Importing required packages

In [87]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [88]:
```
churn_data=pd.read_csv(r"C:\Users\91880\Documents\nareshit\datafiles\telecom_churn_
```

Dataset Overview

- this involves understanding shape,size,types of input features

In [90]:
```
churn_data.head()
```

Out[90]:

	year	customer_id	phone_no	gender	age	no_of_days_subscribed	multi_screen	mail_s
0	2015	100198	409-8743	Female	36	62	no	
1	2015	100643	340-5930	Female	39	149	no	
2	2015	100756	372-3750	Female	65	126	no	
3	2015	101595	331-4902	Female	24	131	no	
4	2015	101653	351-8398	Female	40	191	no	

In [91]:
```
churn_data=churn_data.drop(['year','customer_id','phone_no'],axis=1)
```

In [92]:
```
churn_data.head()
```

Out[92]:

	gender	age	no_of_days_subscribed	multi_screen	mail_subscribed	weekly_mins_watched
0	Female	36	62	no	no	148.3
1	Female	39	149	no	no	294.4
2	Female	65	126	no	no	87.3
3	Female	24	131	no	yes	321.3
4	Female	40	191	no	no	243.0

In [93]:
```
churn_data.shape
```

Out[93]:
```
(2000, 13)
```

In [94]:
```
churn_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 13 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   gender                  1976 non-null   object 
 1   age                     2000 non-null   int64  
 2   no_of_days_subscribed   2000 non-null   int64  
 3   multi_screen            2000 non-null   object 
 4   mail_subscribed         2000 non-null   object 
 5   weekly_mins_watched     2000 non-null   float64
 6   minimum_daily_mins      2000 non-null   float64
 7   maximum_daily_mins      2000 non-null   float64
 8   weekly_max_night_mins   2000 non-null   int64  
 9   videos_watched          2000 non-null   int64  
 10  maximum_days_inactive   1972 non-null   float64
 11  customer_support_calls  2000 non-null   int64  
 12  churn                   1965 non-null   float64
dtypes: float64(5), int64(5), object(3)
memory usage: 203.3+ KB
```

Dividing data into categorical and numerical

In [96]:
```
cat=churn_data.select_dtypes(include='object').columns
num=churn_data.select_dtypes(exclude='object').columns
```

Missing Value Analysis

- identifying the missing values in input features
- filling the missing the values **mode** for categorical and **mean** for numerical data

In [98]:
```
#misiing value analysis
for i in churn_data:
    print(i,churn_data[i].nunique())
```

```
gender 2
age 63
no_of_days_subscribed 204
multi_screen 2
mail_subscribed 2
weekly_mins_watched 1260
minimum_daily_mins 149
maximum_daily_mins 1260
weekly_max_night_mins 111
videos_watched 19
maximum_days_inactive 7
customer_support_calls 10
churn 2
```

In [99]:
```
churn_data.isnull().sum()
```

Out[99]:
```
gender                    24
age                        0
no_of_days_subscribed      0
multi_screen               0
mail_subscribed            0
weekly_mins_watched        0
minimum_daily_mins         0
maximum_daily_mins         0
weekly_max_night_mins      0
videos_watched             0
maximum_days_inactive     28
customer_support_calls     0
churn                     35
dtype: int64
```

In [100…
```
mode1=churn_data['gender'].mode()
mean1=round(churn_data['maximum_days_inactive'].mean())
mode2=churn_data['churn'].mode()
mode1,mean1,mode2
```

Out[100…
```
(0    Male
 Name: gender, dtype: object,
 3,
 0    0.0
 Name: churn, dtype: float64)
```

In [101…
```
churn_data['gender']=churn_data['gender'].fillna('Male')
```

In [102…
```
churn_data['gender'].unique()
```

Out[102…
```
array(['Female', 'Male'], dtype=object)
```

In [103…
```
churn_data['maximum_days_inactive']=churn_data['maximum_days_inactive'].fillna(3)
churn_data['maximum_days_inactive'].isnull().sum()
```

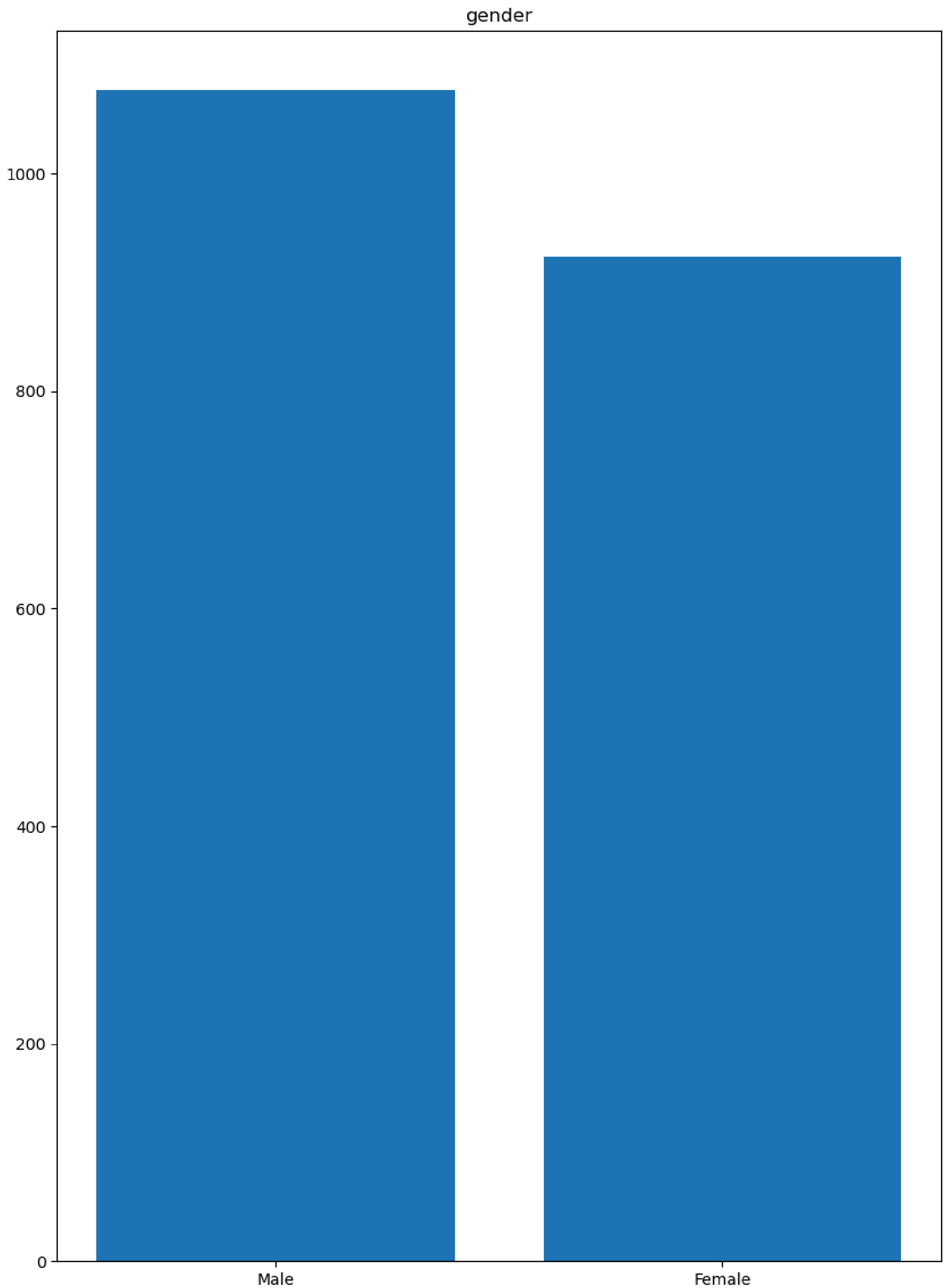
Out[103…
```
0
```

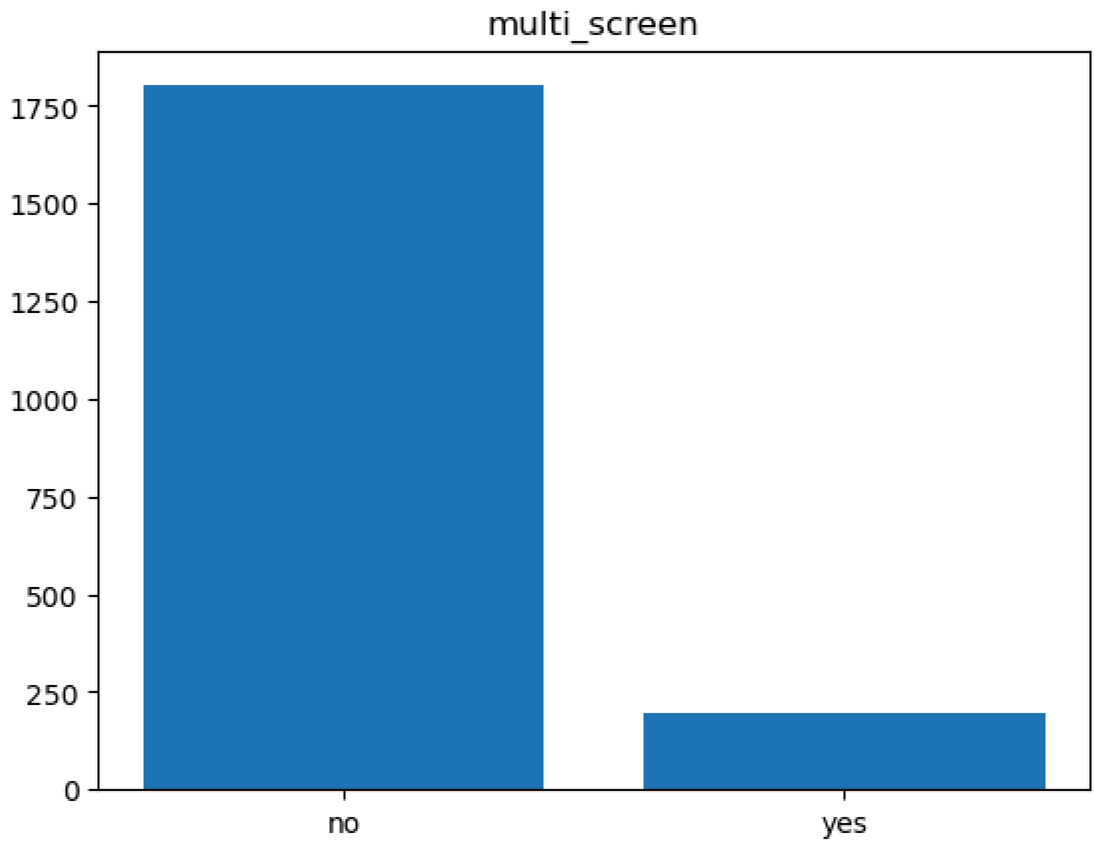
In [104…
```
churn_data['churn']=churn_data['churn'].fillna(0.0)
churn_data['churn'].isnull().sum()
```

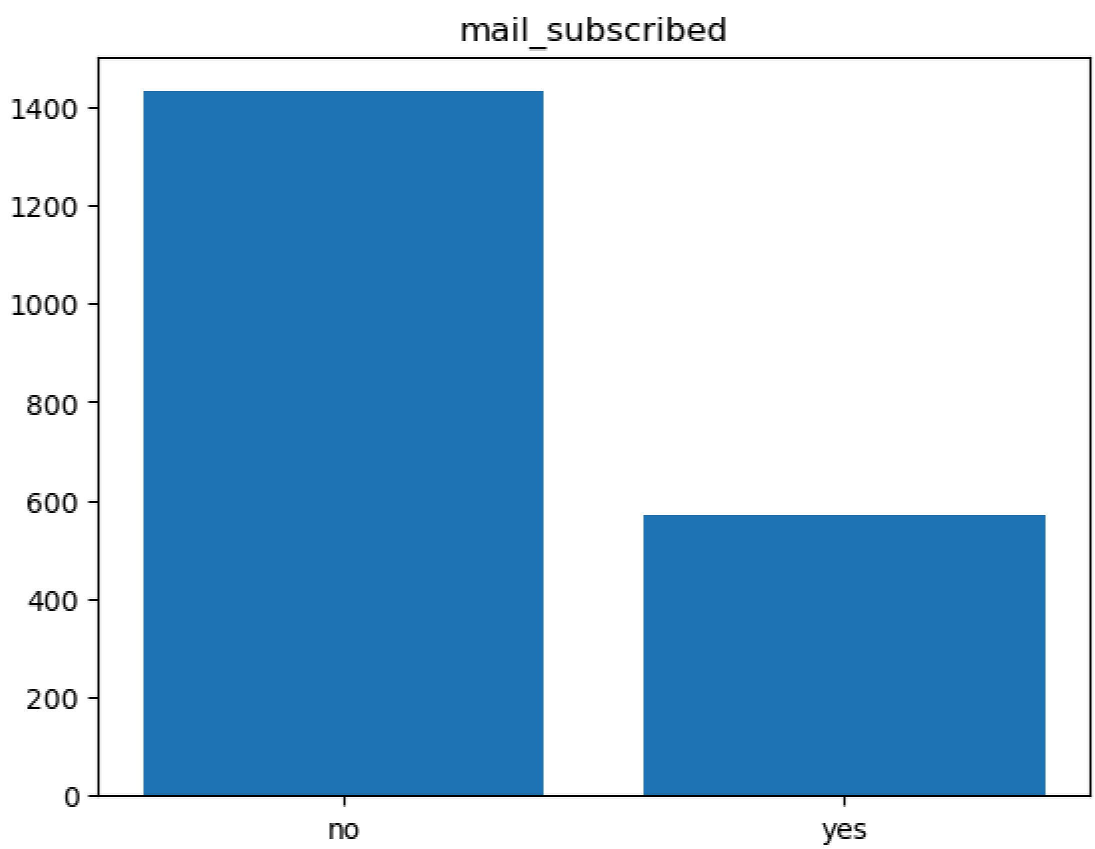
Out[104…
```
0
```

Univariate Analysis

In [106…
```
#univariate analysis
plt.figure(figsize=(10,14))
for i in cat:
    keys=churn_data[i].value_counts().keys()
    values=churn_data[i].value_counts().values
    plt.bar(keys,values)
    plt.title(i)
    plt.show()
```


gender
1000
800
600
400
200
0
Male
Female





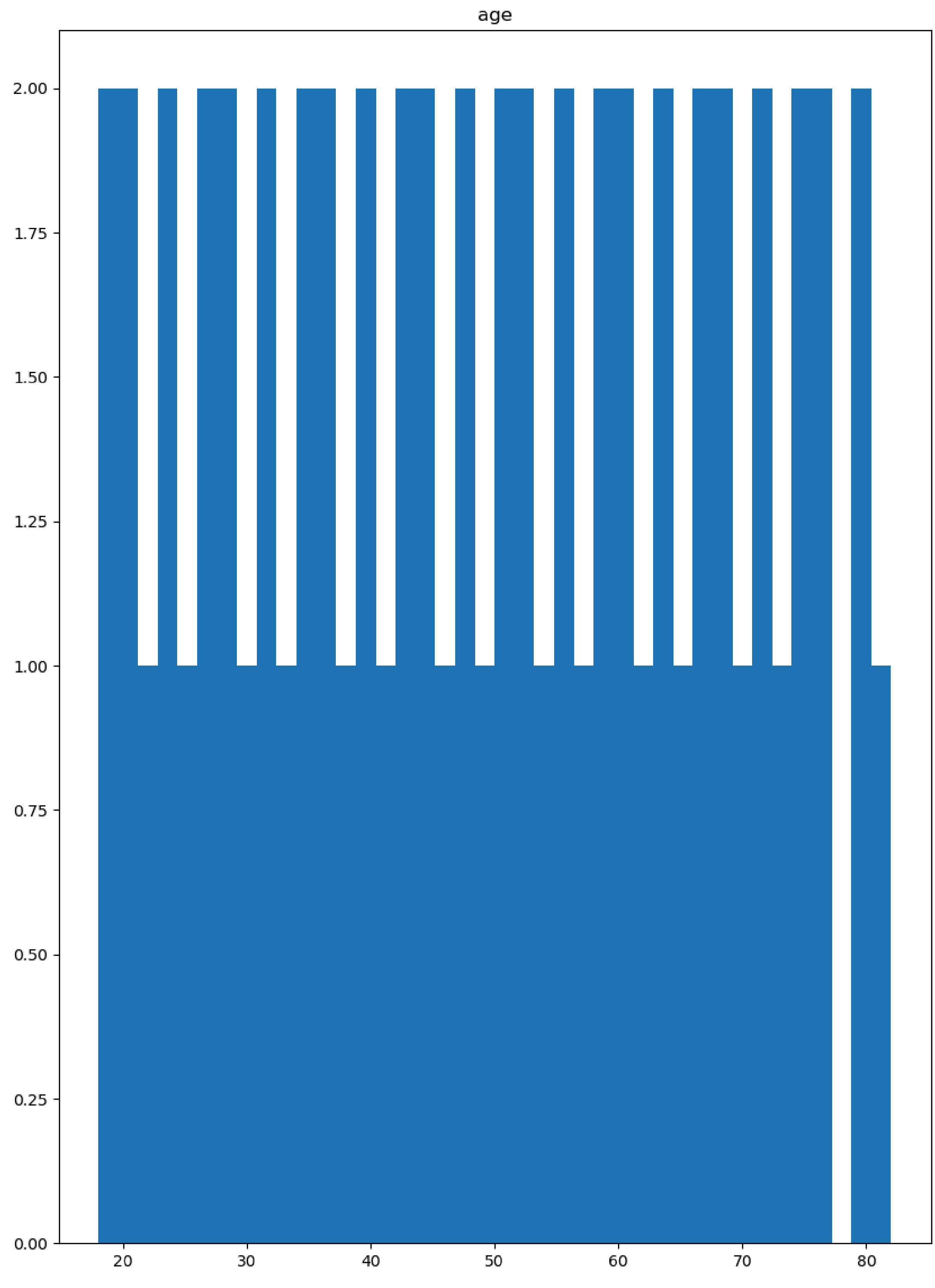
understandins

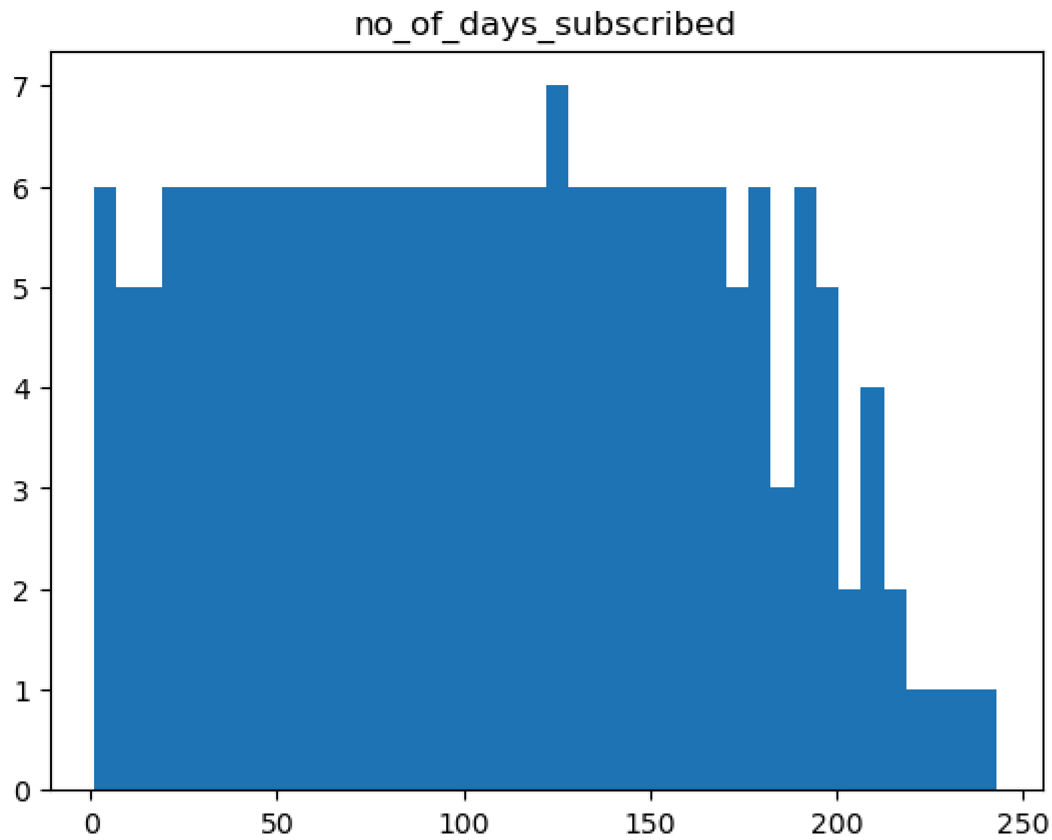
- based on above bar charts we can understand that most of users are *Male*

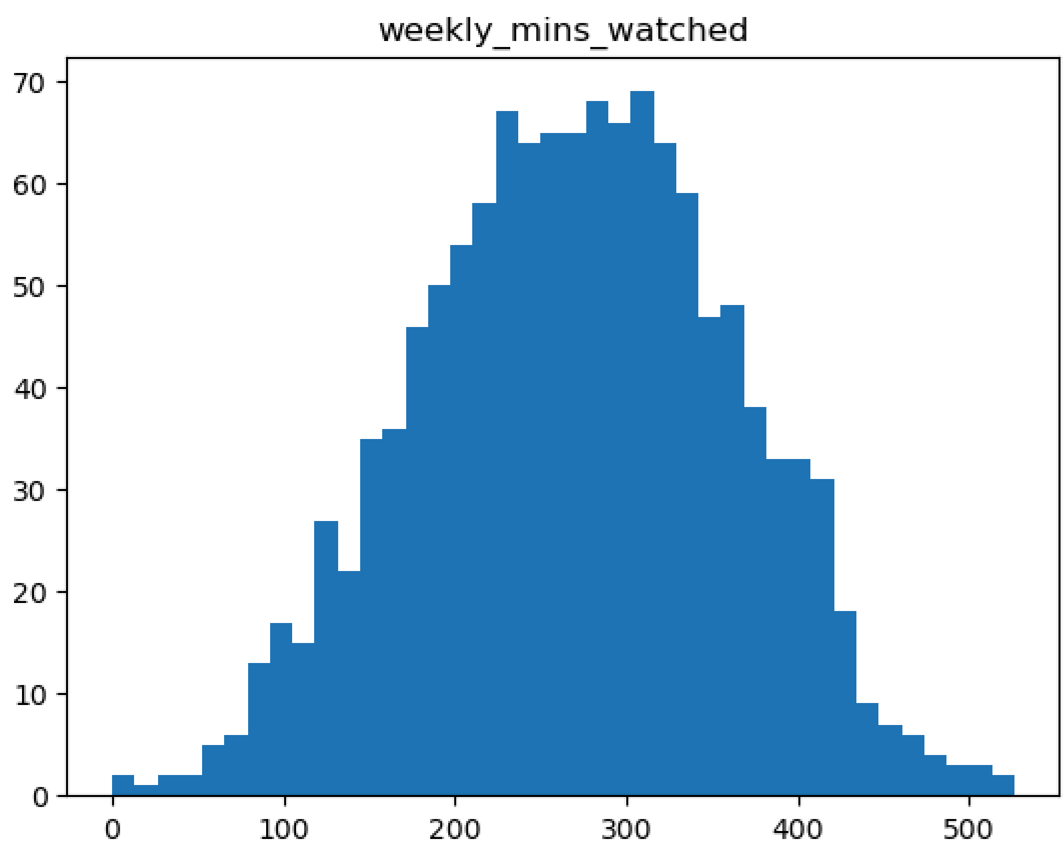
- mails services are not subscribed so customer won't get information like :new offers,plans etc,,,

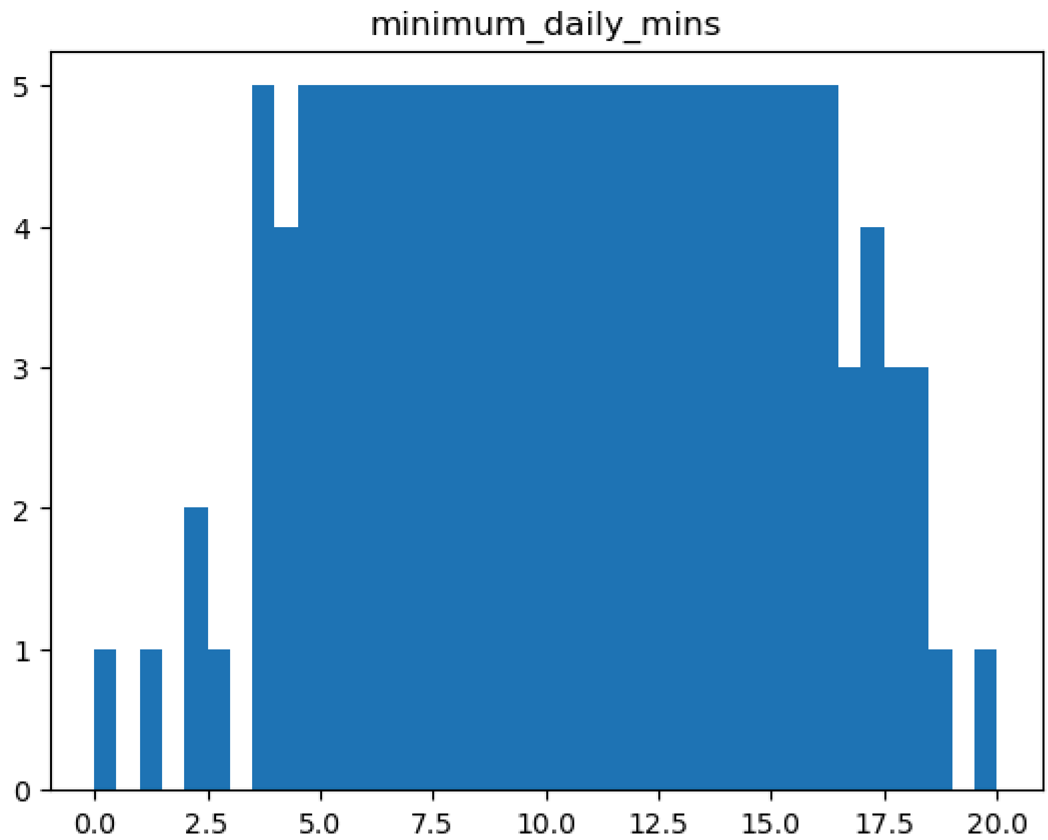
In [108…

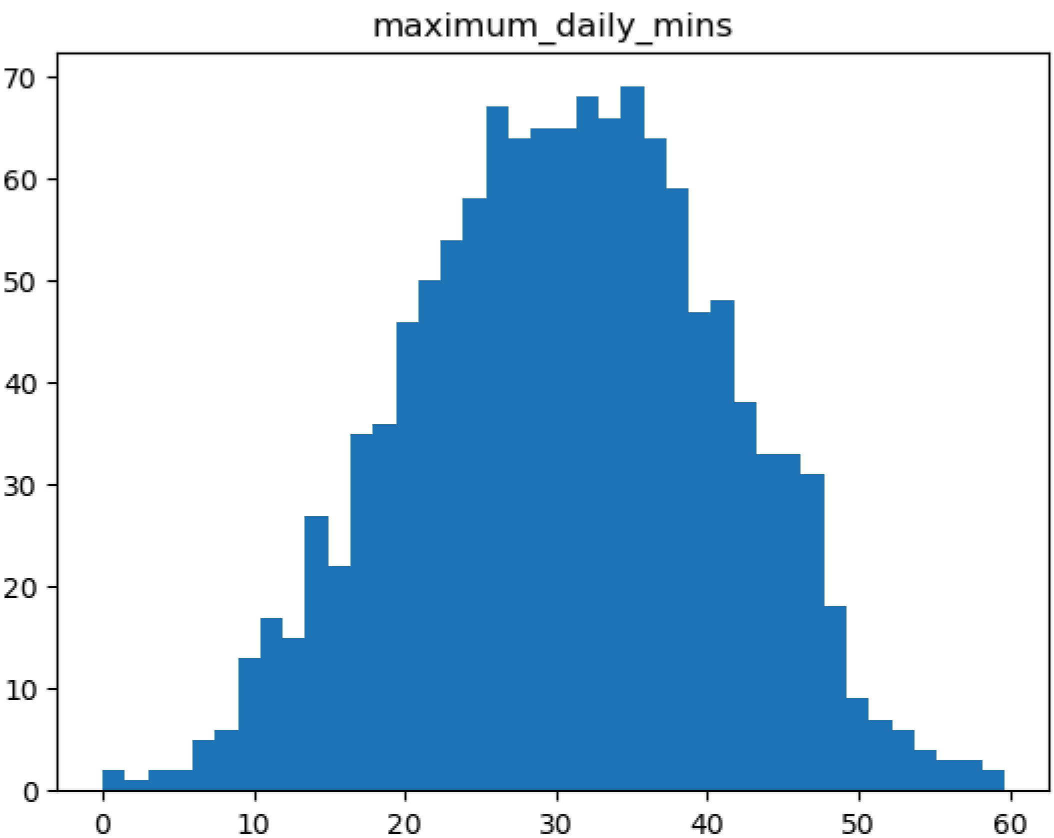
```
plt.figure(figsize=(10,14))
for i in num:
    keys=churn_data[i].value_counts().keys()
    values=churn_data[i].value_counts().values
    plt.hist(keys,bins=40)
    plt.title(i)
    plt.show()
```

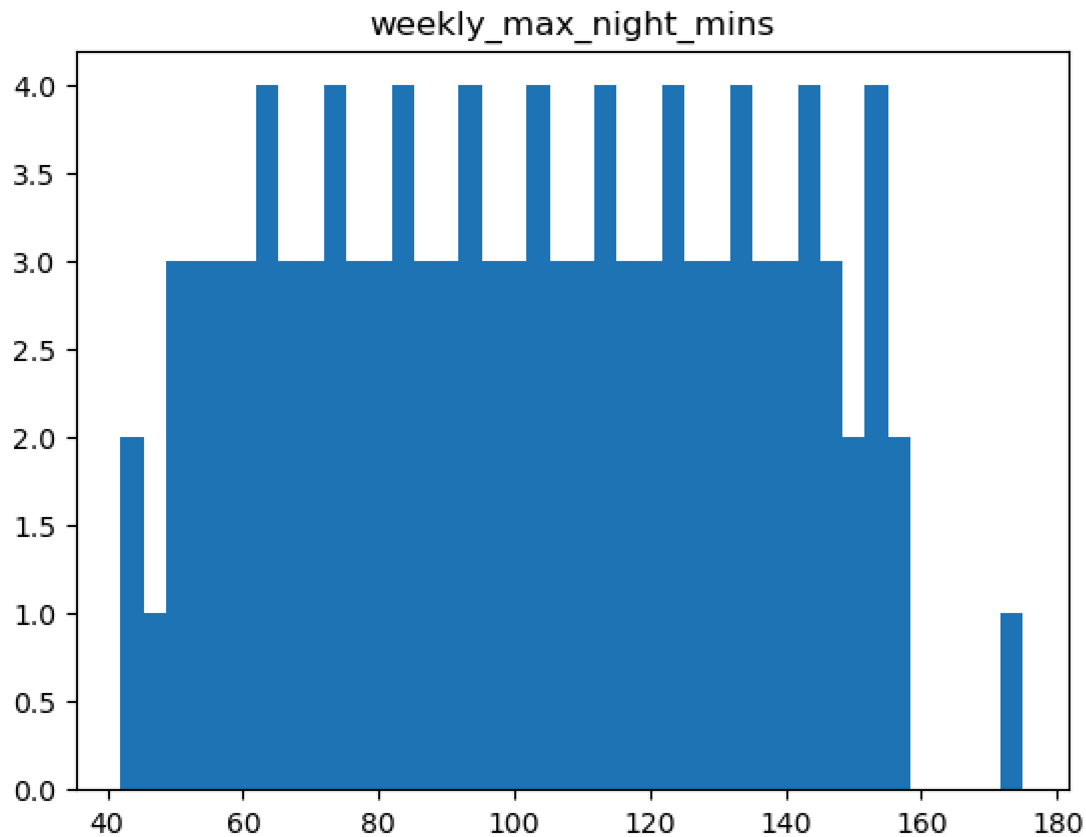

age
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
20
30
40
50
60
70
80

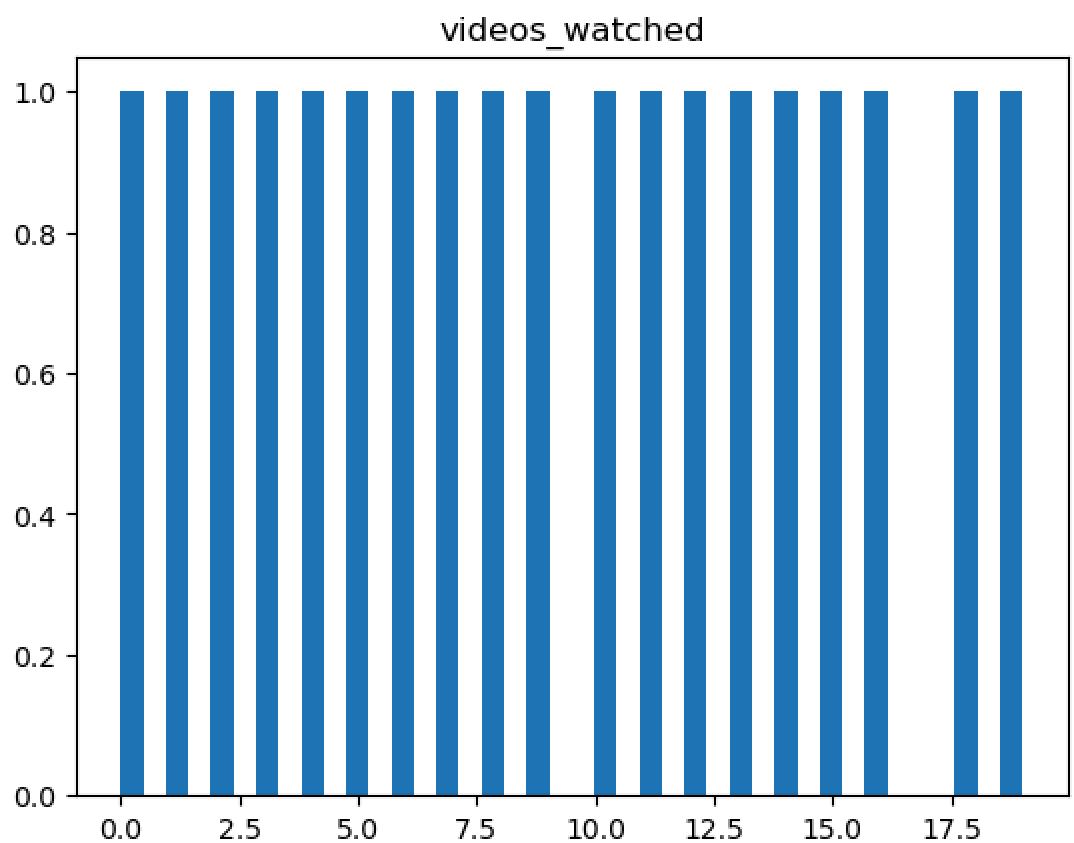

no_of_days_subscribed
7
6
5
4
3
2
1
0
0
50
100
150
200
250

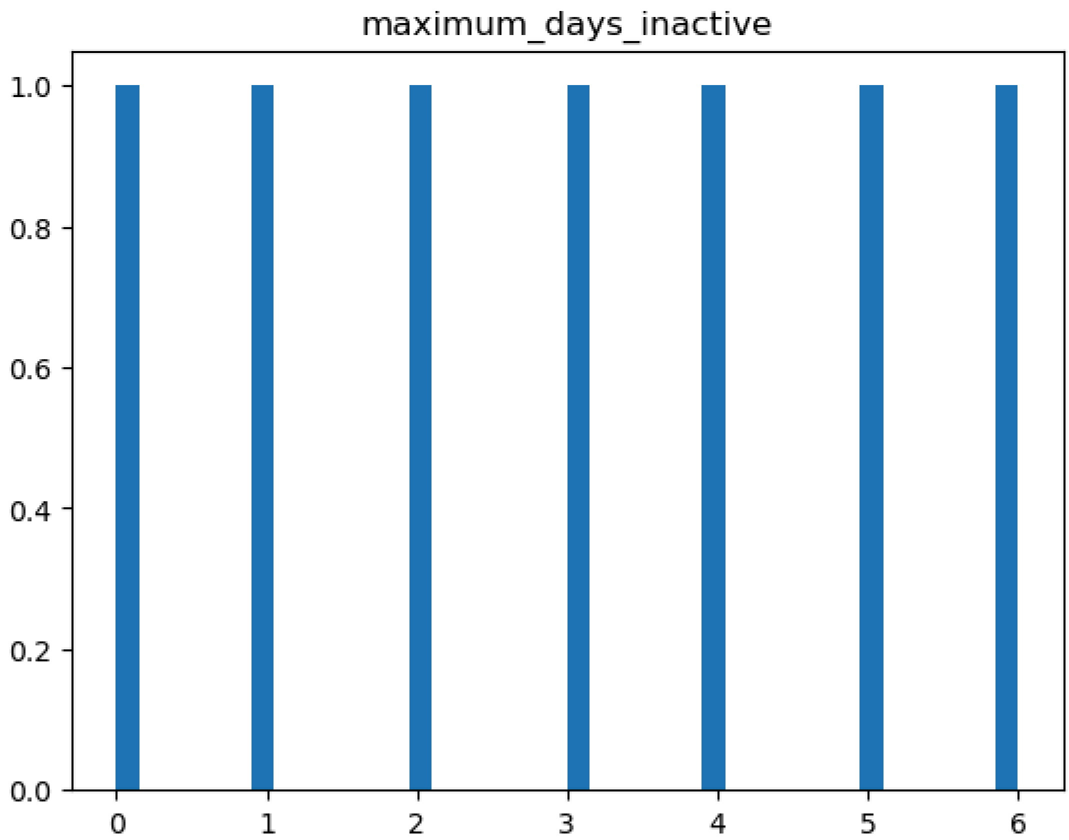

weekly_mins_watched
70
60
50
40
30
20
10
0
0
100
200
300
400
500

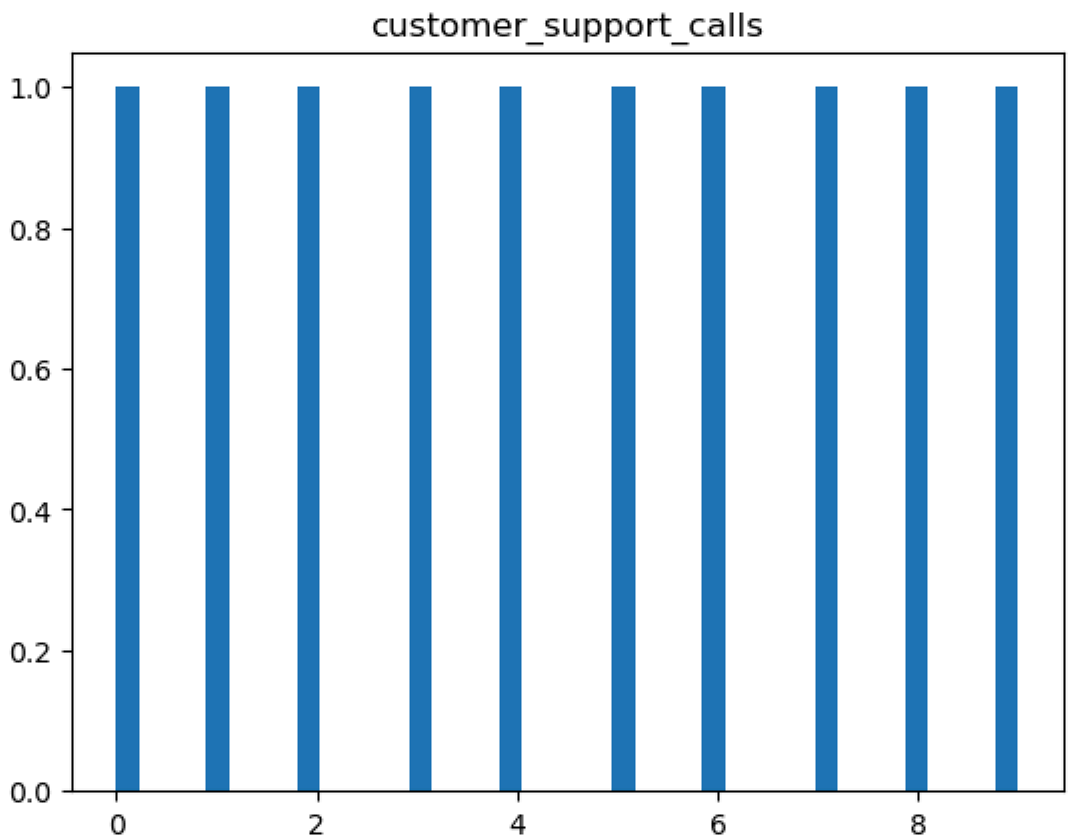

minimum_daily_mins
0
1
2
3
4
5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

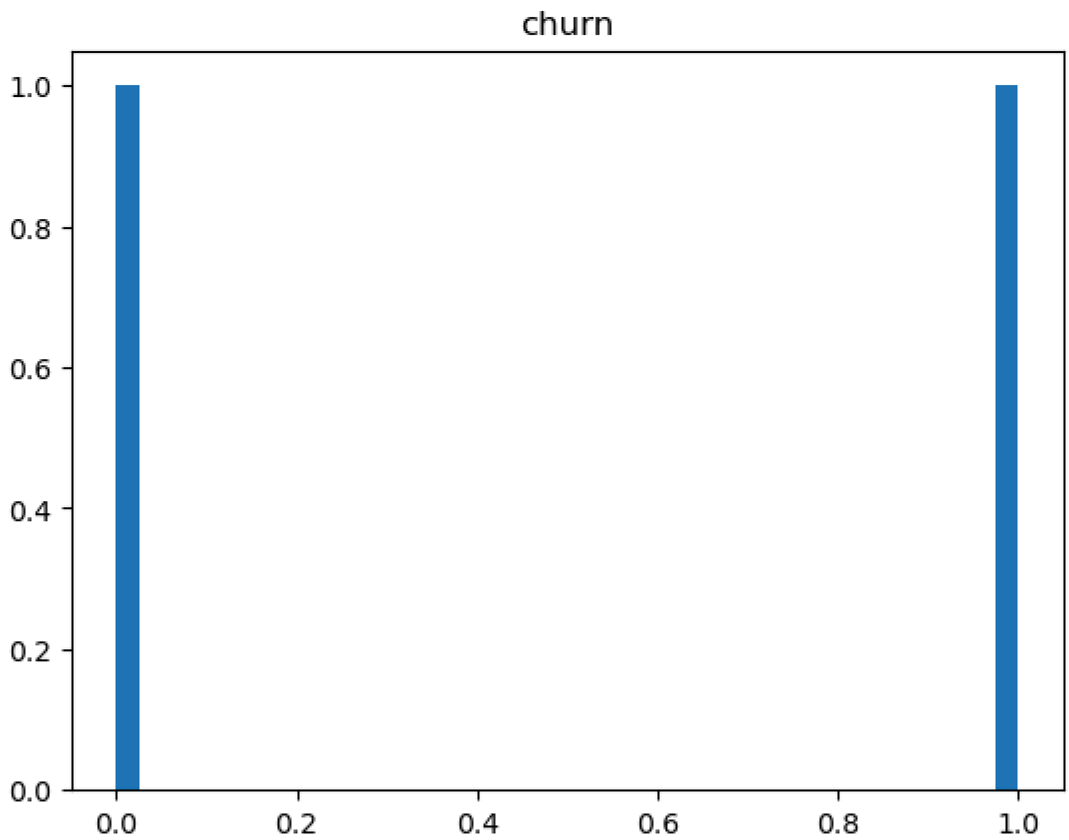

maximum_daily_mins
0
10
20
30
40
50
60
70
0
10
20
30
40
50
60


weekly_max_night_mins
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
40
60
80
100
120
140
160
180


videos_watched
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


maximum_days_inactive
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6


customer_support_calls
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8



Understangings

- there is skewness ,data is not normally distributed

bi-variate analysis

In [111…
```
labels=[i for i in cat]
labels
```

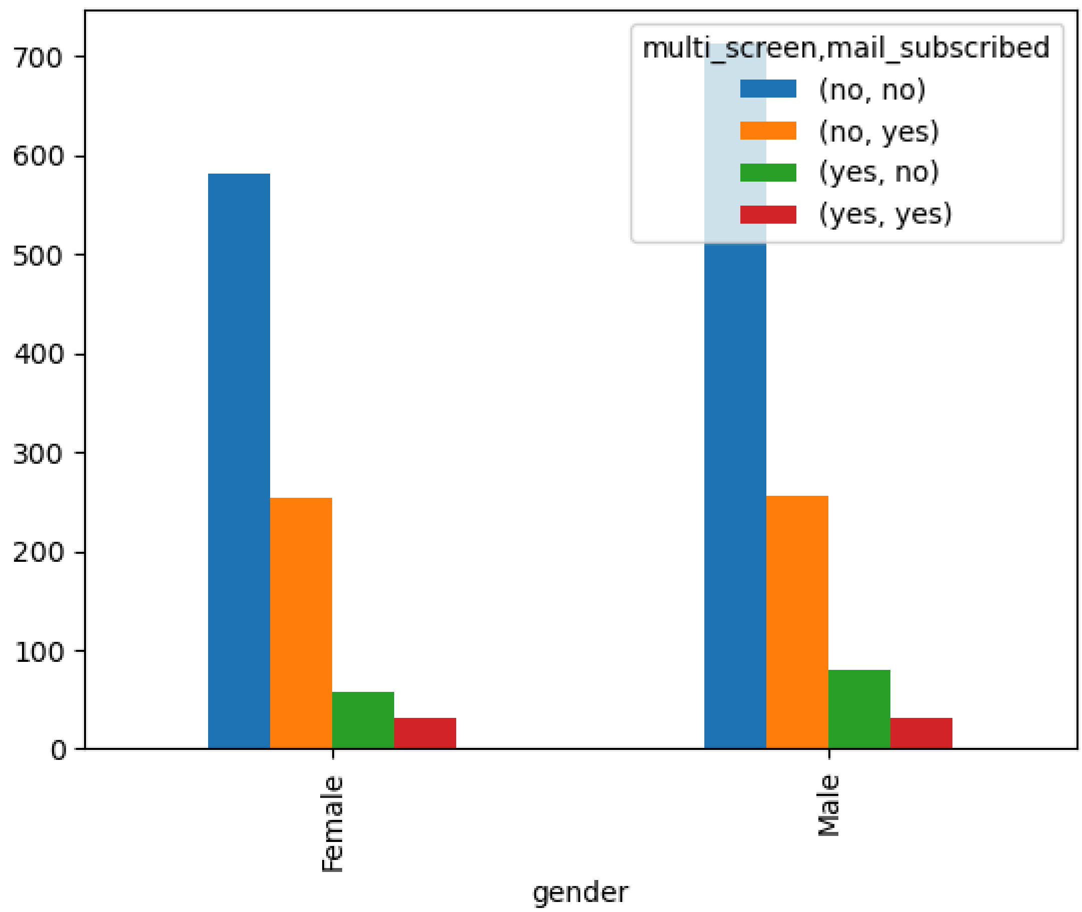
Out[111…
```
['gender', 'multi_screen', 'mail_subscribed']
```

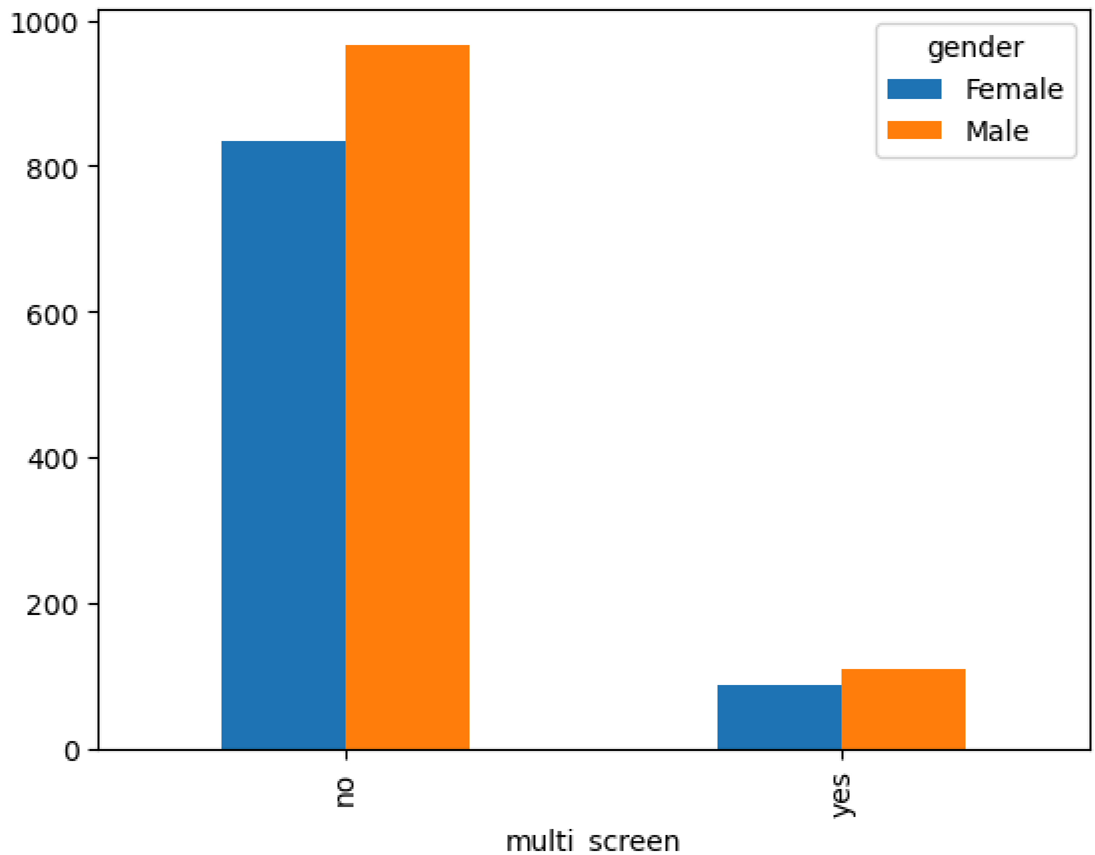
In [112…
```
#cat vs cat
col1=churn_data['gender']
col2=churn_data['multi_screen']
col3=churn_data['mail_subscribed']
idx=col1
cols=[col2,col3]
p1=pd.crosstab(idx,cols)
```

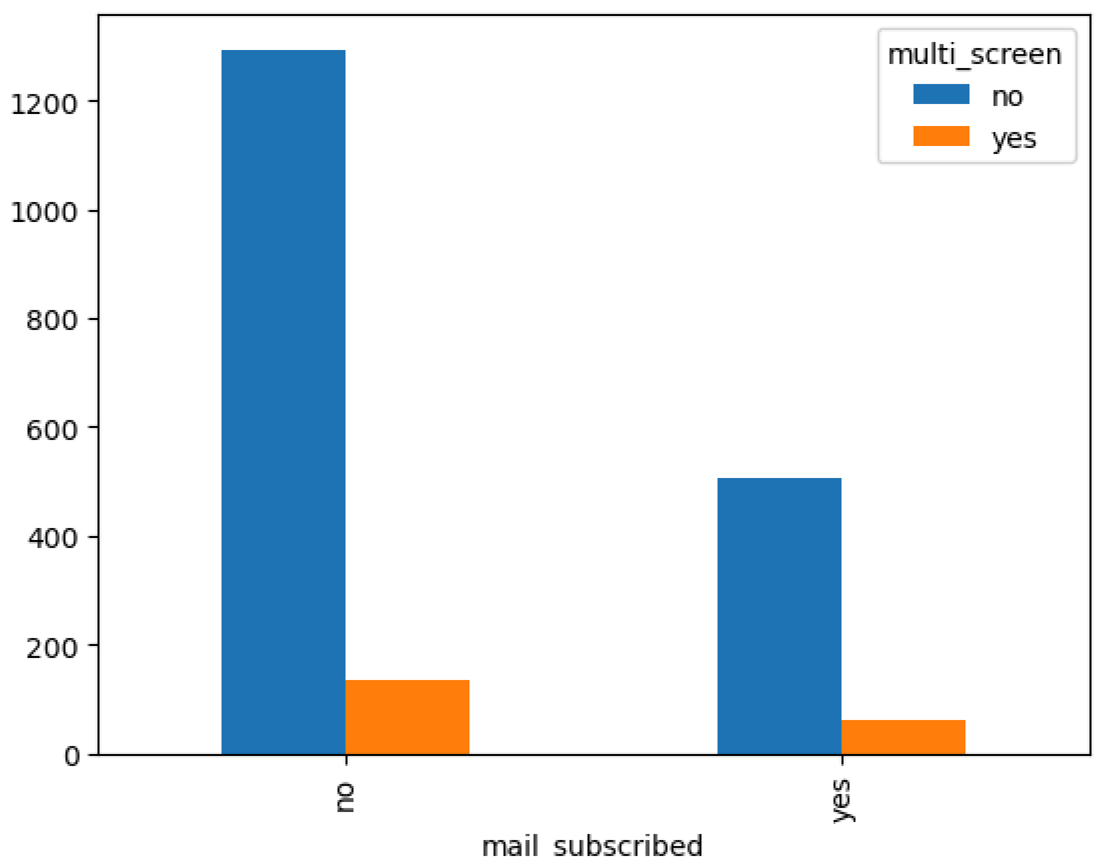
In [113…
```
idx1=col2
cols1=[col1,col3]
idx2=col3
cols2=[col1,col2]
p2=pd.crosstab(idx1,col1)
p3=pd.crosstab(idx2,col2)
```

In [114…
```
p1.plot(kind='bar')
p2.plot(kind='bar')
```

```
p3.plot(kind='bar')
plt.show()
```







In [115…

```
p={}
target=churn_data['churn']
```

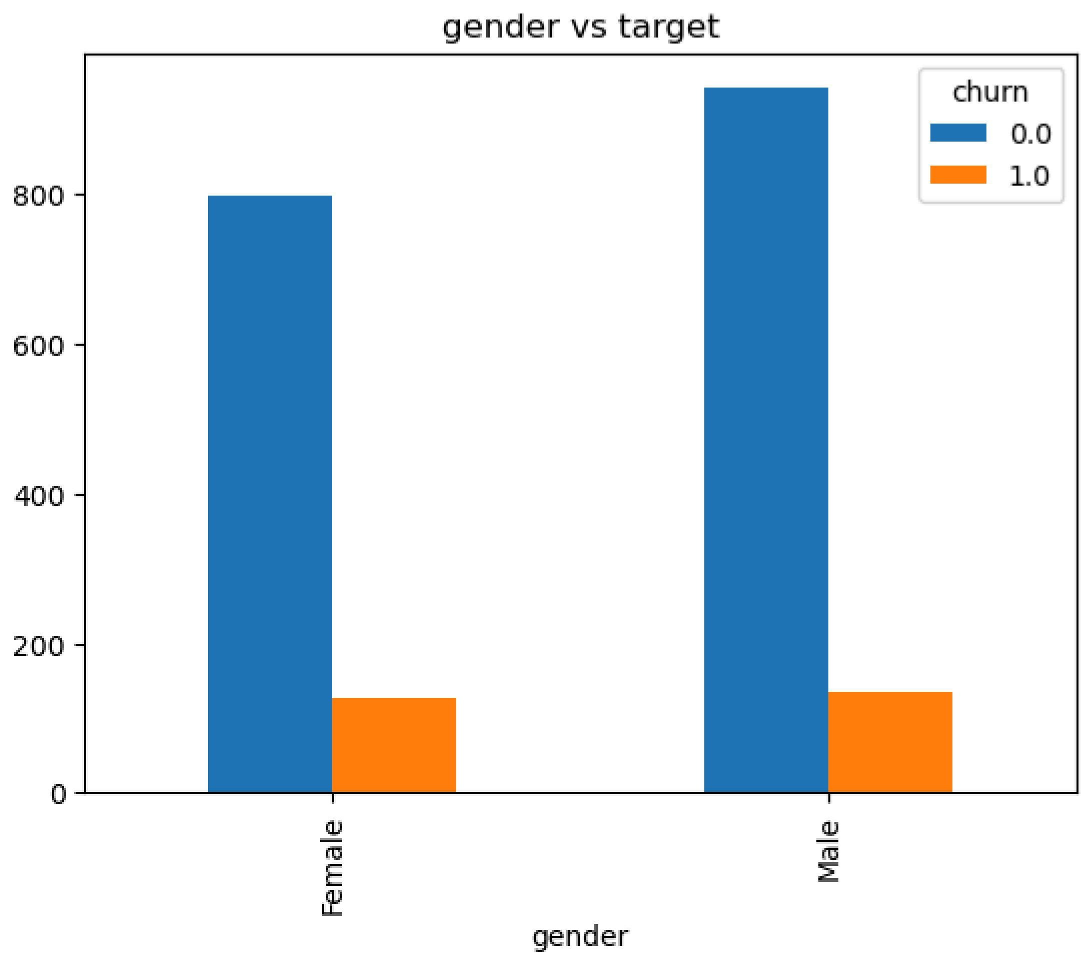
```
for i in cat:
   p[i]=pd.crosstab(churn_data[i],target)
```

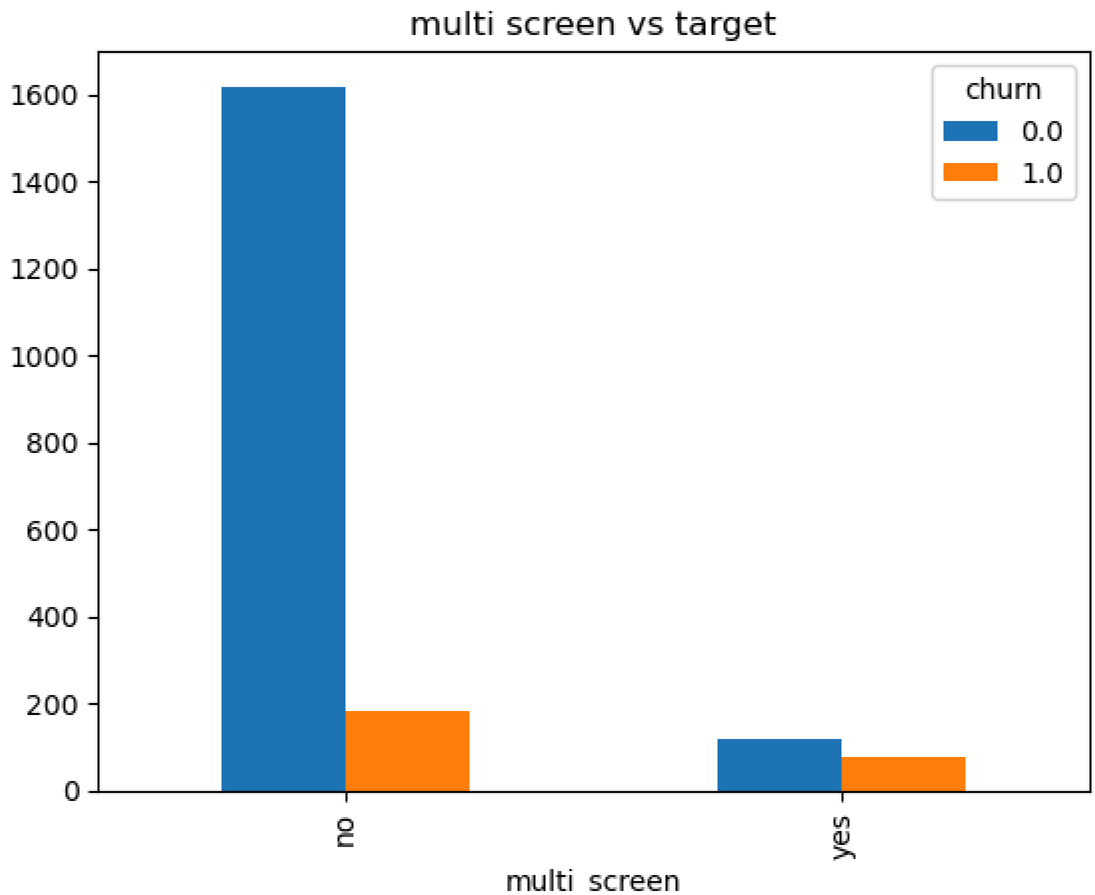
In [116…

```
p['gender'].plot(kind='bar')
plt.title('gender vs target')
p['multi_screen'].plot(kind='bar')
plt.title('multi screen vs target')
p['mail_subscribed'].plot(kind='bar')
plt.title('mail subscribe vs target')
```

Out[116…

```
Text(0.5, 1.0, 'mail subscribe vs target')
```




multi screen vs target
churn
0.0
1.0
0
200
400
600
800
1000
1200
1400
1600
no
yes
multi_screen

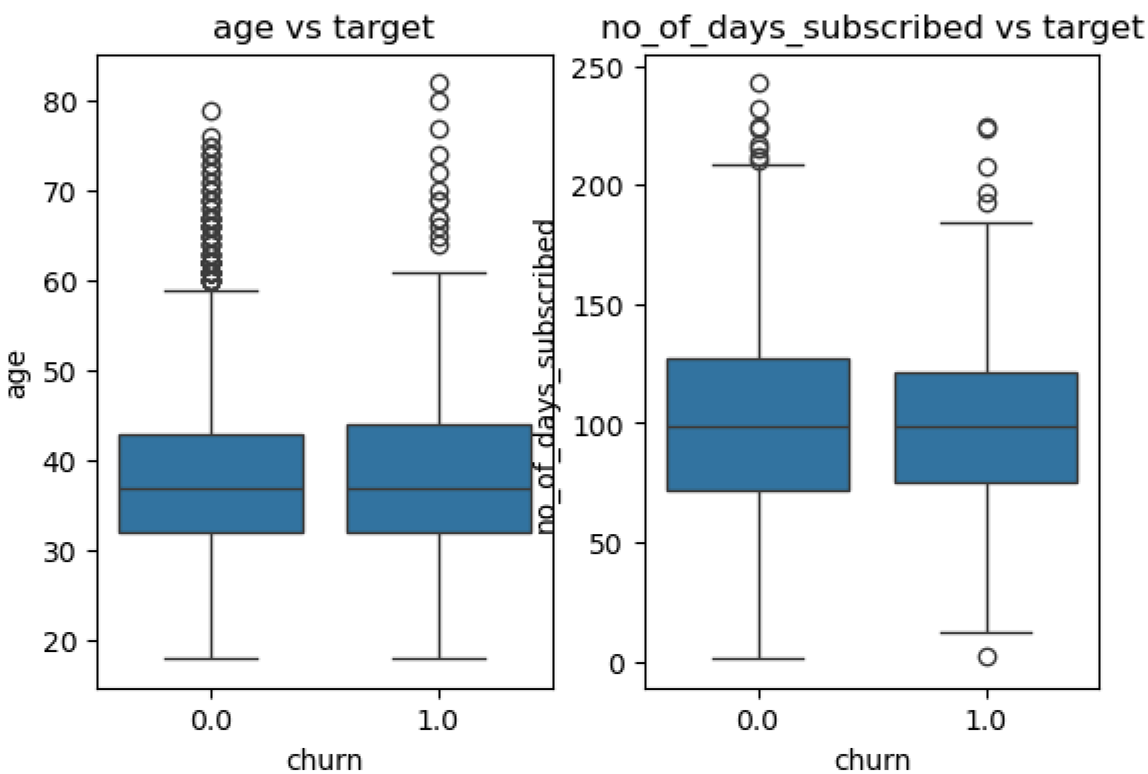

mail subscribe vs target
churn
0.0
1.0
0
200
400
600
800
1000
1200
no
yes
mail_subscribed

In [117…
```
num
```

Out[117…
```
Index(['age', 'no_of_days_subscribed', 'weekly_mins_watched',
       'minimum_daily_mins', 'maximum_daily_mins', 'weekly_max_night_mins',
       'videos_watched', 'maximum_days_inactive', 'customer_support_calls',
       'churn'],
      dtype='object')
```

In [118…
```
label=[i for i in num]
target=churn_data['churn']
```

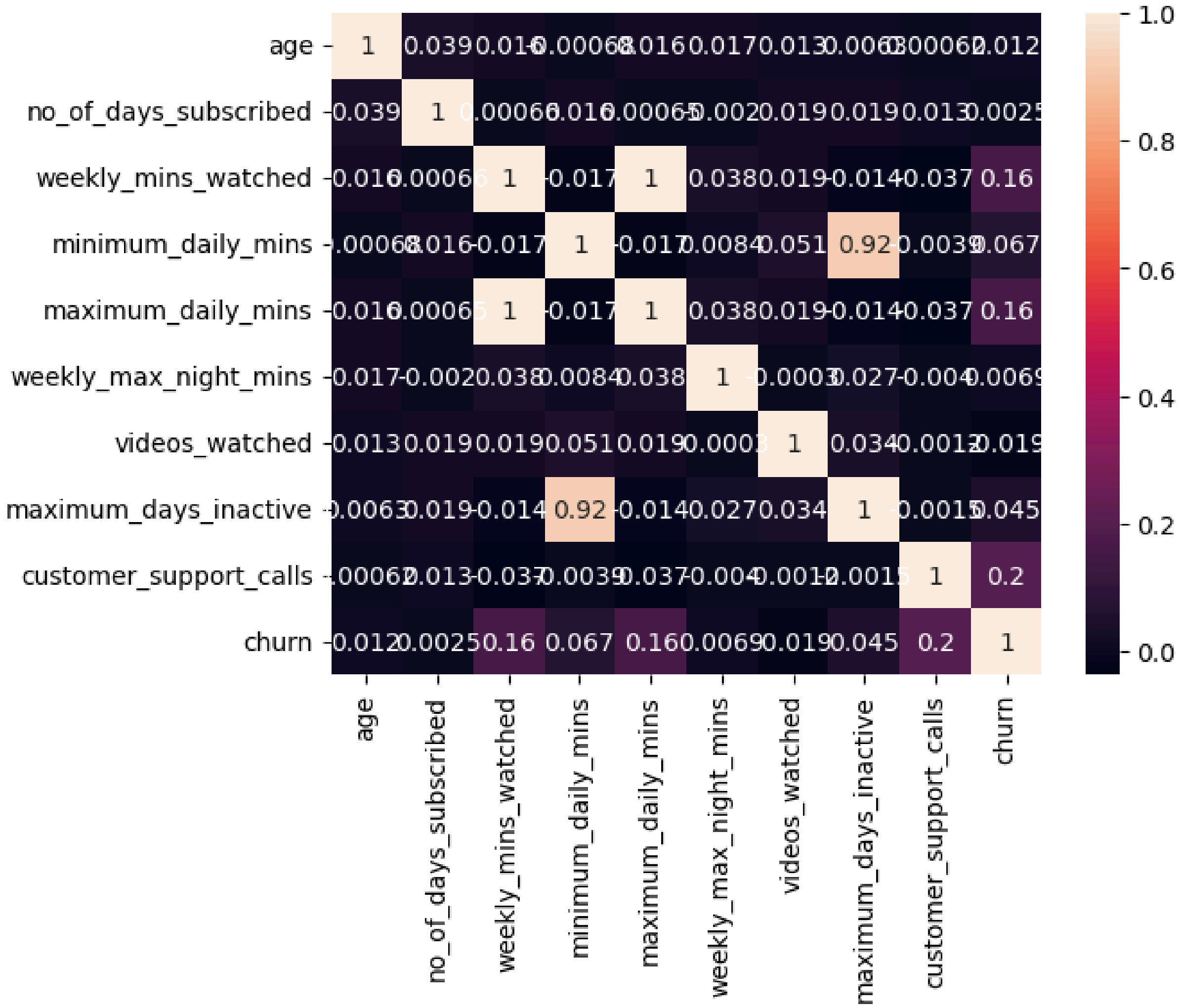
In [119…
```
plt.figure(figsize=(10,14))
for i in range(len(labels[1:10])):
    plt.subplot(3,3,i+1)
    sns.boxplot(y=label[i],x=target,data=churn_data,vert=True)
    plt.title(f'{label[i]} vs target')
```



In [120…
```
corr=churn_data.corr(numeric_only=True)
```

In [121…
```
sns.heatmap(corr,annot=True)
```

Out[121…
```
<Axes: >
```



- BASED ON THE HEATMAP WE CAN UNDERSTAND
  - MAX_DAYS INACTIVE,MIN_DAILY MINUTES AND MAXDAILY MINS AND WEEKLY MINS WATCHED ARE HIGHLY CORRELATED
  - CUSTOMER SUPPORT CALLS,MAX DAILY MINS,WEEKLY MINS ARE CORRELATED WITH TARGET VARIABLE

outlier analysis

In [124…

```
for i in num:
    Q1=np.percentile(churn_data[i],25)
    Q2=np.percentile(churn_data[i],50)
    Q3=np.percentile(churn_data[i],75)
    IQR=Q3-Q1
    lb=Q1-1.5*IQR
    ub=Q3+1.5*IQR
    con=(churn_data[i]<lb)|(churn_data[i]>ub)
    outliers=churn_data[con]
    print(i,len(outliers))
```

```
age 63
no_of_days_subscribed 11
weekly_mins_watched 18
minimum_daily_mins 25
maximum_daily_mins 18
weekly_max_night_mins 9
videos_watched 51
maximum_days_inactive 24
customer_support_calls 161
churn 262
```

understanding

- the outliers are less than 3% of original data so keeping them as it is

Encoding

- converting categorical values into numerical using label encoder

In [127…

```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [128…

```python
for i in cat:
    churn_data[i]=le.fit_transform(churn_data[i])
churn_data
```

Out[128…

	gender	age	no_of_days_subscribed	multi_screen	mail_subscribed	weekly_mins_wat
0	0	36	62	0	0	14
1	0	39	149	0	0	29
2	0	65	126	0	0	8
3	0	24	131	0	1	32
4	0	40	191	0	0	24
...	...	...	...	...	...	
1995	0	54	75	0	1	18
1996	1	45	127	0	0	27
1997	1	53	94	0	0	12
1998	1	40	94	0	0	17
1999	1	37	73	0	0	32

2000 rows × 13 columns

In [129…

```python
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
```

In [130…
```
for i in num[:9]:
    churn_data[i]=ss.fit_transform(churn_data[[i]])
```

In [131…
```
churn_data
```

Out[131…

	gender	age	no_of_days_subscribed	multi_screen	mail_subscribed	weekly_min
0	0	-0.263675	-0.949794	0	0	
1	0	0.030332	1.239136	0	0	
2	0	2.578388	0.660453	0	0	
3	0	-1.439701	0.786254	0	1	
4	0	0.128334	2.295860	0	0	
...	...	...	...	...	...	
1995	0	1.500364	-0.622713	0	1	
1996	1	0.618345	0.685613	0	0	
1997	1	1.402362	-0.144671	0	0	
1998	1	0.128334	-0.144671	0	0	
1999	1	-0.165673	-0.673033	0	0	

2000 rows × 13 columns

In [ ]:

In [ ]: