Task-2: Exploratory data analysis & Data cleaning

Data:

1.ml_case_training_data.csv\ 2.ml_case_training_hist_data.csv\ 3.ml_case_training_output.csv

In [1]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import seaborn as sb
import missingno as msno
```

Loading datasets

In [2]:
```
df1=pd.read_csv('ml_case_training_data.csv')
df2=pd.read_csv('ml_case_training_hist_data.csv')
df3=pd.read_csv('ml_case_training_output.csv')
```

Data exploration

In [3]:
```
df1.head()
```

Out[3]:

	id	activity_new	campaign_disc_ele	
0	48ada52261e7cf58715202705a0451c9	esoiiifxdlbkcsluxmfuacbdckommixw	NaN	lmkeb
1	24011ae4ebbe3035111d65fa7c15bc57	NaN	NaN	foos
2	d29c2c54acc38ff3c0614d0a653813dd	NaN	NaN	
3	764c75f661154dac3a6c254cd082ea7d	NaN	NaN	foos
4	bba03439a292a1e166f80264c16191cb	NaN	NaN	lmkeb

5 rows × 32 columns

In [4]:
```
df2.head()
```

Out[4]:

	id	price_date	price_p1_var	price_p2_var	price_p3_var	pric
0	038af19179925da21a25619c5a24b745	2015-01-01	0.151367	0.0	0.0	4
1	038af19179925da21a25619c5a24b745	2015-02-01	0.151367	0.0	0.0	4
2	038af19179925da21a25619c5a24b745	2015-03-01	0.151367	0.0	0.0	4
3	038af19179925da21a25619c5a24b745	2015-04-01	0.149626	0.0	0.0	4
4	038af19179925da21a25619c5a24b745	2015-05-01	0.149626	0.0	0.0	4

In [5]:
```
df3.head()
```

Out[5]:

	id	churn
0	48ada52261e7cf58715202705a0451c9	0
1	24011ae4ebbe3035111d65fa7c15bc57	1
2	d29c2c54acc38ff3c0614d0a653813dd	0
3	764c75f661154dac3a6c254cd082ea7d	0
4	bba03439a292a1e166f80264c16191cb	0

In [5]:
```
df1.dtypes
```

Out[5]:
```
id                           object
activity_new                 object
campaign_disc_ele           float64
channel_sales                object
cons_12m                      int64
cons_gas_12m                  int64
cons_last_month               int64
date_activ                   object
date_end                     object
date_first_activ             object
date_modif_prod              object
date_renewal                 object
forecast_base_bill_ele      float64
forecast_base_bill_year     float64
forecast_bill_12m           float64
forecast_cons               float64
forecast_cons_12m           float64
forecast_cons_year            int64
forecast_discount_energy    float64
forecast_meter_rent_12m     float64
forecast_price_energy_p1    float64
forecast_price_energy_p2    float64
forecast_price_pow_p1       float64
has_gas                      object
imp_cons                    float64
margin_gross_pow_ele        float64
margin_net_pow_ele          float64
nb_prod_act                   int64
net_margin                  float64
num_years_antig               int64
origin_up                    object
pow_max                     float64
dtype: object
```

In [6]:
```
df2.dtypes
```

Out[6]:
```
id                 object
price_date         object
price_p1_var      float64
price_p2_var      float64
price_p3_var      float64
price_p1_fix      float64
price_p2_fix      float64
price_p3_fix      float64
dtype: object
```

In [7]:
```
df3.dtypes
```

Out[7]:
```
id       object
churn     int64
dtype: object
```

In [8]:
```
df1.shape
```
Out[8]:
```
(16096, 32)
```

In [9]:
```
df2.shape
```
Out[9]:
```
(193002, 8)
```

In [10]:
```
df3.shape
```
Out[10]:
```
(16096, 2)
```

In [11]:
```
df1.describe()
```
Out[11]:

	campaign_disc_ele	cons_12m	cons_gas_12m	cons_last_month	forecast_base_bill_ele
count	0.0	1.609600e+04	1.609600e+04	1.609600e+04	3508.00000C
mean	NaN	1.948044e+05	3.191164e+04	1.946154e+04	335.843857
std	NaN	6.795151e+05	1.775885e+05	8.235676e+04	649.40600C
min	NaN	-1.252760e+05	-3.037000e+03	-9.138600e+04	-364.94000C
25%	NaN	5.906250e+03	0.000000e+00	0.000000e+00	0.00000C
50%	NaN	1.533250e+04	0.000000e+00	9.010000e+02	162.95500C
75%	NaN	5.022150e+04	0.000000e+00	4.127000e+03	396.18500C
max	NaN	1.609711e+07	4.188440e+06	4.538720e+06	12566.08000C

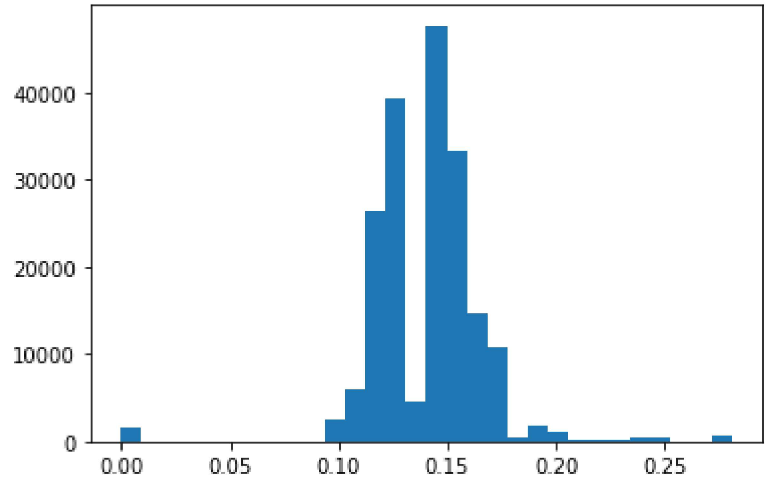
8 rows × 22 columns

Here we can observe mean & median of data features are not equal. So, based on this we can conclude that all features are skewed & do not follow normal distribution.

In [12]:
```
df2.describe()
```
Out[12]:

	price_p1_var	price_p2_var	price_p3_var	price_p1_fix	price_p2_fix	price_p3_f
count	191643.000000	191643.000000	191643.000000	191643.000000	191643.000000	191643.0000(
mean	0.140991	0.054412	0.030712	43.325546	10.698201	6.4554:
std	0.025117	0.050033	0.036335	5.437952	12.856046	7.7822
min	0.000000	0.000000	0.000000	-0.177779	-0.097752	-0.0651
25%	0.125976	0.000000	0.000000	40.728885	0.000000	0.0000(
50%	0.146033	0.085483	0.000000	44.266930	0.000000	0.0000(
75%	0.151635	0.101780	0.072558	44.444710	24.339581	16.2263
max	0.280700	0.229788	0.114102	59.444710	36.490692	17.4582

In [13]:
```
plt.hist(df2['price_p1_var'],bins=30);
```



In df2 price_p1_var have approximately equal mean & median therefore we can visualise that this feature is not skewed or very less skewed & follows normal distibution.

Checking missing values in datasets

In [14]:
```
df1.isnull().sum()
```

Out[14]:
```
id                              0
activity_new                 9545
campaign_disc_ele           16096
channel_sales                4218
cons_12m                        0
cons_gas_12m                    0
cons_last_month                 0
date_activ                      0
date_end                        2
date_first_activ            12588
date_modif_prod               157
date_renewal                   40
forecast_base_bill_ele      12588
forecast_base_bill_year     12588
forecast_bill_12m           12588
forecast_cons               12588
forecast_cons_12m               0
forecast_cons_year              0
forecast_discount_energy      126
forecast_meter_rent_12m         0
forecast_price_energy_p1      126
forecast_price_energy_p2      126
forecast_price_pow_p1         126
has_gas                         0
imp_cons                        0
margin_gross_pow_ele           13
margin_net_pow_ele             13
nb_prod_act                     0
net_margin                     15
num_years_antig                 0
origin_up                      87
pow_max                         3
dtype: int64
```

In [15]:
```
df2.isnull().sum()
```

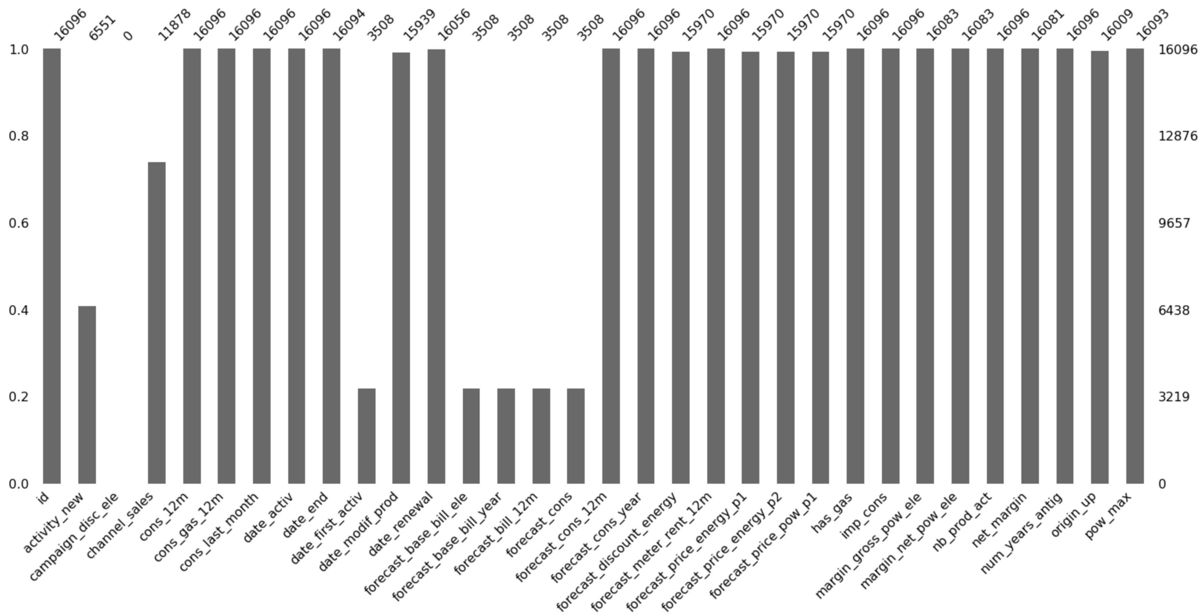
Out[15]:
```
id                 0
price_date         0
price_p1_var    1359
price_p2_var    1359
price_p3_var    1359
price_p1_fix    1359
price_p2_fix    1359
price_p3_fix    1359
dtype: int64
```

In [16]:
```
df3.isnull().sum()
```

Out[16]:
```
id       0
churn    0
dtype: int64
```

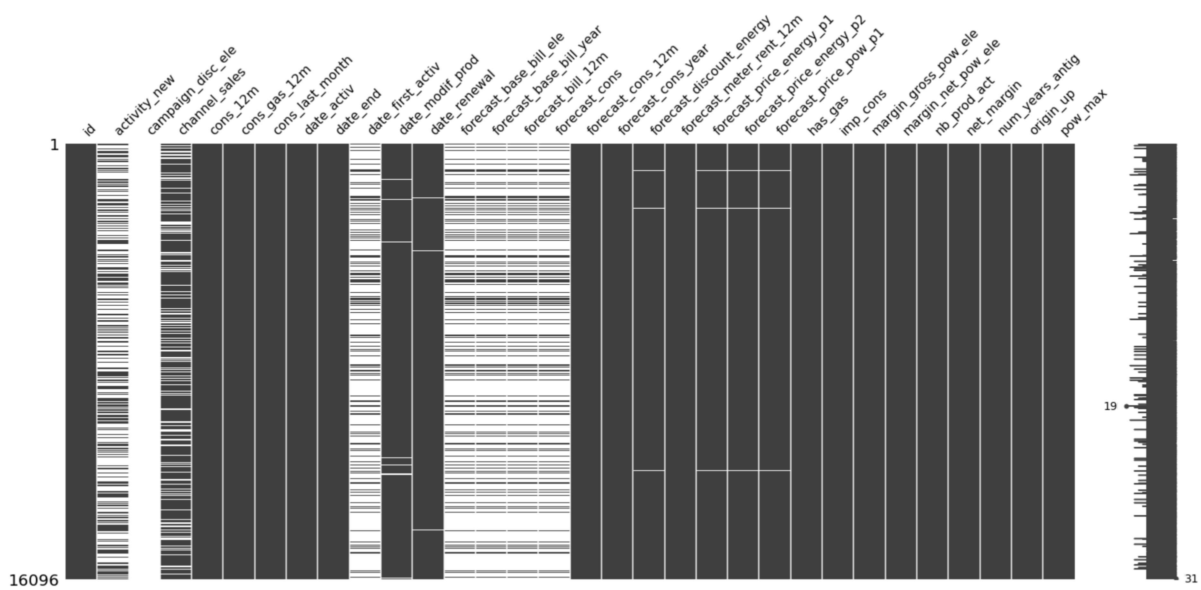
Visualizing missing values

In [17]:
```
msno.bar(df1);
```



In [18]:
```
msno.matrix(df1)
```

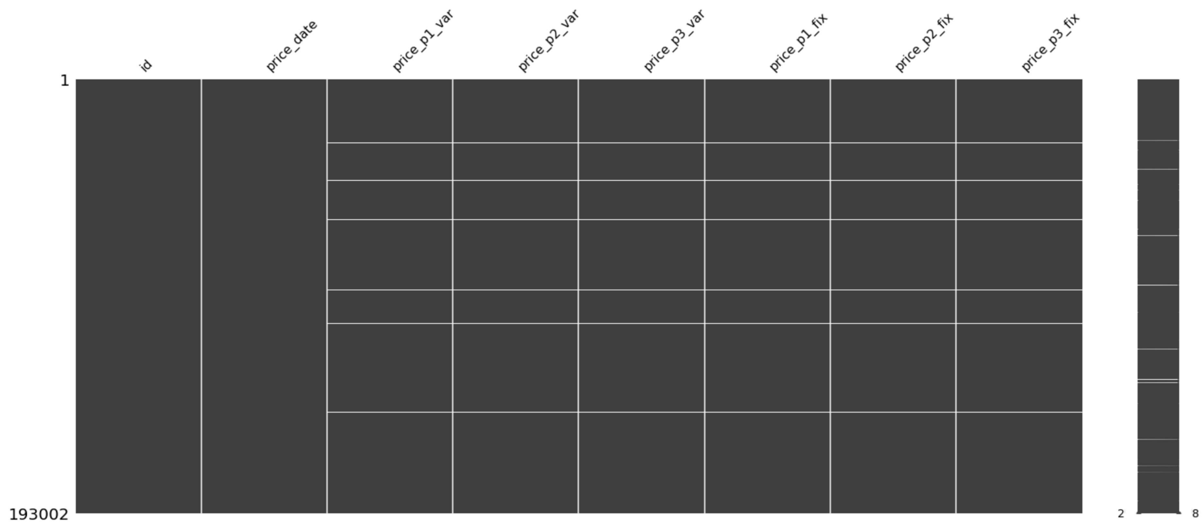
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1728896dfa0>



Campaign_disc_ele is feature in which all values are missing.

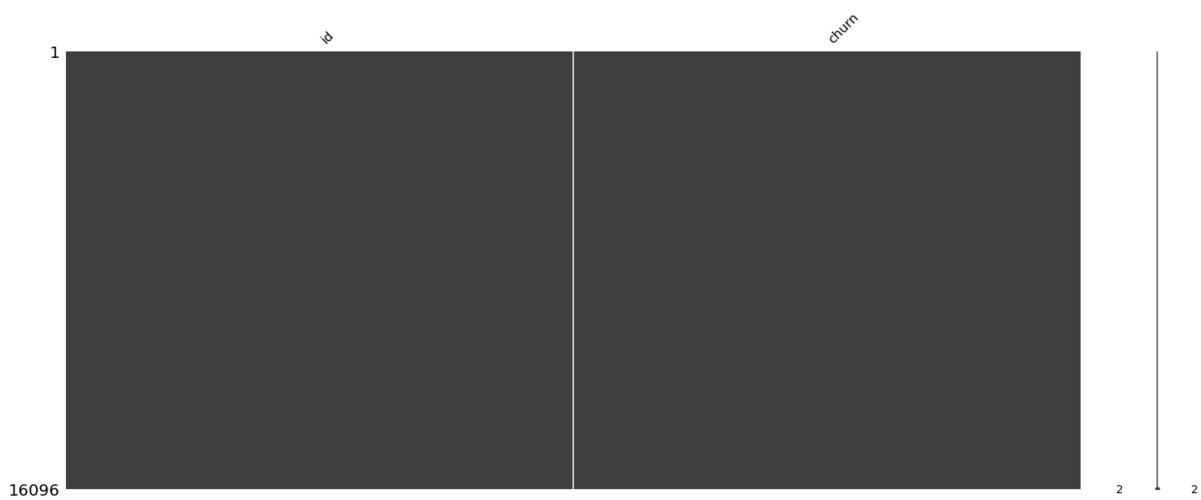
In [19]:
```
msno.matrix(df2)
```

Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x17288cd7df0>
```



In [20]:
```
msno.matrix(df3)
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1728eed2280>
```



There is no missing value present in Churn dataset

Merging main dataset with output dataset

In [21]:
```
df1 = df1.merge(right=df3,on=['id'])
```

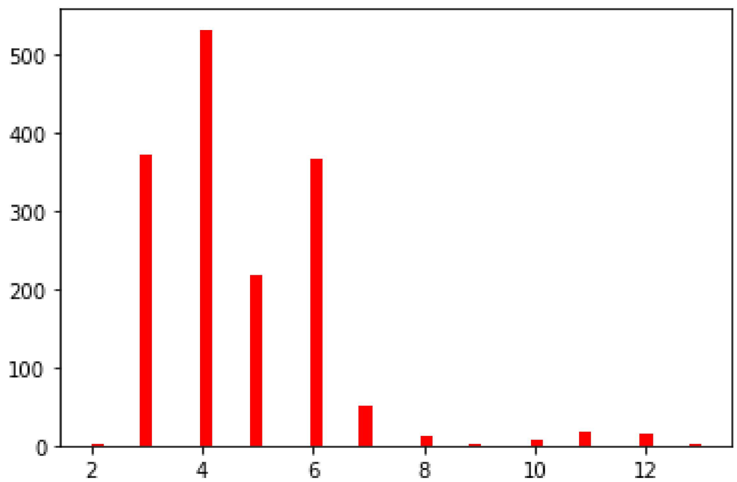
In [22]:
```
df1.head()
```

Out[22]:

	id	activity_new	campaign_disc_ele	
0	48ada52261e7cf58715202705a0451c9	esoiiifxdlbkcsluxmfuacbdckommixw	NaN	lmkeba
1	24011ae4ebbe3035111d65fa7c15bc57	NaN	NaN	foos
2	d29c2c54acc38ff3c0614d0a653813dd	NaN	NaN	
3	764c75f661154dac3a6c254cd082ea7d	NaN	NaN	foos
4	bba03439a292a1e166f80264c16191cb	NaN	NaN	lmkeba

5 rows × 33 columns

In [23]:
```
churn_yes=df1[df1.churn==1].num_years_antig
plt.hist([churn_yes],bins=50,color='red');
```



Maximum customers are leaving in 4th year.

In [24]:
```
top_customers = df1.loc[(df1['churn']>=0) & (df1['net_margin']>0),['id','num_years_antig','net_margin']]
top_customers.sort_values(by=['net_margin'],ascending=False).head(10)
```

Out[24]:

	id	num_years_antig	net_margin
2872	fb7dcb0f4e0dc4ee54874eab2607c4da	3	24570.65
11828	d00e8a9951b5551d8f02e45f9ed2b0dd	3	10203.50
7131	78bd1c5c0c67f2be6de89b19df5f8861	3	5625.14
13622	818b8bca0a9d7668252d46b978169325	4	4346.37
8612	a3a739686fbd5ba8b4a21ec835507b6d	4	4305.79
333	89b3406c3ba717f1b788ceeb5af9e8b9	3	4161.74
12368	4519e6a8928a015819466fc9de0fa49e	3	4040.60
4256	e8948a5469344e9ad0dfcacbb705f709	4	3768.16
6595	933527d7a2f669af49075a2380c10ded	4	3744.72
7048	43580ef6cc40fcfd0a9b76eee17a267a	4	3716.78

The net_margin of 24570.65 is highly obtained customer having id fb7dcb0f4e0dc4ee54874eab2607c4da whose tenure is 3 years.

Replacing missing values

In [25]:
```
df1['campaign_disc_ele']=df1.drop(columns='campaign_disc_ele')
```

Since in campaign_disc_ele there is no value present we can simply delete it.

In [26]:
```
prices=['price_p1_var','price_p2_var','price_p3_var','price_p1_fix','price_p2_fix','price_p3_fix']
for i in prices:
    df2[i]=df2[i].fillna(df2[i].mean())
```

In [27]:
```
df1_columns=['forecast_base_bill_ele','forecast_base_bill_year','forecast_bill_12m','forecast_cons','forecast_discount_energy',
             'forecast_price_energy_p1','forecast_price_energy_p2','forecast_price_pow_p1','margin_gross_pow_ele','margin_net_pow_ele',
            'net_margin','pow_max']
for i in df1_columns:
    df1[i]=df1[i].fillna(df1[i].mean())
```

In [28]:
```
df1.isnull().sum()
```

Out[28]:
```
id                              0
activity_new                 9545
campaign_disc_ele               0
channel_sales                4218
cons_12m                        0
cons_gas_12m                    0
cons_last_month                 0
date_activ                      0
date_end                        2
date_first_activ            12588
date_modif_prod               157
date_renewal                   40
forecast_base_bill_ele          0
forecast_base_bill_year         0
forecast_bill_12m               0
forecast_cons                   0
forecast_cons_12m               0
forecast_cons_year              0
forecast_discount_energy        0
forecast_meter_rent_12m         0
forecast_price_energy_p1        0
forecast_price_energy_p2        0
forecast_price_pow_p1           0
has_gas                         0
imp_cons                        0
margin_gross_pow_ele            0
margin_net_pow_ele              0
nb_prod_act                     0
net_margin                      0
num_years_antig                 0
origin_up                      87
pow_max                         0
churn                           0
dtype: int64
```

In [29]:
```
df3.describe()
```

Out[29]:

	churn
count	16096.000000
mean	0.099093
std	0.298796
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

Outliers cleaning

In [30]:
```
columns=['cons_12m','cons_gas_12m','cons_last_month','forecast_base_bill_ele','forecast_base_bill_year','forecast_bill_12m',
        'forecast_cons','forecast_cons_12m','forecast_cons_year','forecast_price_energy_p1','forecast_price_energy_p2',
        'forecast_price_pow_p1','imp_cons','margin_gross_pow_ele','margin_net_pow_ele','nb_prod_act',
        'net_margin','num_years_antig','pow_max']

for i in columns:
    Q1=df1[i].quantile(.25)
    Q3=df1[i].quantile(.75)
    IQR=Q3-Q1
    lower_limit=Q1-1.5*(IQR)
    upper_limit=Q3+1.5*(IQR)
    def limit_imputer(value):
        if value>upper_limit:
            return upper_limit
        if value<lower_limit:
            return lower_limit
        else:
            return value
    df1[i]=df1[i].apply(limit_imputer)
```

In [31]:
```
columns_df2=['price_p1_var','price_p2_var','price_p3_var','price_p1_fix','price_p2_fix','price_p3_fix']
for i in columns_df2:
    Q1=df2[i].quantile(.25)
    Q3=df2[i].quantile(.75)
    IQR=Q3-Q1
    lower_limit=Q1-1.5*(IQR)
    upper_limit=Q3+1.5*(IQR)

    def limit_imputer(value):
        if value>upper_limit:
            return upper_limit
        if value<lower_limit:
            return lower_limit
        else:
            return value
    df2[i]=df2[i].apply(limit_imputer)
```

Replacing categorical values in has_gas column to 0 & 1.

In [32]:
```
df1['has_gas'].replace({'f':0,'t':1},inplace=True)
```

In [33]:
```
df1['has_gas'].unique()
```

Out[33]:
```
array([0, 1], dtype=int64)
```

In [34]:
```
customers_electricity_and_gas=sum(df1['has_gas']==1)
customers_electricity=sum(df1['has_gas']==0)
```

In [35]:
```
print("Number of electricity clients:",customers_electricity)
print("Number of electricity & gas clients:",customers_electricity_and_gas)
```

```
Number of electricity clients: 13132
Number of electricity & gas clients: 2964
```

In [36]:
```
clients_churn=sum(df3['churn']==1)
clients_not_churn=sum(df3['churn']==0)
```

In [37]:
```
print("Number of customers who are churned over next 3 months is",clients_chur
n)
print("Number of customers who are not churned over next 3 months is",clients_
not_churn)
```

```
Number of customers who are churned over next 3 months is 1595
Number of customers who are not churned over next 3 months is 14501
```

Number of customers churned over next 3 months is less than cutomers who are not churned.

In [48]:
```
df2.describe()
```

Out[48]:

	price_p1_var	price_p2_var	price_p3_var	price_p1_fix	price_p2_fix	price_p3_f
count	193002.000000	193002.000000	193002.000000	193002.000000	193002.000000	193002.0000(
mean	0.140937	0.054412	0.030712	43.185655	10.698201	6.4554:
std	0.019564	0.049857	0.036207	2.439974	12.810704	7.7548:
min	0.087488	0.000000	0.000000	35.155148	-0.097752	-0.0651i
25%	0.125976	0.000000	0.000000	40.728885	0.000000	0.0000(
50%	0.145859	0.085100	0.000000	44.266930	0.000000	0.0000(
75%	0.151635	0.101673	0.072558	44.444710	24.339581	16.2263{
max	0.190123	0.229788	0.114102	50.018447	36.490692	17.4582:

Average price of of energy for 1st period was 0.140937\ Average price of of energy for 2nd period was 0.054412\ Average price of of energy for 3rd period was 0.030712

Average price of of power for 1st period was 43.185655\ Average price of of power for 2nd period was 10.698201\ Average price of of power for 3rd period was 6.455436

Average number of tenure is 5 years