.67
     4998           109.0             0.0        8.5         6.0         2.30

           day.mins  day.calls  day.charge  eve.mins  eve.calls  eve.charge  \
     2        243.4      114.0       41.38     121.2      110.0       10.30
     11       187.7      127.0       31.91     163.4      148.0       13.89
     12       128.8       96.0       30.62     104.9       71.0        8.92
     13       156.6       88.0       26.62     247.6       75.0       21.05
     17       190.7      114.0       32.42     218.2      111.0       18.55
     …            …          …           …         …          …           …
     4993     177.9       89.0       30.24     131.2       82.0       11.15
     4994     170.7      101.0       29.02     193.1      126.0       16.41
     4996     184.2       90.0       31.31     256.8       73.0       21.83
     4997     140.6       89.0       23.90     172.8      128.0       14.69
     4998     188.8       67.0       32.10     171.7       92.0       14.59

           night.mins  night.calls  night.charge  customer.calls  ord__area.code  \
     2          162.6        104.0          7.32             0.0             1.0
     11         196.0         94.0          8.82             0.0             1.0
     12         141.1        128.0          6.35             1.0             0.0
     13         192.3        115.0          8.65             3.0             2.0
     17         129.6        121.0          5.83             3.0             2.0
     …              …            …             …               …               …
     4993       186.2         89.0          8.38             3.0             0.0
     4994       129.1        104.0          5.81             1.0             0.0
     4996       213.6        113.0          9.61             3.0             1.0
     4997       212.4         97.0          9.56             1.0             1.0
     4998       224.4         89.0         10.10             0.0             2.0
```

```
      ord__voice.plan  ord__intl.plan
2                 0.0             0.0
11                0.0             0.0
12                0.0             0.0
13                0.0             0.0
17                0.0             0.0
…                 …               …
4993              0.0             0.0
4994              0.0             0.0
4996              0.0             0.0
4997              0.0             0.0
4998              0.0             0.0

[2653 rows x 18 columns]
```

```
[ ]: ### Merging the dataset of features and target
     df_data=pd.merge(df1,target,left_index=True,right_index=True)
```

```
[ ]: df_data
```

```
[ ]:       account.length  voice.messages  intl.mins  intl.calls  intl.charge  \
     0           0.969809             0.0   0.747391    0.537221     0.740938
     1          -0.645949             0.0  -0.468755    0.537221    -0.465179
     2           1.764864             0.0   0.355086   -1.154802     0.348587
     3          -0.107363             0.0   0.786622    0.537221     0.784533
     4          -0.158657             0.0  -0.861061   -0.590795    -0.857530
     …                 …               …         …           …          …
     2648       -0.671596             0.0   0.472778    1.101229     0.479371
     2649       -0.620302             0.0  -1.331827    1.665237    -1.337071
     2650        1.354513             0.0   1.728155   -1.154802     1.729083
     2651       -0.979360             0.0   1.296619   -0.026787     1.293137
     2652        0.251694             0.0  -0.704139    1.101229    -0.697683

           day.mins  day.calls  day.charge  eve.mins  eve.calls  eve.charge  \
     0     1.226174   0.741493    1.226104 -1.618567   0.507922   -1.616715
     1     0.157799   1.427583    0.157112 -0.747886   2.484212   -0.747416
     2    -0.971955  -0.208479    0.011494 -1.954873  -1.520375   -1.950875
     3    -0.438727  -0.630688   -0.440034  0.989351  -1.312345    0.986339
     4     0.215341   0.741493    0.214682  0.382762   0.559930    0.380977
     …          …          …          …         …          …           …
     2648 -0.030174  -0.577912   -0.031401 -1.412244  -0.948291   -1.410892
     2649 -0.168276   0.055402   -0.169117 -0.135107   1.340044   -0.137212
     2650  0.090666  -0.525136    0.089383  1.179167  -1.416360    1.175212
     2651 -0.745621  -0.577912   -0.747072 -0.553942   1.444060   -0.553701
     2652  0.178898  -1.738989    0.178559 -0.576638  -0.428215   -0.577915
```

```
      night.mins  night.calls  night.charge  customer.calls  ord__area.code  \
0      -0.789161     0.226653     -0.787808       -1.350649        0.007971
1      -0.101390    -0.290692     -0.101426       -1.350649        0.007971
2      -1.231888     1.468281     -1.231668       -0.338711       -1.401778
3      -0.177580     0.795732     -0.179216        1.685165        1.417720
4      -1.468695     1.106139     -1.469613        1.685165        1.417720
…             …           …              …               …              …
2648   -0.303191    -0.549364     -0.302765        1.685165       -1.401778
2649   -1.478991     0.226653     -1.478765       -0.338711       -1.401778
2650    0.261029     0.692263      0.260068        1.685165        0.007971
2651    0.236318    -0.135489      0.237188       -0.338711        0.007971
2652    0.483422    -0.549364      0.484286       -1.350649        1.417720

      ord__voice.plan  ord__intl.plan churn
0                 0.0             0.0    no
1                 0.0             0.0    no
2                 0.0             0.0    no
3                 0.0             0.0    no
4                 0.0             0.0    no
…                …              …     …
2648              0.0             0.0    no
2649              0.0             0.0    no
2650              0.0             0.0    no
2651              0.0             0.0    no
2652              0.0             0.0    no

[2653 rows x 19 columns]
```

### 0.0.4 Building the Model

[ ]:
```
## Dividing the dataset into features and target for predicting the results
features=df_data.drop(columns='churn')
target=df_data['churn']
```

[ ]:
```
## the features columns
features
```

[ ]:
```
      account.length  voice.messages  intl.mins  intl.calls  intl.charge  \
0           0.969809             0.0   0.747391    0.537221     0.740938
1          -0.645949             0.0  -0.468755    0.537221    -0.465179
2           1.764864             0.0   0.355086   -1.154802     0.348587
3          -0.107363             0.0   0.786622    0.537221     0.784533
4          -0.158657             0.0  -0.861061   -0.590795    -0.857530
…                 …               …          …          …           …
2648       -0.671596             0.0   0.472778    1.101229     0.479371
2649       -0.620302             0.0  -1.331827    1.665237    -1.337071
2650        1.354513             0.0   1.728155   -1.154802     1.729083
```

```
2651       -0.979360             0.0   1.296619   -0.026787     1.293137
2652        0.251694             0.0  -0.704139    1.101229    -0.697683

      day.mins  day.calls  day.charge  eve.mins  eve.calls  eve.charge  \
0     1.226174   0.741493    1.226104 -1.618567   0.507922   -1.616715
1     0.157799   1.427583    0.157112 -0.747886   2.484212   -0.747416
2    -0.971955  -0.208479    0.011494 -1.954873  -1.520375   -1.950875
3    -0.438727  -0.630688   -0.440034  0.989351  -1.312345    0.986339
4     0.215341   0.741493    0.214682  0.382762   0.559930    0.380977
…        …        …         …       …        …         …
2648 -0.030174  -0.577912   -0.031401 -1.412244  -0.948291   -1.410892
2649 -0.168276   0.055402   -0.169117 -0.135107   1.340044   -0.137212
2650  0.090666  -0.525136    0.089383  1.179167  -1.416360    1.175212
2651 -0.745621  -0.577912   -0.747072 -0.553942   1.444060   -0.553701
2652  0.178898  -1.738989    0.178559 -0.576638  -0.428215   -0.577915

      night.mins  night.calls  night.charge  customer.calls  ord__area.code  \
0      -0.789161     0.226653     -0.787808       -1.350649        0.007971
1      -0.101390    -0.290692     -0.101426       -1.350649        0.007971
2      -1.231888     1.468281     -1.231668       -0.338711       -1.401778
3      -0.177580     0.795732     -0.179216        1.685165        1.417720
4      -1.468695     1.106139     -1.469613        1.685165        1.417720
…          …          …           …             …             …
2648   -0.303191    -0.549364     -0.302765        1.685165       -1.401778
2649   -1.478991     0.226653     -1.478765       -0.338711       -1.401778
2650    0.261029     0.692263      0.260068        1.685165        0.007971
2651    0.236318    -0.135489      0.237188       -0.338711        0.007971
2652    0.483422    -0.549364      0.484286       -1.350649        1.417720

      ord__voice.plan  ord__intl.plan
0                 0.0             0.0
1                 0.0             0.0
2                 0.0             0.0
3                 0.0             0.0
4                 0.0             0.0
…               …             …
2648              0.0             0.0
2649              0.0             0.0
2650              0.0             0.0
2651              0.0             0.0
2652              0.0             0.0

[2653 rows x 18 columns]
```

[ ]:

```
### target column
target
```

```
[ ]: 0       no
     1       no
     2       no
     3       no
     4       no
             ..
     2648    no
     2649    no
     2650    no
     2651    no
     2652    no
     Name: churn, Length: 2653, dtype: object
```

```
[ ]: from sklearn.model_selection import train_test_split    ### used for splitting␣
      ↪the data to train and test
```

```
[ ]: ## Splitting the data into x_train,x_testmy_train,y_test
     ## taking the training dataset as 75% and test size as 25%
     ### random_state refers to shuffling the dataset
     x_train,x_test,y_train,y_test=train_test_split(features,target,train_size=0.
      ↪75,random_state=100)
```

```
[ ]: ### checking the sizes of the x_train,x_test,y_train,y_test
     print(x_train.shape)
     print(y_train.shape)
     print(x_test.shape)
     print(y_test.shape)
```

```
(1989, 18)
(1989,)
(664, 18)
(664,)
```

```
[ ]: from sklearn.linear_model import LogisticRegression
     from sklearn.ensemble import BaggingClassifier,RandomForestClassifier
     from sklearn.svm import SVC
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.metrics import accuracy_score,classification_report
```

```
[ ]: models={'Logistic Regression':LogisticRegression(),
            "Bagging Classifier":
      ↪BaggingClassifier(estimator=RandomForestClassifier(),n_estimators=100,max_features=1.
      ↪0,max_samples=0.9),
            'Random Forest Classifier':RandomForestClassifier(),
            'Support Vector Classifier':SVC(),
            'KNeighbors Classifier':KNeighborsClassifier(),
            'Decision Tree Classifier':DecisionTreeClassifier()}      ### Taking the␣
      ↪algorithms into one dictionary form
```

[ ]:
```
results={}
for model_name,model in models.items():
    model.fit(x_train,y_train)
    y_pred=model.predict(x_test)
    accuracy=accuracy_score(y_test,y_pred)
    results[model_name]=accuracy
    accuracy1=(accuracy)*100
```

[ ]:
```
for model_name,accuracy in results.items():
    print(f'{model_name}:{accuracy}')
```

```
Logistic Regression:0.8674698795180723
Bagging Classifier:0.8674698795180723
Random Forest Classifier:0.8674698795180723
Support Vector Classifier:0.8674698795180723
KNeighbors Classifier:0.8373493975903614
Decision Tree Classifier:0.7560240963855421
```

### 0.0.5 Hyper Parameter Tuning

[ ]:
```
param_grid = {
    'Logistic Regression': {'C': [0.1, 1, 10]},
    'Bagging Classifier': {'n_estimators': [50, 100]},
    'Random Forest Classifier': {'n_estimators': [50, 100], 'max_depth': [3, 5,␣
 ↪10]},
    'Support Vector Classifier': {'C': [0.1, 1, 10], 'kernel': ['linear',␣
 ↪'rbf']},
    'KNeighbors Classifier': {'n_neighbors': [3, 5, 7]},
    'Decision Tree Classifier': {'criterion': ['gini', 'entropy'], 'splitter':␣
 ↪['best', 'random'], 'max_depth': [2, 3, 4]}
}
```

[ ]:
```
from sklearn.model_selection import GridSearchCV
grid_results = {}
for model_name, pipeline in models.items():
    print(f"Running GridSearchCV for {model_name}...")
    grid_search = GridSearchCV(pipeline, param_grid[model_name], cv=5,␣
 ↪scoring='accuracy')
    grid_search.fit(x_train, y_train)
    grid_results[model_name] = grid_search.best_score_

# Print best results
for model_name, score in grid_results.items():
    print(f'{model_name} Best Score: {score:.4f}')
```

```
Running GridSearchCV for Logistic Regression…
Running GridSearchCV for Bagging Classifier…
Running GridSearchCV for Random Forest Classifier…
Running GridSearchCV for Support Vector Classifier…
Running GridSearchCV for KNeighbors Classifier…
Running GridSearchCV for Decision Tree Classifier…
Logistic Regression Best Score: 0.8577
Bagging Classifier Best Score: 0.8572
Random Forest Classifier Best Score: 0.8577
Support Vector Classifier Best Score: 0.8577
KNeighbors Classifier Best Score: 0.8517
Decision Tree Classifier Best Score: 0.8577
```

### 0.0.6 Model Evaluation

```
[ ]: model1=LogisticRegression()
     model1.fit(x_train,y_train)
```

```
[ ]: LogisticRegression()
```

```
[ ]: y_pred1=model1.predict(x_test)
```

```
[ ]: accuracy_score(y_test,y_pred1)
```

```
[ ]: 0.8674698795180723
```

```
[ ]: print(classification_report(y_test,y_pred1))
```

```
              precision    recall  f1-score   support

          no       0.87      1.00      0.93       576
         yes       0.00      0.00      0.00        88

    accuracy                           0.87       664
   macro avg       0.43      0.50      0.46       664
weighted avg       0.75      0.87      0.81       664
```

### 0.0.7 Deployment

```
[ ]: import pickle
```

```
[ ]: file = 'model.pkl'
     pickle.dump(model1,open(file,'wb'))
```

```
[ ]: !pip install streamlit
```

```
Collecting streamlit
  Downloading streamlit-1.40.1-py2.py3-none-any.whl.metadata (8.5 kB)
```

```
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.10/dist-
packages (from streamlit) (4.2.2)
Requirement already satisfied: blinker<2,>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from streamlit) (1.9.0)
Requirement already satisfied: cachetools<6,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from streamlit) (5.5.0)
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.10/dist-
packages (from streamlit) (8.1.7)
Requirement already satisfied: numpy<3,>=1.20 in /usr/local/lib/python3.10/dist-
packages (from streamlit) (1.26.4)
Requirement already satisfied: packaging<25,>=20 in
/usr/local/lib/python3.10/dist-packages (from streamlit) (24.2)
Requirement already satisfied: pandas<3,>=1.4.0 in
/usr/local/lib/python3.10/dist-packages (from streamlit) (2.2.2)
Requirement already satisfied: pillow<12,>=7.1.0 in
/usr/local/lib/python3.10/dist-packages (from streamlit) (11.0.0)
Requirement already satisfied: protobuf<6,>=3.20 in
/usr/local/lib/python3.10/dist-packages (from streamlit) (4.25.5)
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.10/dist-
packages (from streamlit) (17.0.0)
Requirement already satisfied: requests<3,>=2.27 in
/usr/local/lib/python3.10/dist-packages (from streamlit) (2.32.3)
Requirement already satisfied: rich<14,>=10.14.0 in
/usr/local/lib/python3.10/dist-packages (from streamlit) (13.9.4)
Requirement already satisfied: tenacity<10,>=8.1.0 in
/usr/local/lib/python3.10/dist-packages (from streamlit) (9.0.0)
Requirement already satisfied: toml<2,>=0.10.1 in
/usr/local/lib/python3.10/dist-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.3.0 in
/usr/local/lib/python3.10/dist-packages (from streamlit) (4.12.2)
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in
/usr/local/lib/python3.10/dist-packages (from streamlit) (3.1.43)
Collecting pydeck<1,>=0.8.0b4 (from streamlit)
  Downloading pydeck-0.9.1-py2.py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: tornado<7,>=6.0.3 in
/usr/local/lib/python3.10/dist-packages (from streamlit) (6.3.3)
Collecting watchdog<7,>=2.1.5 (from streamlit)
  Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl.metadata (44 kB)
                           44.3/44.3 kB
2.2 MB/s eta 0:00:00
Requirement already satisfied: entrypoints in
/usr/local/lib/python3.10/dist-packages (from altair<6,>=4.0->streamlit) (0.4)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from altair<6,>=4.0->streamlit) (3.1.4)
Requirement already satisfied: jsonschema>=3.0 in
/usr/local/lib/python3.10/dist-packages (from altair<6,>=4.0->streamlit)
(4.23.0)
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages
```

```
(from altair<6,>=4.0->streamlit) (0.12.1)
Requirement already satisfied: gitdb<5,>=4.0.1 in
/usr/local/lib/python3.10/dist-packages (from
gitpython!=3.1.19,<4,>=3.0.7->streamlit) (4.0.11)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas<3,>=1.4.0->streamlit)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas<3,>=1.4.0->streamlit) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas<3,>=1.4.0->streamlit) (2024.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.27->streamlit)
(3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests<3,>=2.27->streamlit) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.27->streamlit)
(2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.27->streamlit)
(2024.8.30)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich<14,>=10.14.0->streamlit)
(3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich<14,>=10.14.0->streamlit)
(2.18.0)
Requirement already satisfied: smmap<6,>=3.0.1 in
/usr/local/lib/python3.10/dist-packages (from
gitdb<5,>=4.0.1->gitpython!=3.1.19,<4,>=3.0.7->streamlit) (5.0.1)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->altair<6,>=4.0->streamlit)
(3.0.2)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (24.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/usr/local/lib/python3.10/dist-packages (from
jsonschema>=3.0->altair<6,>=4.0->streamlit) (2024.10.1)
Requirement already satisfied: referencing>=0.28.4 in
/usr/local/lib/python3.10/dist-packages (from
jsonschema>=3.0->altair<6,>=4.0->streamlit) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (0.21.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich<14,>=10.14.0->streamlit) (0.1.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas<3,>=1.4.0->streamlit) (1.16.0)
```

```
Downloading streamlit-1.40.1-py2.py3-none-any.whl (8.6 MB)
                         8.6/8.6 MB
47.5 MB/s eta 0:00:00
Downloading pydeck-0.9.1-py2.py3-none-any.whl (6.9 MB)
                         6.9/6.9 MB
62.0 MB/s eta 0:00:00
Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl (79 kB)
                         79.1/79.1 kB
5.4 MB/s eta 0:00:00
Installing collected packages: watchdog, pydeck, streamlit
Successfully installed pydeck-0.9.1 streamlit-1.40.1 watchdog-6.0.0
```

[ ]:

```
%%writefile Model_Dep_Telecommunication.py
import streamlit as st
import pickle
import pandas as pd
model1=pickle.load(open('model.pkl','rb'))
st.title("Model Deployment for TeleCommunication Dataset")
def user_input_parameters():
  account_length=st.number_input("Account Length")
  voice_messages=st.number_input("voice messages")
  day_minutes=st.number_input("Total day minutes")
  day_calls=st.number_input("Total day calls")
  day_charge=st.number_input("Total day charge")
  eve_minutes=st.number_input("Total eve minutes")
  eve_calls=st.number_input("Total eve calls")
  eve_charge=st.number_input("Total eve charge")
  night_minutes=st.number_input("Total night minutes")
  night_calls=st.number_input("Total night calls")
  night_charge=st.number_input("Total night charge")
  intl_minutes=st.number_input("Total intl minutes")
  intl_calls=st.number_input("Total intl calls")
  intl_charge=st.number_input("Total intl charge")
  customer_calls=st.number_input("Customer service calls")

  ord__area_code=st.selectbox("Area Codes 0-area_code_408 1-area_code_415␣
↪2-area_code_512",[0,1,2])
  ord__voice_plan=st.selectbox("Voice plan 1-yes 0-No",[0,1])
  ord__intl_plan=st.selectbox("Order int plan No-0,Yes-1",[0,1])
  data={"account.length":account_length,'voice.messages':voice_messages,'intl.
↪mins':intl_minutes,'intl.calls':intl_calls,'intl.charge':intl_charge,'day.
↪mins':day_minutes,'day.calls':day_calls,'day.charge':day_charge,
        'eve.mins':eve_minutes,'eve.calls':eve_calls,'eve.charge':
↪eve_charge,'night.mins':night_minutes,'night.calls':night_calls,'night.
↪charge':night_charge,
        'customer.calls':customer_calls,
        'ord__area.code':ord__area_code,'ord__voice.plan':
  ↪ord__voice_plan,'ord__intl.plan':ord__intl_plan}
    features=pd.DataFrame(data,index=[0])
    return features
df=user_input_parameters()
st.subheader('User Inputs')
st.write(df)
predict=model1.predict(df)
predict_proba=model1.predict_proba(df)
st.subheader(df)
st.write('yes The Customer is Discontinued' if predict_proba[0][1]>0.5 else 'No␣
  ↪The Customer is Active')
st.subheader('predict_prob')
st.write(predict_proba)
```

```
Writing Model_Dep_Telecommunication.py
```

[ ]:
```
!npm install -g localtunnel -U
```

```

added 22 packages, and audited 23 packages in 3s

3 packages are looking for funding
  run `npm fund` for details

1 moderate severity vulnerability

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

[ ]:
```
!wget -q -O - ipv4.icanhazip.com
```

```
35.224.81.225
```

[ ]:
```
!streamlit run Model_Dep_Telecommunication.py & npx localtunnel --port 8501
```

```

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to
false.

your url is: https://cool-wolves-dance.loca.lt

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://172.28.0.12:8501
  External URL: http://35.224.81.225:8501
```

[ ]:

[ ]:

[ ]: