BCG- TASK 2

July 07, 2023

1 Libraries

[350]:
```python
# Data analysis and wrangling
import pandas as pd
import numpy as np

# Data visualisation
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# Dates
import datetime

#remove warnings
import warnings
warnings.filterwarnings('ignore')
```

2 Import and read data

[351]:
```python
churn_data=pd.read_csv('ml_case_training_output.csv')
churn_data.head()
```

[351]:
```
                                 id churn
0  48ada52261e7cf58715202705a0451c9     0
1  24011ae4ebbe3035111d65fa7c15bc57     1
2  d29c2c54acc38ff3c0614d0a653813dd     0
3  764c75f661154dac3a6c254cd082ea7d     0
4  bba03439a292a1e166f80264c16191cb     0
```

[352]:
```python
 churn_data['churn'] =
churn_data['churn'].replace({0:'stayed',1:'churned'})
churn_data.head()
```

[352]:
```
                                 id    churn
0  48ada52261e7cf58715202705a0451c9   stayed
1  24011ae4ebbe3035111d65fa7c15bc57  churned
2  d29c2c54acc38ff3c0614d0a653813dd   stayed
3  764c75f661154dac3a6c254cd082ea7d   stayed
4  bba03439a292a1e166f80264c16191cb   stayed
```

[353]:
```
history_data=pd.read_csv('ml_case_training_hist_data.csv')
history_data.head()
```

[353]:
```
                                 id price_date price_p1_var price_p2_var \
0  038af19179925da21a25619c5a24b745 2015-01-01   0.151367   0.0
1  038af19179925da21a25619c5a24b745 2015-02-01   0.151367   0.0
2  038af19179925da21a25619c5a24b745 2015-03-01   0.151367   0.0
3  038af19179925da21a25619c5a24b745 2015-04-01   0.149626   0.0
4  038af19179925da21a25619c5a24b745 2015-05-01   0.149626   0.0

   price_p3_var price_p1_fix price_p2_fix price_p3_fix
0    0.0   44.266931   0.0   0.0 1 0.0   44.266931
     0.0   0.0 2 0.0   44.266931   0.0   0.0 3 0.0
     44.266931  0.0   0.0
4         0.0    44.266931         0.0          0.0
```

[354]:
```
training_data=pd.read_csv('ml_case_training_data.csv')
training_data.tail()
```

[354]:
```
                                     id activity_new campaign_disc_ele \
16091 18463073fb097fc0ac5d3e040f356987     NaN                   NaN
16092 d0a6f71671571ed83b2645d23af6de00     NaN                   NaN
16093 10e6828ddd62cbcf687cb74928c4c2d2     NaN                   NaN
16094 1cf20fd6206d7678d5bcafd28c53b4db     NaN                   NaN
16095 563dde550fd624d7352f3de77c0cdfcd     NaN                   NaN

                        channel_sales cons_12m cons_gas_12m \
16091                    foosdfpfkusacimwkcsosbicdxkicaua
                           32270 47940
16092                    foosdfpfkusacimwkcsosbicdxkicaua
                           7223  0
16093                    foosdfpfkusacimwkcsosbicdxkicaua
                           1844  0
16094                    foosdfpfkusacimwkcsosbicdxkicaua
                           131   0
16095                    NaN    8730  0
      cons_last_month date_activ date_end date_first_activ … \
16091             0  2012-05-24 2016-05-08  NaN …
16092             181  2012-08-27 2016-08-27     2012-08-27 …
                  16093    179 2012-02-08 2016-02-07   NaN …
16094             0  2012-08-30 2016-08-30  NaN …
16095             0  2009-12-18 2016-12-17  NaN …

      forecast_price_pow_p1 has_gas imp_cons margin_gross_pow_ele \
16091           44.311378  t     0.00  27.88
16092           58.995952  f     15.94 0.00
16093          40.606701      f    18.05             39.84
```

```
      16094         44.311378      f     0.00              13.08
      16095         45.311378      f     0.00              11.84
        margin_net_pow_ele nb_prod_act net_margin num_years_antig \
16091            27.88     2     381.77     4

16092            0.00      1     90.34 3

16093            39.84     1     20.38 4

16094            13.08     1     0.96  3

16095            11.84     1     96.34 6

origin_up pow_max

16091 lxidpiddsbxsbosboudacockeimpuepw 15.000

16092 lxidpiddsbxsbosboudacockeimpuepw6.000

16093 lxidpiddsbxsbosboudacockeimpuepw15.935

16094 lxidpiddsbxsbosboudacockeimpuepw11.000

16095 ldkssxwpmemidmecebumciepifcamkci10.392

      [5 rows x 32 columns]
```

[355]:
```
merge=pd.merge(churn_data, training_data, on='id')
merge.tail()
```

[355]:
```
                                      id  churn activity_new \
      16091 18463073fb097fc0ac5d3e040f356987stayed      NaN
      16092 d0a6f71671571ed83b2645d23af6de00 churned    NaN
      16093 10e6828ddd62cbcf687cb74928c4c2d2 churned    NaN
      16094 1cf20fd6206d7678d5bcafd28c53b4dbstayed      NaN
      16095 563dde550fd624d7352f3de77c0cdfcdstayed      NaN
          campaign_disc_ele                   channel_sales cons_12m \
      16091             NaN foosdfpfkusacimwkcsosbicdxkicaua  32270
      16092             NaN foosdfpfkusacimwkcsosbicdxkicaua   7223
      16093             NaN foosdfpfkusacimwkcsosbicdxkicaua   1844
      16094             NaN foosdfpfkusacimwkcsosbicdxkicaua    131
      16095               NaN                             NaN    8730
          cons_gas_12m cons_last_month date_activ  date_end … \
      16091      47940               0 2012-05-24 2016-05-08
                                       …
      16092           0            181  2012-08-27  2016-08-27
                                     …
      16093           0            179  2012-02-08  2016-02-07
                                     …
      16094           0               0 2012-08-30 2016-08-30
                                       …
      16095           0              0 2009-12-18 2016-12-17 …
     forecast_price_pow_p1 has_gas imp_cons margin_gross_pow_ele \
```

```
16091            44.311378  t     0.00  27.88
16092            58.995952  f     15.94 0.00
16093           40.606701      f   18.05                39.84
16094           44.311378      f    0.00                13.08
16095           45.311378      f    0.00                11.84
          margin_net_pow_ele nb_prod_act net_margin num_years_antig \
16091             27.88     2     381.77     4

16092             0.00      1     90.34 3

16093             39.84     1     20.38 4

16094             13.08     1     0.96  3

16095             11.84     1     96.34 6

origin_up pow_max

16091 lxidpiddsbxsbosboudacockeimpuepw 15.000

16092 lxidpiddsbxsbosboudacockeimpuepw6.000

16093 lxidpiddsbxsbosboudacockeimpuepw15.935

16094 lxidpiddsbxsbosboudacockeimpuepw11.000

16095 ldkssxwpmemidmecebumciepifcamkci10.392

[5 rows x 33 columns]
```

3 CHURN DATA

[356]:
```
churn_data.count()
```

[356]:
```
id  16096 churn
    16096
dtype: int64
```

[357]:
```
churn_count=churn_data['churn'].value_counts()
print(churn_count)
```

```
stayed      14501
churned     1595
Name: churn, dtype: int64
```

It can be seen that, the number of companies that have churned out is

[358]:
```
rate_of_churn = pd.DataFrame(churn_data['churn'].value_counts() / churn_data.
 ↪shape[0] * 100)
print(rate_of_churn )
```

```
            churn
```

```
stayed     90.090706
churned    9.909294
```

It can be seen that, the number of companies that have churned out is 1595 which represent 9.9%, approximately, 10%.

[359]:
```
#changing the column names
merge['churn'] = merge['churn'].replace({0:'stayed',1:'churned'})
merge.head()
```

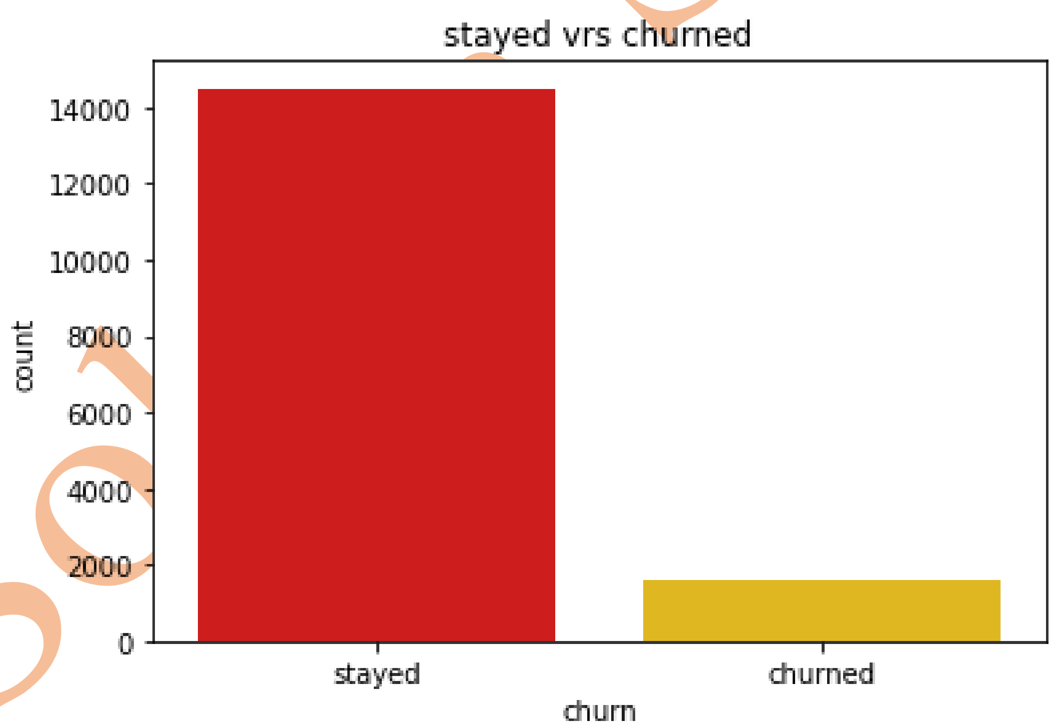
[359]:
```
                                 id    churn \
0  48ada52261e7cf58715202705a0451c9   stayed
1  24011ae4ebbe3035111d65fa7c15bc57  churned
2  d29c2c54acc38ff3c0614d0a653813dd   stayed
3  764c75f661154dac3a6c254cd082ea7d   stayed
4  bba03439a292a1e166f80264c16191cb   stayed

                      activity_new campaign_disc_ele \
0 esoiiifxdlbkcsluxmfuacbdckommixw               NaN
1                              NaN               NaN
2                              NaN               NaN
3                              NaN               NaN
4                              NaN               NaN
                     channel_sales cons_12m cons_gas_12m cons_last_month \
0 lmkebamcaaclubfxadlmueccxoimlema309275           0          10025
1 foosdfpfkusacimwkcsosbicdxkicaua     0       54946               0
2                              NaN   4660           0               0
3 foosdfpfkusacimwkcsosbicdxkicaua   544           0               0
4 lmkebamcaaclubfxadlmueccxoimlema  1584           0               0

  date_activdate_end … forecast_price_pow_p1 has_gas imp_cons \
0 2012-11-07 2016-11-06 … 58.995952 f 831.8 1 2013-06-15
2016-06-15 … 40.606701 t 0.0
2  2009-08-21 2016-08-30 …  44.311378  f     0.0
3  2010-04-16 2016-04-16 …  44.311378  f     0.0
4  2010-03-30 2016-03-30 …  44.311378  f     0.0
  margin_gross_pow_ele margin_net_pow_ele nb_prod_act net_margin \
0              -41.76-41.76      1     1732.36
1              25.44 25.44 2     678.99
2             16.38              16.38           1      18.89
3             28.60              28.60           1       6.60
```

```
4                  30.22              30.22            1      25.46

  num_years_antig                        origin_up pow_max
0                3 ldkssxwpmemidmecebumciepifcamkci
                 180.000
1                3 lxidpiddsbxsbosboudacockeimpuepw
                  43.648
2                6 kamkkxfxxuwbdslkwifmmcsiusiuosws
                  13.800
3                6 kamkkxfxxuwbdslkwifmmcsiusiuosws
                  13.856
4                6 kamkkxfxxuwbdslkwifmmcsiusiuosws
                  13.200
[5 rows x 33 columns]
```

4 Data visualization of churn

[360]:
```
sns.countplot(x= 'churn', data = churn_data, palette = 'hot')
plt.title('stayed vrs churned')
```

[360]:
```
Text(0.5, 1.0, 'stayed vrs churned')
```



5 Describing data

[361]:
```
merge.info()
```

```
<class
'pandas.core.frame.DataFrame'>
Int64Index: 16096 entries, 0 to
16095 Data columns (total 33
columns):
#   Column                    Non-Null Count
                              Dtype
--- ------                    -------------- ----
                              -
0    id                       16096 non-null
                              object
1   churn                     16096 non-null
                              object
2   activity_new              6551 non-null object
3   campaign_disc_ele         0 non-null   float64
4   channel_sales             11878 non-null
                              object
5   cons_12m                  16096 non-null
                              int64
6   cons_gas_12m              16096 non-null
                              int64
 7  cons_last_month           16096 non-null
                              int64
 8  date_activ                16096 non-null
                              object
 9  date_end                  16094 non-null
                              object
 10 date_first_activ          3508 non-null object
 11 date_modif_prod           15939 non-null
                              object
 12 date_renewal              16056 non-null
                              object
 13 forecast_base_bill_ele 3508 non-null
                                     float64
 14 forecast_base_bill_year3508 non-null
                                     float64
 15 forecast_bill_12m         3508 non-null
                                     float64
 16 forecast_cons             3508 non-null
                                     float64
 17 forecast_cons_12m         16096      non-null
                              float64
 18 forecast_cons_year        16096 non-null
                              int64
 19  forecast_discount_energy 15970 non-null float64
 20  forecast_meter_rent_12m16096 non-null float64
 21  forecast_price_energy_p1 15970 non-null float64
```

```
 22  forecast_price_energy_p2 15970 non-null float64
 23 forecast_price_pow_p1  15970       non-null
                        float64
 24 has_gas                16096 non-null
                        object
 25 imp_cons               16096       non-null
                        float64
 26 margin_gross_pow_ele   16083       non-null
                        float64
 27 margin_net_pow_ele     16083       non-null
                        float64
 28 nb_prod_act            16096 non-null
                        int64
 29 net_margin             16081       non-null
                        float64
 30 num_years_antig        16096 non-null
                        int64
 31 origin_up              16009 non-null
                        object
 32 pow_max                16093       non-null
                        float64
dtypes: float64(16), int64(6),
object(11) memory usage: 4.2+ MB
```

It can be seen that the types of date is object, but needs to be in datetime.

[362]:
```
merge.describe()
```

[362]:
```
     campaign_disc_ele    cons_12m cons_gas_12m cons_last_month \
count               0.0 1.609600e+04             1.609600e+04
                  1.609600e+04
mean                NaN 1.948044e+05             1.946154e+04
                  3.191164e+04
std                 NaN 6.795151e+05             8.235676e+04
                  1.775885e+05
min                 NaN -1.252760e+05 -           -
                  3.037000e+03              9.138600e+04
25%                 NaN 5.906250e+03             0.000000e+00
                  0.000000e+00
50%                 NaN 1.533250e+04             9.010000e+02
                  0.000000e+00
75%                 NaN 5.022150e+04             4.127000e+03
                  0.000000e+00
max                 NaN 1.609711e+07             4.538720e+06
                  4.188440e+06
forecast_base_bill_ele forecast_base_bill_year forecast_bill_12m \
count         3508.000000          3508.000000      3508.000000
mean           335.843857           335.843857      3837.441866
std            649.406000           649.406000      5425.744327
```

min	-364.940000	-364.940000	-2503.480000
25%	0.000000	0.000000	1158.175000
50%	162.955000	162.955000	2187.230000
75%	396.185000	396.185000	4246.555000
max	12566.080000	12566.080000	81122.630000

	forecast_cons	forecast_cons_12m	forecast_cons_year	… \
count	3508.000000	16096.000000	16096.000000	…
mean	206.845165	2370.555949	1907.347229	…
std	455.634288	4035.085664	5257.364759	…
min	0.000000	-16689.260000	-85627.000000	…
25%	0.000000	513.230000	0.000000	…
50%	42.215000	1179.160000	378.000000	…
75%	228.117500	2692.077500	1994.250000	…
max	9682.890000	103801.930000	175375.000000	…

	forecast_price_energy_p1	forecast_price_energy_p2 \
count	15970.000000	15970.000000
mean	0.135901	0.052951
std	0.026252	0.048617
min	0.000000	0.000000
25%	0.115237	0.000000
50%	0.142881	0.086163
75%	0.146348	0.098837
max	0.273963	0.195975

	forecast_price_pow_p1	imp_cons	margin_gross_pow_ele \
count	15970.000000	16096.000000	16083.000000
mean	43.533496	196.123447	22.462276
std	5.212252	494.366979	23.700883
min	-0.122184	-9038.210000	-525.540000
25%	40.606701	0.000000	11.960000
50%	44.311378	44.465000	21.090000
75%	44.311378	218.090000	29.640000
max	59.444710	15042.790000	374.640000

	margin_net_pow_ele	nb_prod_act	net_margin	num_years_antig \
count	16083.000000	16096.000000	16081.000000	16096.000000
mean	21.460318	1.347788	217.987028	5.030629
std	27.917349	1.459808	366.742030	1.676101
min	-615.660000	1.000000	-4148.990000	1.000000
25%	11.950000	1.000000	51.970000	4.000000
50%	20.970000	1.000000	119.680000	5.000000
75%	29.640000	1.000000	275.810000	6.000000

```
max         374.640000   32.000000 24570.650000     16.000000

             pow_max
count 16093.000000
mean     20.604131
std      21.772421
min       1.000000
25%      12.500000
50%      13.856000
75%      19.800000
max     500.000000
[8 rows x 22 columns]
```

[363]:
```python
merge['has_gas'] = merge['has_gas'].replace({'f':'No','t':'Yes'})
merge.head()
```

[363]:
```
                                 id   churn  \
0  48ada52261e7cf58715202705a0451c9   stayed
1  24011ae4ebbe3035111d65fa7c15bc57 churned
2  d29c2c54acc38ff3c0614d0a653813dd   stayed
3  764c75f661154dac3a6c254cd082ea7d   stayed
4  bba03439a292a1e166f80264c16191cb   stayed

                      activity_new campaign_disc_ele \
0 esoiiifxdlbkcsluxmfuacbdckommixw               NaN
1                              NaN               NaN
2                              NaN               NaN
3                              NaN               NaN
4                              NaN               NaN
                    channel_sales cons_12m cons_gas_12m cons_last_month \
0 lmkebamcaaclubfxadlmueccxoimlema309275            0           10025
1 foosdfpfkusacimwkcsosbicdxkicaua     0        54946               0
2                              NaN     4660            0               0
3 foosdfpfkusacimwkcsosbicdxkicaua   544            0               0
4 lmkebamcaaclubfxadlmueccxoimlema  1584            0               0

  date_activ date_end … forecast_price_pow_p1 has_gas imp_cons \
0  2012-11-07 2016-11-06 …  58.995952  No    831.8
1  2013-06-15 2016-06-15 …  40.606701  Yes   0.0
2  2009-08-21 2016-08-30 …  44.311378  No    0.0
3  2010-04-16 2016-04-16 …  44.311378  No    0.0
4  2010-03-30 2016-03-30 …  44.311378  No    0.0
  margin_gross_pow_ele margin_net_pow_ele nb_prod_act net_margin \
0               -41.76             -41.76           1    1732.36
1                25.44              25.44           2     678.99
2                16.38              16.38           1      18.89
3                28.60              28.60           1       6.60
```

```
4                 30.22              30.22          1      25.46
  num_years_antig                        origin_up pow_max
0                3 ldkssxwpmemidmecebumciepifcamkci 180.000
1                3 lxidpiddsbxsbosboudacockeimpuepw     43.648
2                6 kamkkxfxxuwbdslkwifmmcsiusiuosws     13.800
3                6 kamkkxfxxuwbdslkwifmmcsiusiuosws     13.856
4                6 kamkkxfxxuwbdslkwifmmcsiusiuosws     13.200

[5 rows x 33 columns]
```

[364]:
```
history_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193002 entries, 0 to
193001 Data columns (total 8
columns):
 #   Column      Non-Null Count  Dtype
--- ------ --------------   ----0 id
     193002 non-null object
 1   price_date 193002 non-null object
 2   price_p1_var 191643 non-null float64
 3   price_p2_var 191643 non-null float64
 4   price_p3_var 191643 non-null float64
 5   price_p1_fix 191643 non-null float64
 6   price_p2_fix 191643 non-null float64
 7   price_p3_fix 191643 non-null float64 dtypes: float64(6),
 object(2) memory usage: 11.8+ MB
```

[365]:
```
history_data.describe()
```

[365]:
```
price_p1_var price_p2_var price_p3_var price_p1_fix \ count
191643.000000 191643.000000 191643.000000 191643.000000
mean       0.140991     0.054412     0.030712     43.325546
std        0.025117     0.050033     0.036335      5.437952
min        0.000000     0.000000     0.000000     -0.177779
25%        0.125976     0.000000     0.000000     40.728885
50%        0.146033     0.085483     0.000000     44.266930
75%        0.151635     0.101780     0.072558     44.444710
max        0.280700     0.229788     0.114102     59.444710

      price_p2_fix price_p3_fix
count 191643.000000
191643.000000 mean    10.698201
      6.455436 std    12.856046
      7.782279 min    -0.097752
      -0.065172 25%    0.000000
      0.000000
```

```
50%    0.000000    0.000000    75%
24.339581 16.226389
```

```
max      36.490692    17.458221
```

6 Checking for missing data

[366]:
```
missing_figures_1 = history_data.isnull().sum()
missing_figures_1 = missing_figures_1[missing_figures_1
> 0]
pd.DataFrame({"missing_figures 1": missing_figures_1,
"Missing_values_1(%)":␣
↪history_data.isnull().sum()/len(history_data.index)*100}).sort_va
lues(by =␣
↪"Missing_values_1(%)", ascending = False)
```

[366]:
```
missing_figures 1 Missing_values_1(%)
price_p1_fix    1359.0      0.704138
price_p1_var    1359.0      0.704138
price_p2_fix    1359.0      0.704138
price_p2_var    1359.0      0.704138
price_p3_fix    1359.0      0.704138
price_p3_var    1359.0      0.704138 id      NaN
     0.000000 price_date   NaN   0.000000
```

[367]:
```
missing_figures = merge.isnull().sum() missing_figures =
missing_figures[missing_figures > 0]
pd.DataFrame({"missing_figures": missing_figures, "Missing values
(%)": merge.
↪isnull().sum()/len(merge.index)*100}).sort_values(by = "Missing
values (%)",␣ ↪ascending = False)
```

[367]:
```
missing_figures Missing values (%) campaign_disc_ele
     16096.0    100.000000 date_first_activ12588.0
     78.205765 forecast_base_bill_ele 12588.0    78.205765
forecast_cons   12588.0    78.205765 forecast_bill_12m
     12588.0    78.205765 forecast_base_bill_year
     12588.0    78.205765 activity_new      9545.0
```

```
    59.300447 channel_sales     4218.0     26.205268
date_modif_prod 157.0 0.975398 forecast_price_pow_p1   126.0
    0.782803 forecast_price_energy_p2      126.0 0.782803
forecast_discount_energy   126.0 0.782803
forecast_price_energy_p1   126.0 0.782803 origin_up    87.0
    0.540507 date_renewal 40.0  0.248509 net_margin   15.0
    0.093191 margin_gross_pow_ele    13.0  0.080765
margin_net_pow_ele    13.0  0.080765 pow_max 3.0   0.018638
date_end                          2.0          0.012425
forecast_meter_rent_12m           NaN          0.000000
forecast_cons_year                NaN          0.000000
date_activ                        NaN          0.000000
has_gas                           NaN          0.000000
id                                NaN          0.000000
imp_cons                          NaN          0.000000
cons_last_month                   NaN          0.000000
cons_gas_12m                      NaN          0.000000
nb_prod_act                       NaN          0.000000
cons_12m                          NaN          0.000000
num_years_antig                   NaN          0.000000
churn                             NaN          0.000000
forecast_cons_12m                 NaN          0.000000
```

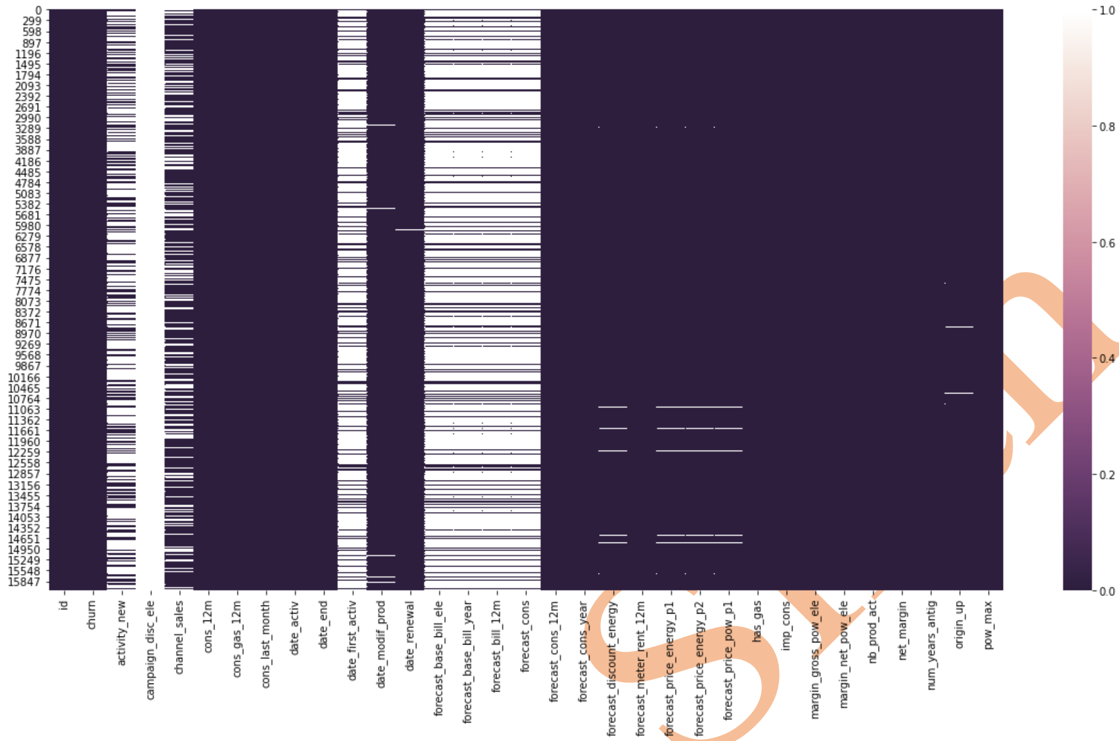
We need to drop columns with alot of missing data. Hence all columns with more than 70% missing data should be dropped

The history data looks good with less than 1% missing data.

7 Visualization of missing figures

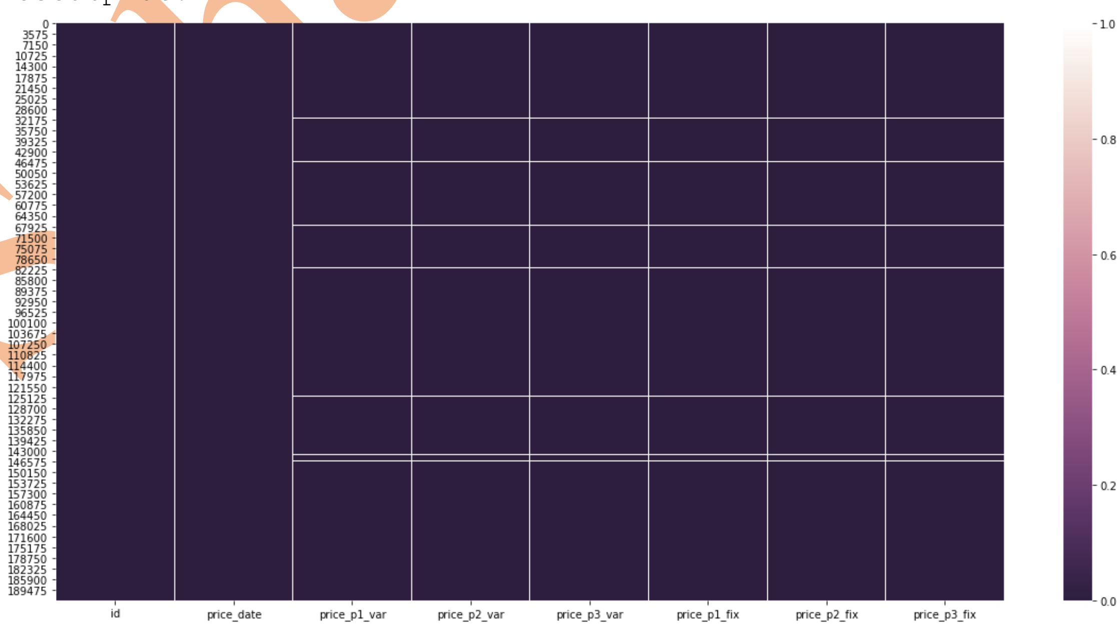
[368]:
```
plt.figure(figsize=(20, 10)) cmap = sns.cubehelix_palette(light=1, as_cmap=True, reverse=True) sns.heatmap(merge.isnull(), cmap=cmap)
```

[368]: <AxesSubplot:>



[369]:
```
plt.figure(figsize=(20, 10)) cmap = sns.cubehelix_palette(light=1, as_cmap=True, reverse=True) sns.heatmap(history_data.isnull(), cmap=cmap)
```

[369]: <AxesSubplot:>



[370]:
```
pip install missingno
```

```
Requirement already satisfied: missingno in
/Users/barbarazen/anaconda3/lib/python3.8/site-packages (0.4.2)
Requirement already satisfied: numpy in
/Users/barbarazen/anaconda3/lib/python3.8/site-packages (from
missingno) (1.19.5)
Requirement already satisfied: matplotlib in
/Users/barbarazen/anaconda3/lib/python3.8/site-packages (from
missingno) (3.3.2)
Requirement already satisfied: scipy in
/Users/barbarazen/anaconda3/lib/python3.8/site-packages (from
missingno) (1.5.2)
Requirement already satisfied: seaborn in
/Users/barbarazen/anaconda3/lib/python3.8/site-packages (from
missingno) (0.11.0)
Requirement already satisfied:
pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
/Users/barbarazen/anaconda3/lib/python3.8/site-packages (from
matplotlib->missingno) (2.4.7)
Requirement already satisfied: certifi>=2020.06.20 in
/Users/barbarazen/anaconda3/lib/python3.8/site-packages (from
matplotlib->missingno) (2020.6.20)
Requirement already satisfied: python-dateutil>=2.1 in
/Users/barbarazen/anaconda3/lib/python3.8/site-packages (from
matplotlib->missingno) (2.8.1)
Requirement already satisfied: pillow>=6.2.0 in
/Users/barbarazen/anaconda3/lib/python3.8/site-packages (from
matplotlib->missingno) (8.0.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Users/barbarazen/anaconda3/lib/python3.8/site-packages (from
matplotlib->missingno) (1.3.0)
Requirement already satisfied: cycler>=0.10 in
/Users/barbarazen/anaconda3/lib/python3.8/site-
packages (from matplotlib->missingno) (0.10.0)
Requirement already satisfied: six in
/Users/barbarazen/anaconda3/lib/python3.8/site-
packages (from cycler>=0.10->matplotlib->missingno)
(1.15.0) Requirement already satisfied: pandas>=0.23
in
/Users/barbarazen/anaconda3/lib/python3.8/site-packages (from
seaborn->missingno) (1.1.3)
Requirement already satisfied: pytz>=2017.2 in
/Users/barbarazen/anaconda3/lib/python3.8/site-packages (from
pandas>=0.23->seaborn->missingno) (2020.1)
Note: you may need to restart the kernel to use updated packages.
```

[371]:
```python
import missingno as msno
```

The missingno correlation heatmap measures nullity correlation: how strongly the presence or absence of one variable affects the presence of another:

Nullity correlation ranges from -1 (if one variable appears the other definitely does not) to 0 (variables appearing or not appearing have no effect on one another) to 1 (if one variable appears the other definitely also does).

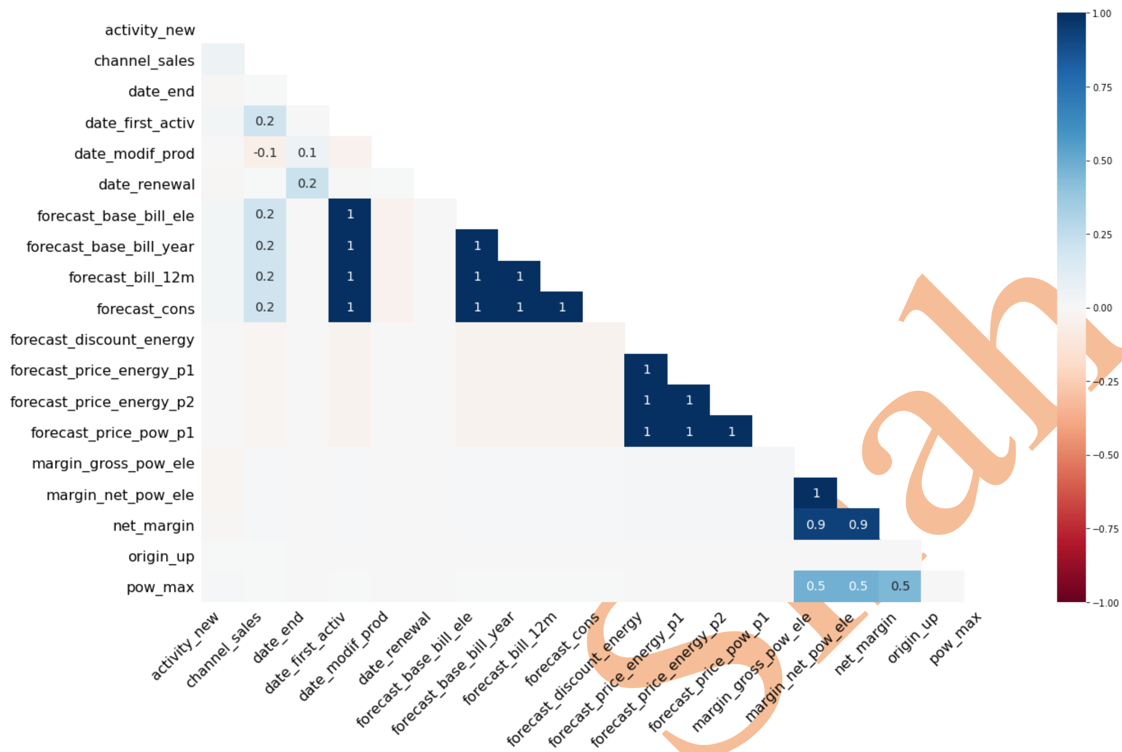
Variables that are always full or always empty have no meaningful correlation, and so are silently removed from the visualization—in this case for instance the datetime and injury number columns, which are completely filled, are not included.

[372]:
```python
msno.heatmap(history_data)
```

[372]: <AxesSubplot:>

[373]:
```python
msno.heatmap(merge)
```

[373]: <AxesSubplot:>



Variables that are always full or always empty have no meaningful correlation, and so are silently removed from the visualization—in this case for instance the datetime and injury number columns, which are completely filled, are not included. It can be seen that, the columns with alot of missing data is already dropped from the correlation heatmap.

We need to drop columns with alot of missing data. Hence all columns with more than 70% missing data should be dropped

8 Dropping missing figures

[374]:
```
merge=merge.drop(columns= ["forecast_base_bill_ele", "date_first_activ",␣
↪"campaign_disc_ele", "forecast_base_bill_year", "forecast_bill_12m",␣
↪"forecast_cons", ])
merge.head()
```

[374]:
```
                                 id    churn  \
0  48ada52261e7cf58715202705a0451c9   stayed
1  24011ae4ebbe3035111d65fa7c15bc57  churned
2  d29c2c54acc38ff3c0614d0a653813dd   stayed
3  764c75f661154dac3a6c254cd082ea7d   stayed
4  bba03439a292a1e166f80264c16191cb   stayed
```

```
                   activity_new                  channel_sales \
0                          esoiiifxdlbkcsluxmfuacbdckommixw
                           lmkebamcaaclubfxadlmueccxoimlema
1                          NaN foosdfpfkusacimwkcsosbicdxkicaua
2                          NaN NaN
3                          NaN foosdfpfkusacimwkcsosbicdxkicaua
4                          NaN lmkebamcaaclubfxadlmueccxoimlema

 cons_12m cons_gas_12m cons_last_month date_activ  date_end \
0        309275 0     10025 2012-11-07 2016-11-06
1        0 54946 0  2013-06-15 2016-06-15
2        4660    0      0 2009-08-21 2016-08-30
3        544  0  0 2010-04-16 2016-04-16 4 1584 0 0 2010-03-30 2016-
         03-30

  date_modif_prod … forecast_price_pow_p1 has_gas imp_cons \
0     2012-11-07 …           58.995952     No    831.8
1           NaN …            40.606701    Yes     0.0
2     2009-08-21 …           44.311378     No      0.0
3     2010-04-16 …           44.311378     No      0.0
4     2010-03-30 …           44.311378     No      0.0
margin_gross_pow_ele margin_net_pow_ele nb_prod_act net_margin \
0                 -41.76-41.76      1     1732.36
1                 25.44  25.44 2      678.99
2                 16.38             16.38           1      18.89
3                 28.60             28.60           1       6.60
4                 30.22             30.22           1      25.46

  num_years_antig                    origin_up pow_max
0              3 ldkssxwpmemidmecebumciepifcamkci
               180.000
1              3 lxidpiddsbxsbosboudacockeimpuepw
                43.648
2              6 kamkkxfxxuwbdslkwifmmcsiusiuosws
                13.800
3              6 kamkkxfxxuwbdslkwifmmcsiusiuosws
                13.856
4              6 kamkkxfxxuwbdslkwifmmcsiusiuosws
                13.200
[5 rows x 27 columns]
```

8.1 Replacing the Missing data with the mean of the data

To start, we need to find the mean and replace the mean with the null values

[375]:
```
#finding the mean of the data mean_cons_12m=
merge["cons_12m"].mean() mean_cons_gas_12m=
```

```
merge["cons_gas_12m"].mean() mean_cons_last_month=
merge["cons_last_month"].mean() mean_forecast_cons_12m=
merge["forecast_cons_12m"].mean() mean_forecast_cons_year=
merge["forecast_cons_year"].mean()
mean_forecast_discount_energy=
merge["forecast_discount_energy"].mean()
mean_forecast_meter_rent_12m=
merge["forecast_meter_rent_12m"].mean()
mean_forecast_price_energy_p1=
merge["forecast_price_energy_p1"].mean()
mean_forecast_price_energy_p2=
merge["forecast_price_energy_p2"].mean()
mean_forecast_price_pow_p1=
merge["forecast_price_pow_p1"].mean() mean_imp_cons=
merge["imp_cons"].mean() mean_margin_gross_pow_ele=
merge["margin_gross_pow_ele"].mean() mean_margin_net_pow_ele=
merge["margin_net_pow_ele"].mean() mean_nb_prod_act=
merge["nb_prod_act"].mean() mean_net_margin=
merge["net_margin"].mean() mean_num_years_antig=
merge["num_years_antig"].mean() mean_pow_max=
merge["pow_max"].mean()
```

```
[376]: merge["cons_12m"] = merge["cons_12m"].fillna(mean_cons_12m)
merge["cons_gas_12m"] =
merge["cons_gas_12m"].fillna(mean_cons_gas_12m)
merge["cons_last_month"] =
merge["cons_last_month"].fillna(mean_cons_last_month)
merge["forecast_cons_12m"] = merge["forecast_cons_12m"].
↪fillna(mean_forecast_cons_12m)
merge["forecast_cons_year"] =
merge["forecast_cons_year"].
↪fillna(mean_forecast_cons_year)
merge["forecast_discount_energy"] =
merge["forecast_discount_energy"].
↪fillna(mean_forecast_discount_energy)
merge["forecast_meter_rent_12m"] =
merge["forecast_meter_rent_12m"].
↪fillna(mean_forecast_meter_rent_12m)
merge["forecast_price_energy_p1"] =
merge["forecast_price_energy_p1"].
```

```
 ↪fillna(mean_forecast_price_energy_p1)
merge["forecast_price_energy_p2"] =
merge["forecast_price_energy_p2"].
 ↪fillna(mean_forecast_price_energy_p2)
merge["forecast_price_pow_p1"] =
merge["forecast_price_pow_p1"].
 ↪fillna(mean_forecast_price_pow_p1) merge["imp_cons"]
= merge["imp_cons"].fillna(mean_imp_cons)
merge["margin_gross_pow_ele"] =
merge["margin_gross_pow_ele"].
 ↪fillna(mean_margin_gross_pow_ele)
merge["margin_net_pow_ele"] =
merge["margin_net_pow_ele"].
 ↪fillna(mean_margin_net_pow_ele) merge["nb_prod_act"] =
merge["nb_prod_act"].fillna(mean_nb_prod_act) merge["net_margin"] =
merge["net_margin"].fillna(mean_net_margin) merge["num_years_antig"]
= merge["num_years_antig"].fillna(mean_num_years_antig)
merge["pow_max"] = merge["pow_max"].fillna(mean_pow_max)
```

[377]:

```
mean_price_p1_var=
history_data["price_p1_var"].mean()
mean_price_p2_var=
history_data["price_p2_var"].mean()
mean_price_p3_var=
history_data["price_p3_var"].mean()
mean_price_p1_fix=
history_data["price_p1_fix"].mean()
mean_price_p2_fix=
history_data["price_p2_fix"].mean()
mean_price_p3_fix=
history_data["price_p3_fix"].mean()
```

[378]:

```
history_data["price_p1_var"] = history_data["price_p1_var"].
 ↪fillna(mean_price_p1_var)
history_data["price_p2_var"] =
history_data["price_p2_var"].
 ↪fillna(mean_price_p2_var)
history_data["price_p3_var"] =
history_data["price_p3_var"].
```

```
→fillna(mean_price_p3_var)
history_data["price_p1_fix"] =
history_data["price_p1_fix"].
→fillna(mean_price_p1_fix)
history_data["price_p2_fix"] =
history_data["price_p2_fix"].
→fillna(mean_price_p2_fix)
history_data["price_p3_fix"] =
history_data["price_p3_fix"].
→fillna(mean_price_p3_fix)
```

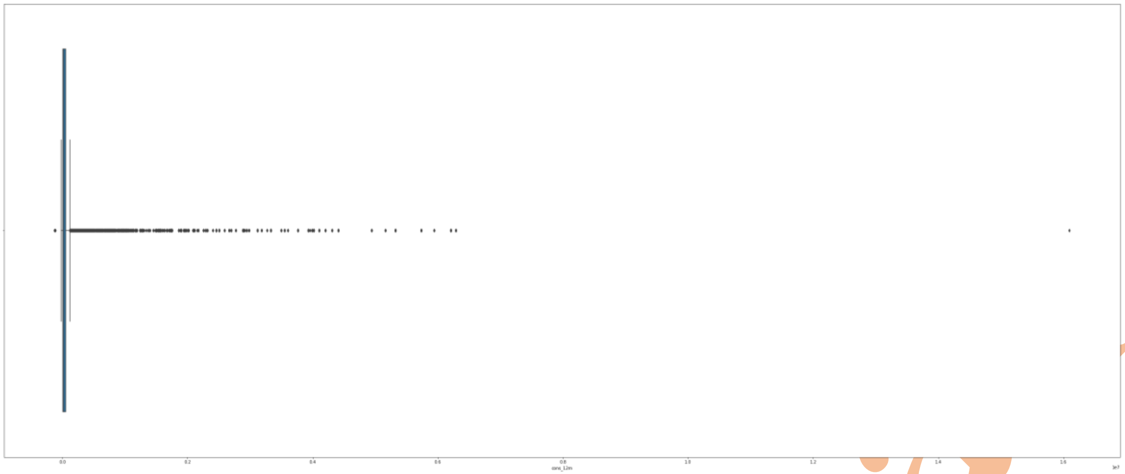
[379]:
```
merge.isnull().sum()
```

[379]:
```
id  0 churn    0 activity_new
    9545 channel_sales    4218
cons_12m   0 cons_gas_12m   0
cons_last_month 0 date_activ
    0 date_end 2
date_modif_prod 157 date_renewal
    40 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_p10 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
dtype: int64
```

9 Checking Skewness

[380]:
```
fig, axs = plt.subplots(figsize=(48,20))
sns.boxplot(x=merge['cons_12m'])
```

[380]:
```
<AxesSubplot:xlabel='cons_12m'>
```

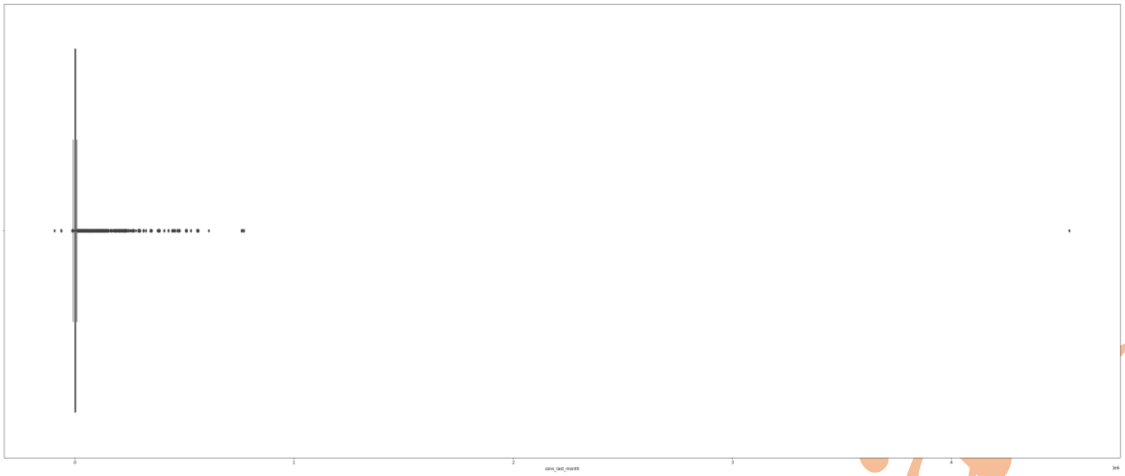


[381]:
```
fig, axs = plt.subplots(figsize=(48,20))
sns.boxplot(x=merge['cons_last_month'])
```

[381]: <AxesSubplot:xlabel='cons_last_month'>

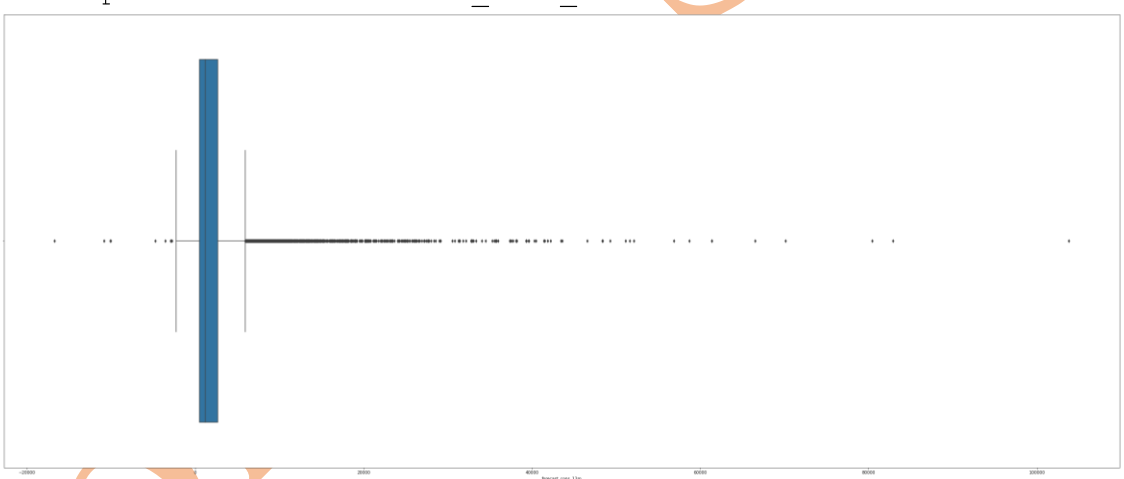
[382]:
```
fig, axs = plt.subplots(figsize=(48,20))
sns.boxplot(x=merge['cons_last_month'])
```

[382]: <AxesSubplot:xlabel='cons_last_month'>



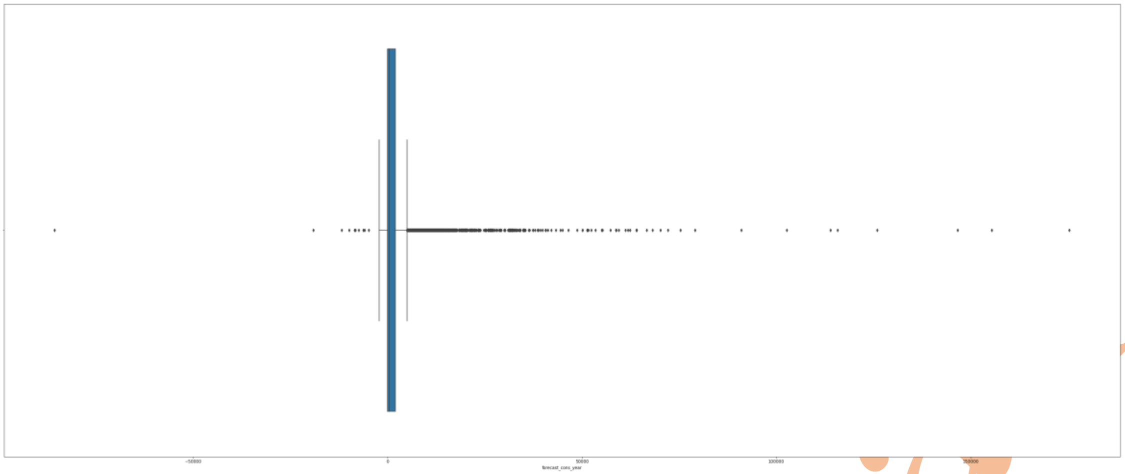
[383]:
```
fig, axs = plt.subplots(figsize=(48,20))
sns.boxplot(x=merge['forecast_cons_12m'])
```

[383]: <AxesSubplot:xlabel='forecast_cons_12m'>



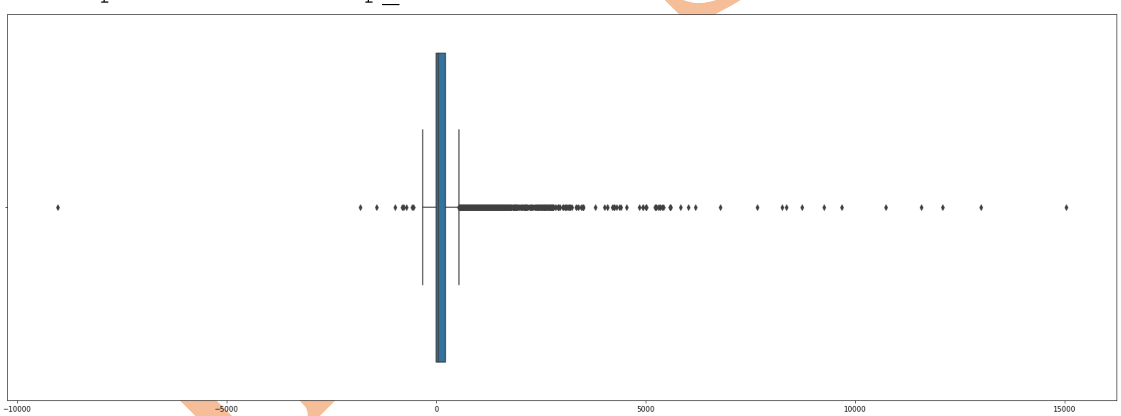
[384]:
```
fig, axs = plt.subplots(figsize=(48,20))
sns.boxplot(x=merge['forecast_cons_year'])
```

[384]: <AxesSubplot:xlabel='forecast_cons_year'>



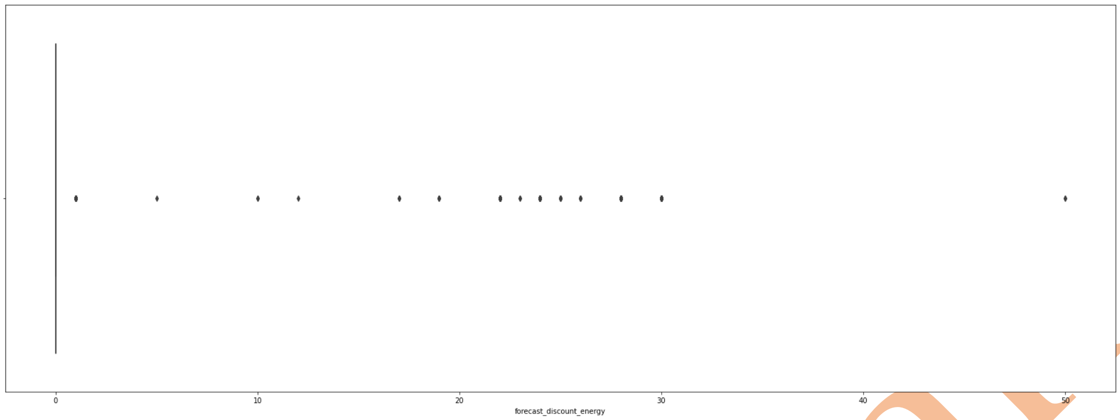
```
[385]: fig, axs = plt.subplots(figsize=(28,10))
       sns.boxplot(x=merge['imp_cons'])
```

[385]: <AxesSubplot:xlabel='imp_cons'>



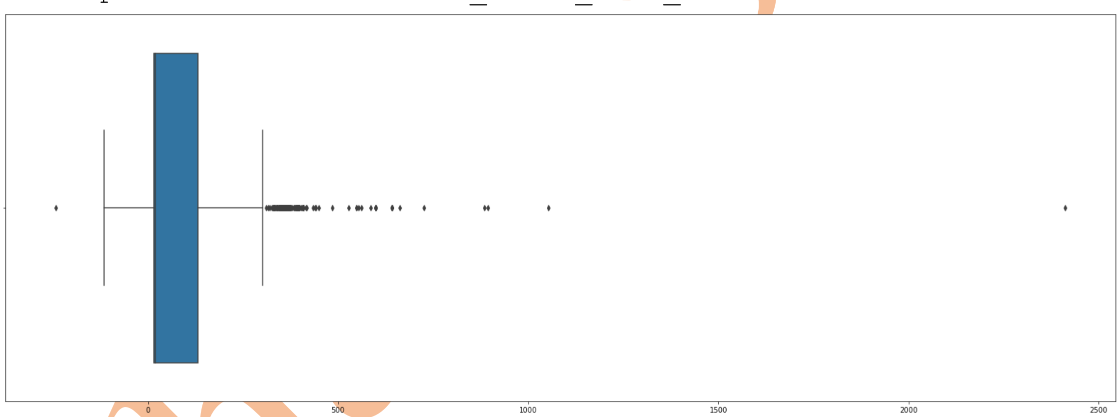

```
[386]: fig, axs = plt.subplots(figsize=(28,10))
       sns.boxplot(x=merge['forecast_discount_energy'])
```

[386]: <AxesSubplot:xlabel='forecast_discount_energy'>



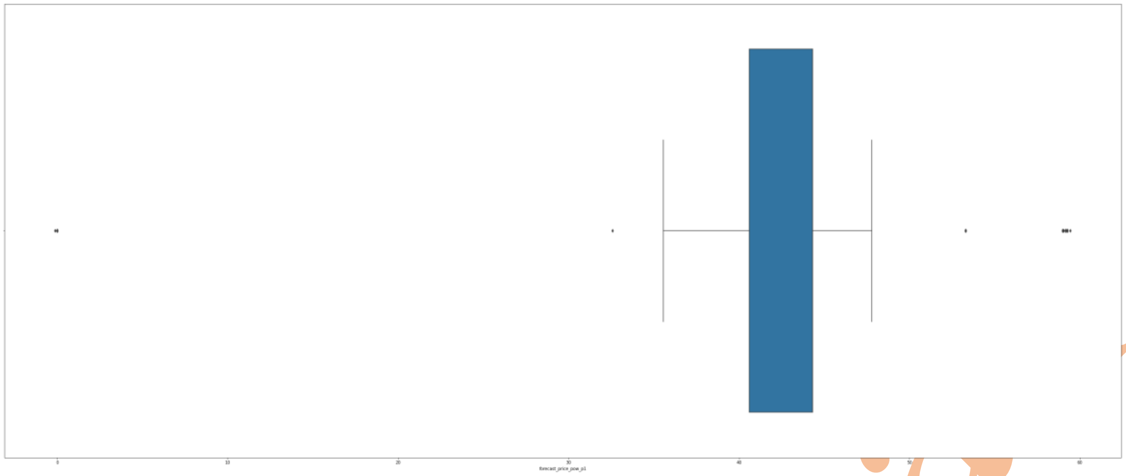
[387]:
```
fig, axs = plt.subplots(figsize=(28,10))
sns.boxplot(x=merge['forecast_meter_rent_12m'])
```

[387]: <AxesSubplot:xlabel='forecast_meter_rent_12m'>



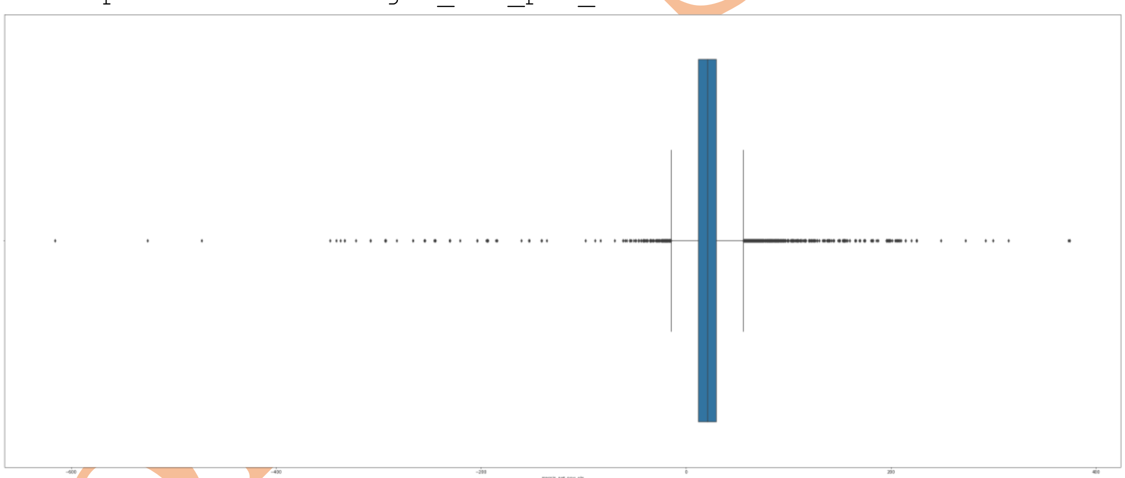
[388]:
```
fig, axs = plt.subplots(figsize=(48,20))
sns.boxplot(x=merge['forecast_price_pow_p1'])
```

[388]: <AxesSubplot:xlabel='forecast_price_pow_p1'>



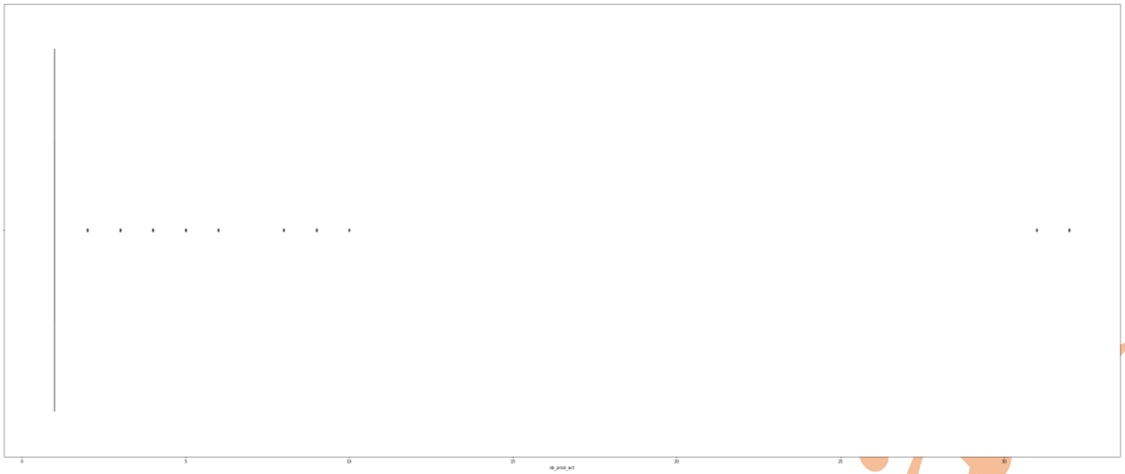
[389]:
```python
fig, axs = plt.subplots(figsize=(48,20))
sns.boxplot(x=merge['margin_net_pow_ele'])
```

[389]:
```
<AxesSubplot:xlabel='margin_net_pow_ele'>
```



[390]:
```python
fig, axs = plt.subplots(figsize=(48,20))
sns.boxplot(x=merge['nb_prod_act'])
```

[390]:
```
<AxesSubplot:xlabel='nb_prod_act'>
```

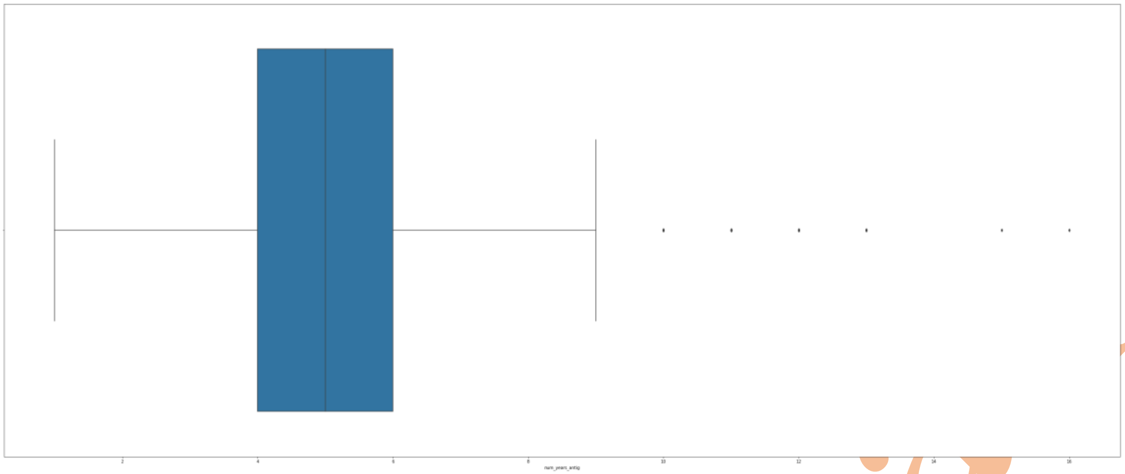


[391]:
```
fig, axs = plt.subplots(figsize=(48,20))
sns.boxplot(x=merge['net_margin'])
```

[391]: <AxesSubplot:xlabel='net_margin'>

[392]:
```
fig, axs = plt.subplots(figsize=(48,20))
sns.boxplot(x=merge['num_years_antig'])
```

[392]: <AxesSubplot:xlabel='num_years_antig'>



```
[393]: fig, axs = plt.subplots(figsize=(48,20))
       sns.boxplot(x=merge['pow_max'])
```

```
[393]: <AxesSubplot:xlabel='pow_max'>
```

10 Finding factors that affect churning

Now let's find the code of the sales channel that companies suscribed to and check to see if it has correlation with the churning.

```
[394]:
channelsales_count=pd.DataFrame(merge['channel_sales'].value_counts
()) print(channelsales_count)
```

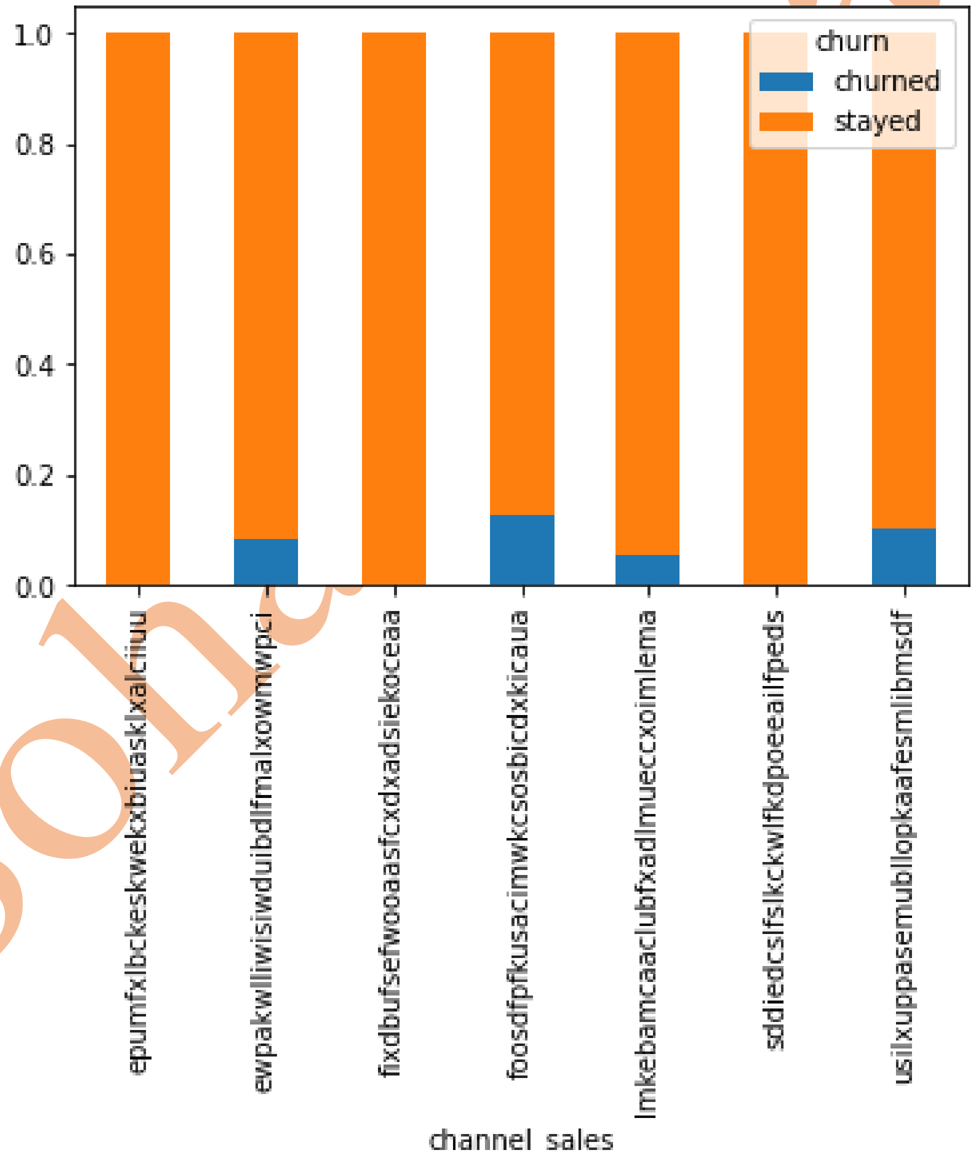
```
channel_sales
```

```
foosdfpfkusacimwkcsosbicdxkicaua      7377
  lmkebamcaaclubfxadlmueccxoimlema    2073
  usilxuppasemubllopkaafesmlibmsdf    1444
  ewpakwlliwisiwduibdlfmalxowmwpci     966
  sddiedcslfslkckwlfkdpoeeailfpeds      12
  epumfxlbckeskwekxbiuasklxalciiuu       4
  fixdbufsefwooaasfcxdxadsiekoceaa       2
```

[395]:

```
pd.crosstab(merge['channel_sales'],merge['churn'],normalize='index').plot.bar(stacked=True)
```

[395]: <AxesSubplot:xlabel='channel_sales'>

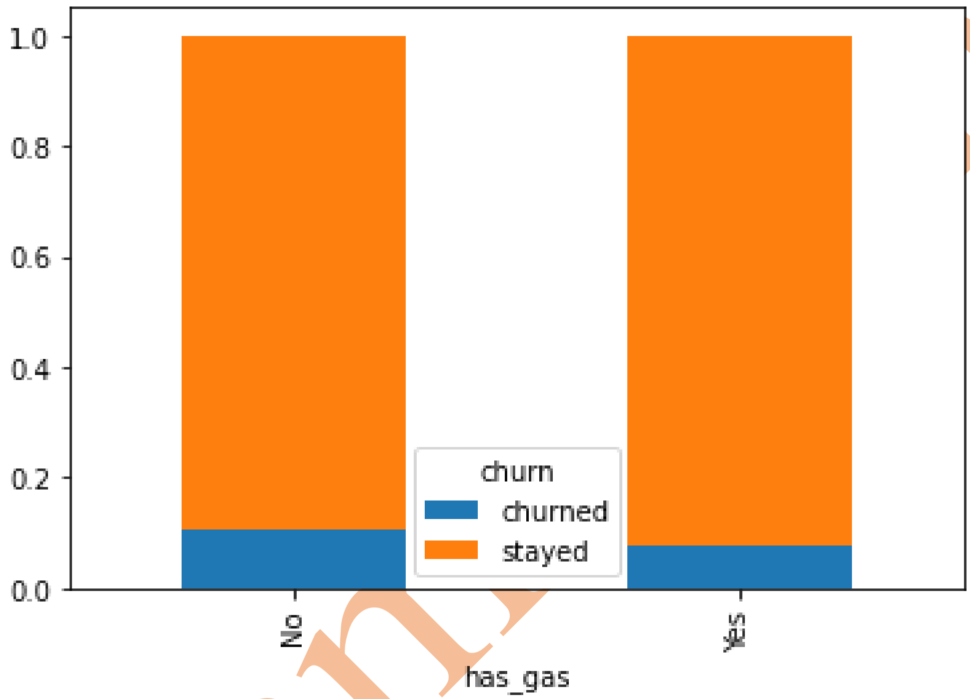


Relatively, companies with who joined the firm through ‘foosdfpfkusacimwkcsosbicdxkicaua’ and ‘usilxuppasemubllopkaafesmlibmsdf’ sales channel are more likely to churn.

Finding out if churning is dependent on whether a company has gas or not

```
[396]: pd.crosstab(merge['has_gas'],merge['churn'],normalize='index').plo
       t. ↪bar(stacked=True)
```

```
[396]: <AxesSubplot:xlabel='has_gas'>
```



It can be seen that, relatively, companies without gas churned more than the companies with gas. Finding out if churning is dependent on whether a company suscribed power or not

```
[397]: merge.groupby(['pow_max','churn']).size().unstack().head()
```

```
[397]: churn  churned stayed
       pow_max
       1.000       NaN   1.0 3.300
            2.0   5.0 3.450   2.0
            2.0 3.464  NaN   4.0
       3.500        1.0    1.0
```

```
[398]: pow_id = merge[['id','churn','pow_max']]
       pow_id.head()
```

```
[398]:                                  id   churn pow_max
        0   48ada52261e7cf58715202705a0451c9 stayed 180.000
        1   24011ae4ebbe3035111d65fa7c15bc57 churned   43.648
```

```
2    d29c2c54acc38ff3c0614d0a653813dd stayed     13.800
3    764c75f661154dac3a6c254cd082ea7d stayed     13.856
4    bba03439a292a1e166f80264c16191cb stayed     13.200
```

we need to group the pow_max into grades

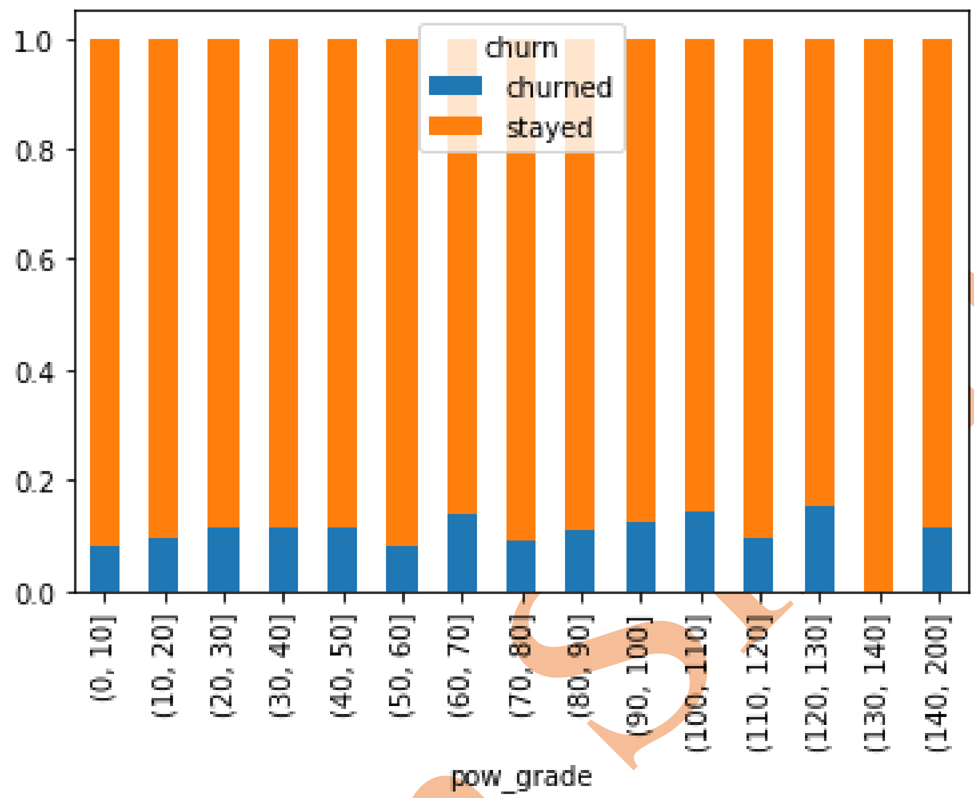
```
[399]: bins = [0,10,20,30,40,50,60,70,80,90,100,110,120,130,140,200]
```

```
[400]: pow_id['pow_grade'] = pd.cut(pow_id['pow_max'],bins, labels=None)
pow_id.head()
```

```
[400]:                                  id   churn pow_max  pow_grade
0 48ada52261e7cf58715202705a0451c9stayed 180.000 (140, 200]
  1 24011ae4ebbe3035111d65fa7c15bc57       43.648   (40,
  churned                                            50]
  2 d29c2c54acc38ff3c0614d0a653813ddstayed 13.800   (10,
                                                     20]
  3 764c75f661154dac3a6c254cd082ea7dstayed 13.856   (10,
                                                     20]
  4 bba03439a292a1e166f80264c16191cbstayed 13.200   (10,
                                                     20]
```

```
[401]: plt.figure(figsize = (40, 20))
pd.crosstab(pow_id['pow_grade'],pow_id['churn'],normalize='index').
plot. ↪bar(stacked=True)
```

```
[401]: <AxesSubplot:xlabel='pow_grade'>

<Figure size 2880x1440 with 0 Axes>
```



Relatively, companies with (60,70), (100,110) and (120,130) suscribed power are more likely to churn.

[402]:
```
merge['date_activ'] =
pd.to_datetime(merge['date_activ']).dt.to_period('m')
merge['date_end'] =
pd.to_datetime(merge['date_end']).dt.to_period('m')
merge['date_modif_prod'] =
pd.to_datetime(merge['date_modif_prod']).dt.
↪to_period('m') merge['date_renewal'] =
pd.to_datetime(merge['date_renewal']).dt.to_period('m')
```

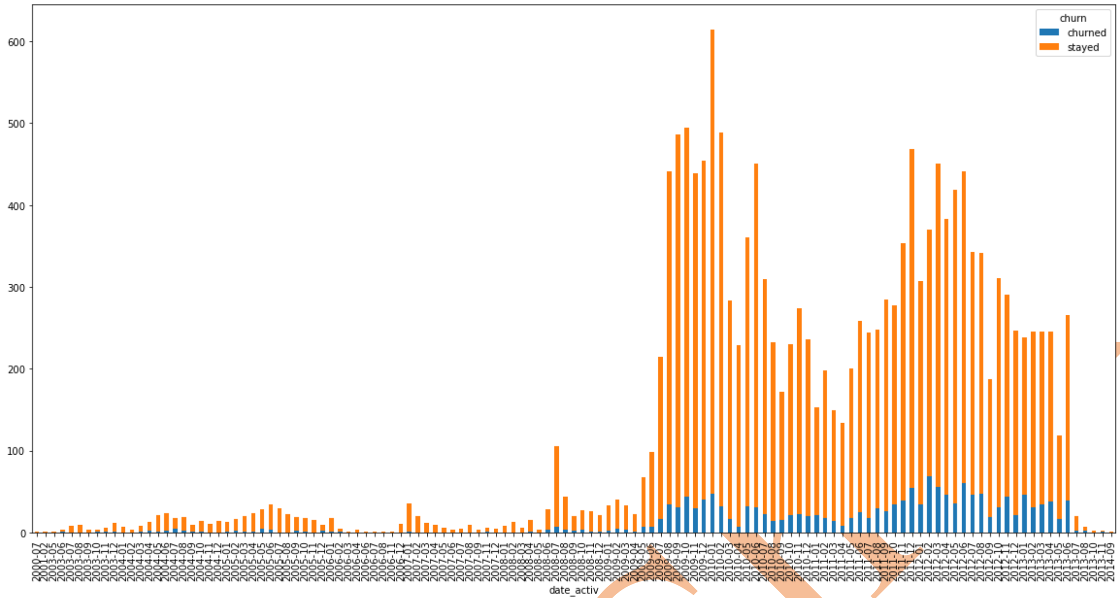
[403]:
```
merge.head()
```

[403]:
```
                                 id    churn \
0  48ada52261e7cf58715202705a0451c9   stayed
1  24011ae4ebbe3035111d65fa7c15bc57  churned
2  d29c2c54acc38ff3c0614d0a653813dd   stayed
3  764c75f661154dac3a6c254cd082ea7d   stayed
4  bba03439a292a1e166f80264c16191cb   stayed

                       activity_new                     channel_sales \
```

```
0                               esoiiifxdlbkcsluxmfuacbdckommixw
                                lmkebamcaaclubfxadlmueccxoimlema
1                               NaN foosdfpfkusacimwkcsosbicdxkicaua
2                               NaN NaN
3                               NaN foosdfpfkusacimwkcsosbicdxkicaua
4                               NaN lmkebamcaaclubfxadlmueccxoimlema

cons_12m cons_gas_12m cons_last_month date_activ date_end \
0         309275 0     10025 2012-11 2016-11
1         0 54946 0     2013-06 2016-06
2         4660    0     0     2009-08 2016-08
3         544 0 0 2010-04 2016-04 4 1584 0 0 2010-03 2016-03

  date_modif_prod … forecast_price_pow_p1 has_gas imp_cons \
0     2012-11 …  58.995952  No    831.8 1    NaT …
      40.606701  Yes   0.0
2         2009-08 …    44.311378  No    0.0
3         2010-04 …    44.311378  No    0.0
4         2010-03 …    44.311378  No    0.0

   margin_gross_pow_ele margin_net_pow_ele nb_prod_act net_margin \
0                -41.76     -41.76      1     1732.36
1                25.44 25.44 2     678.99
2                16.38 16.38 1     18.89
3                28.60 28.60 1     6.60  4     30.22 30.22 1     25.46

  num_years_antig                      origin_up pow_max
0               3 ldkssxwpmemidmecebumciepifcamkci 180.000
1               3 lxidpiddsbxsbosboudacockeimpuepw     43.648
2               6 kamkkxfxxuwbdslkwifmmcsiusiuosws     13.800
3               6 kamkkxfxxuwbdslkwifmmcsiusiuosws     13.856
4               6 kamkkxfxxuwbdslkwifmmcsiusiuosws     13.200

[5 rows x 27 columns]
```

[404]:
```python
plt.rcParams['figure.figsize']=(20,10)
merge.groupby(['date_activ','churn']).size().unstack().plot.bar(stacked=True)
```

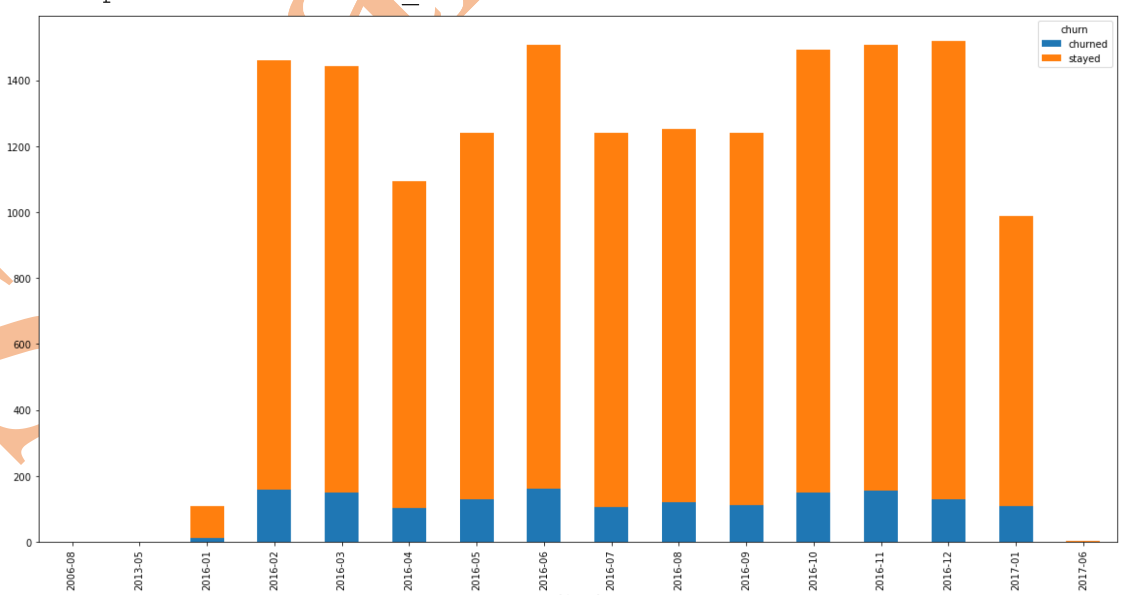
[404]: <AxesSubplot:xlabel='date_activ'>



Relatively, companies with who joined the firm through from december 2011 to 2014 are more likely to churn.

```
[405]: plt.rcParams['figure.figsize']=(20,10)
merge.groupby(['date_end','churn']).size().unstack().plot.bar(stacked=True)
```

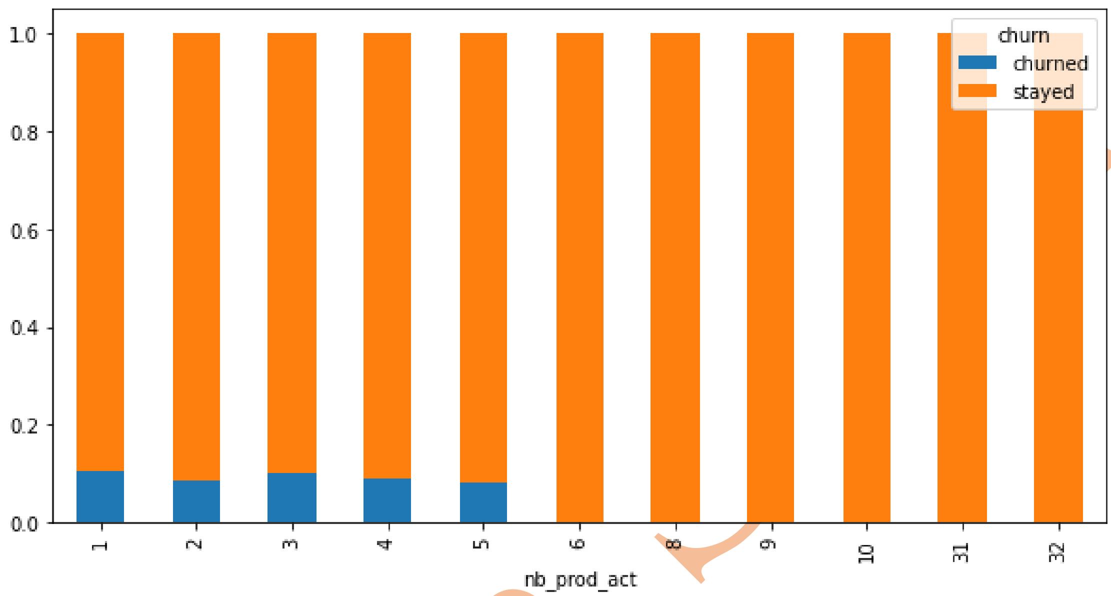
```
[405]: <AxesSubplot:xlabel='date_end'>
```



Active products and services

```
[406]: plt.rcParams['figure.figsize']=(10,5)
pd.crosstab(merge['nb_prod_act'],merge['churn'],normalize='index').
plot. ↪bar(stacked=True)
```

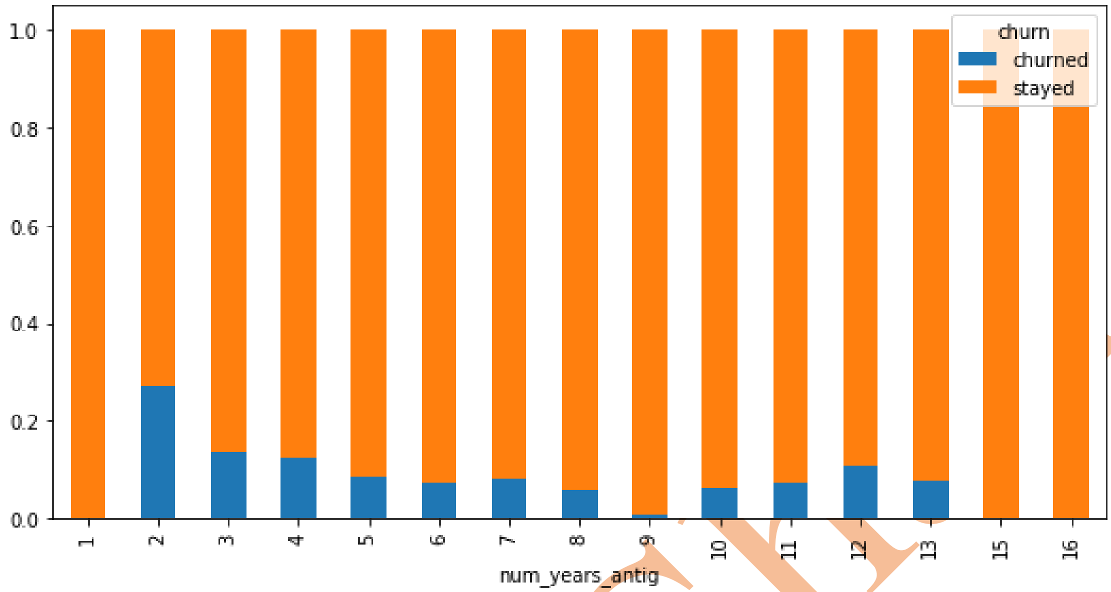
```
[406]: <AxesSubplot:xlabel='nb_prod_act'>
```



Relatively, companies with 1-5 active products and services are more likely to churn.

```
[407]: plt.rcParams['figure.figsize']=(10,5)
pd.crosstab(merge['num_years_antig'],merge['churn'],normalize='inde
x').plot. ↪bar(stacked=True)
```

```
[407]: <AxesSubplot:xlabel='num_years_antig'>
```

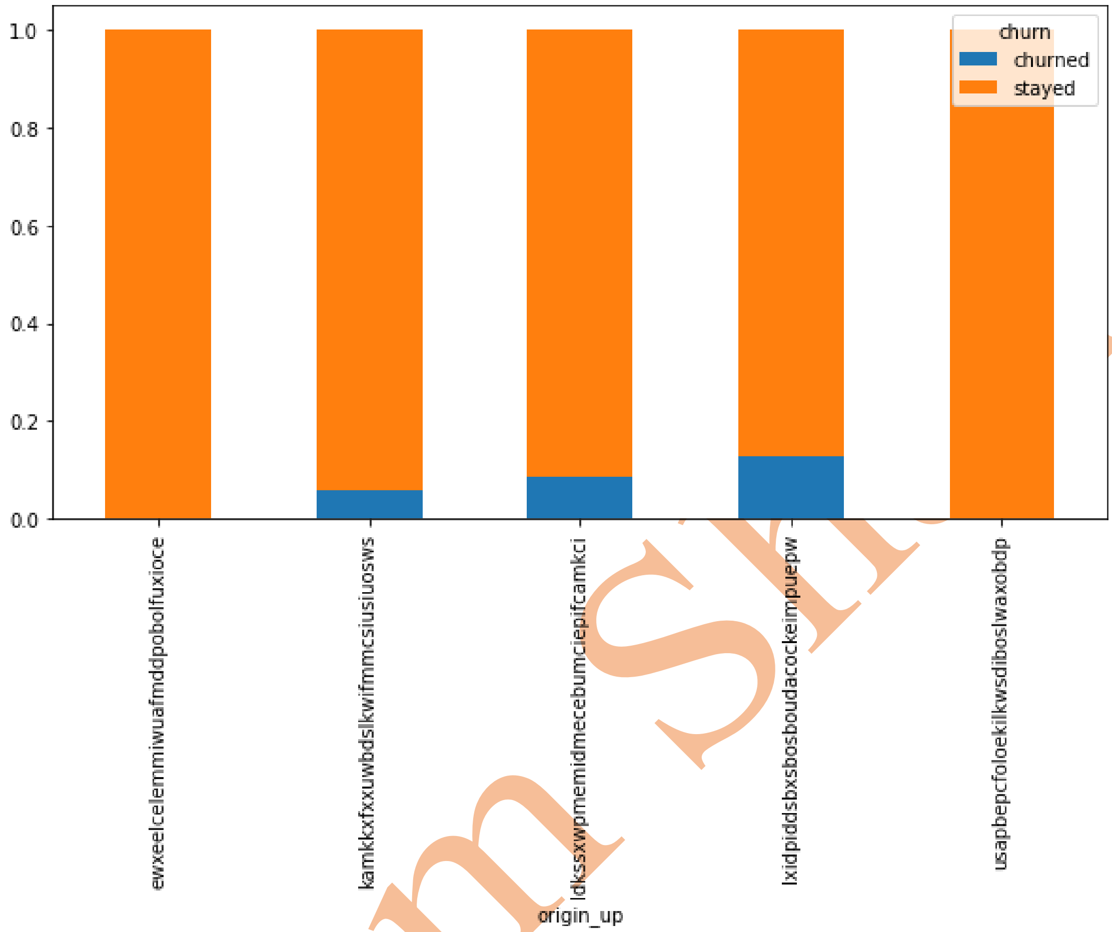


Relatively, companies are likely to churn in the second year. After, the probability for companies to churn diminishes up to the ninth year and starts rising again. By the 15th year, the companies are more likely to stay.

Code of the electricity campaign the customer first subscribed to

```
[408]: plt.rcParams['figure.figsize']=(10,5)
pd.crosstab(merge['origin_up'],merge['churn'],normalize='index').pl
ot. ↪bar(stacked=True)
```

```
[408]: <AxesSubplot:xlabel='origin_up'>
```

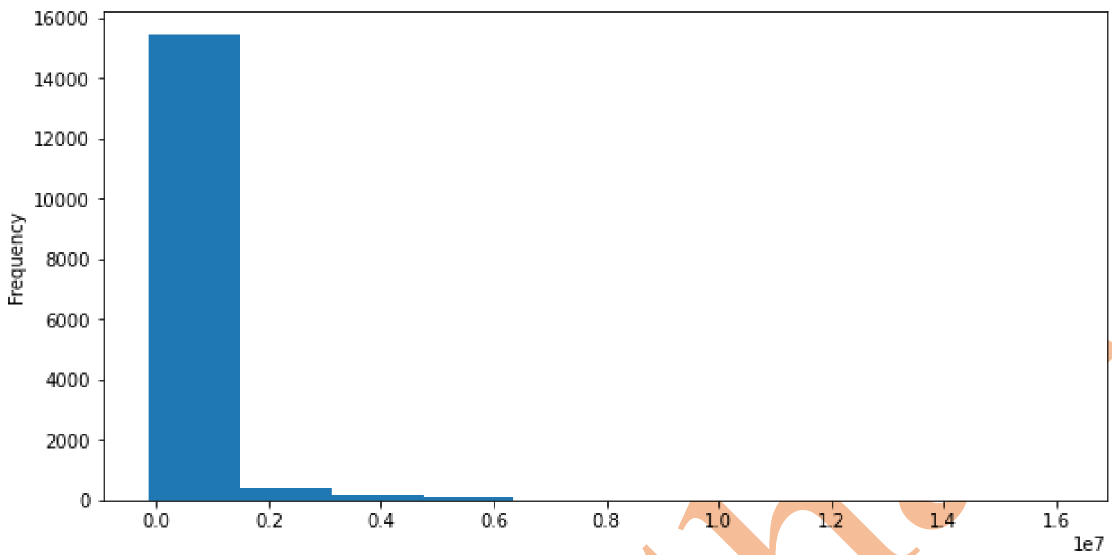


Relatively, companies that first suscribed to the code of the electricity campaign 'lxidpiddsbx' is likely to churn.

11 Histogram of the data

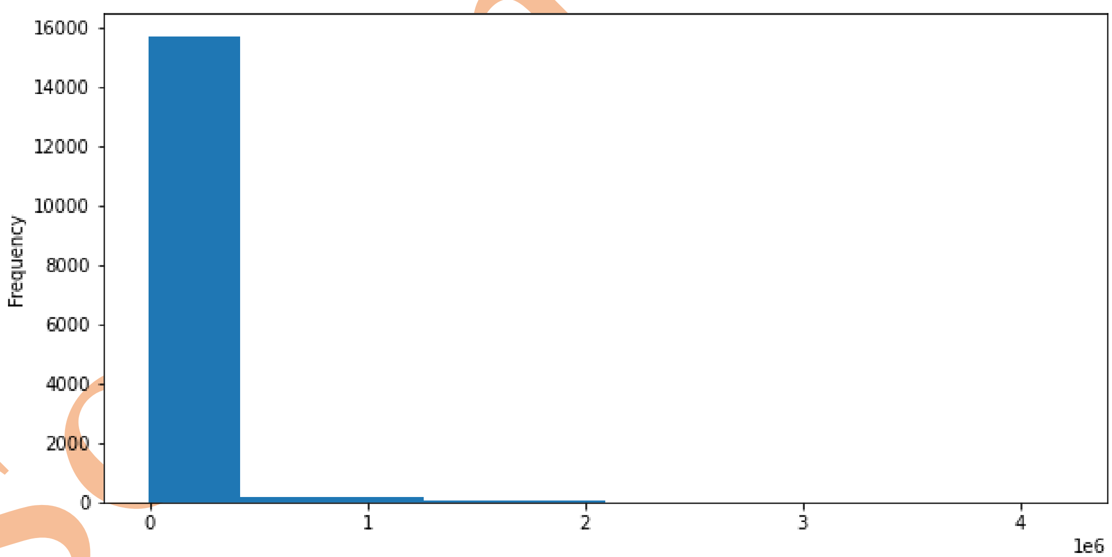
```
[409]: merge["cons_12m"].plot.hist()
```

```
[409]: <AxesSubplot:ylabel='Frequency'>
```



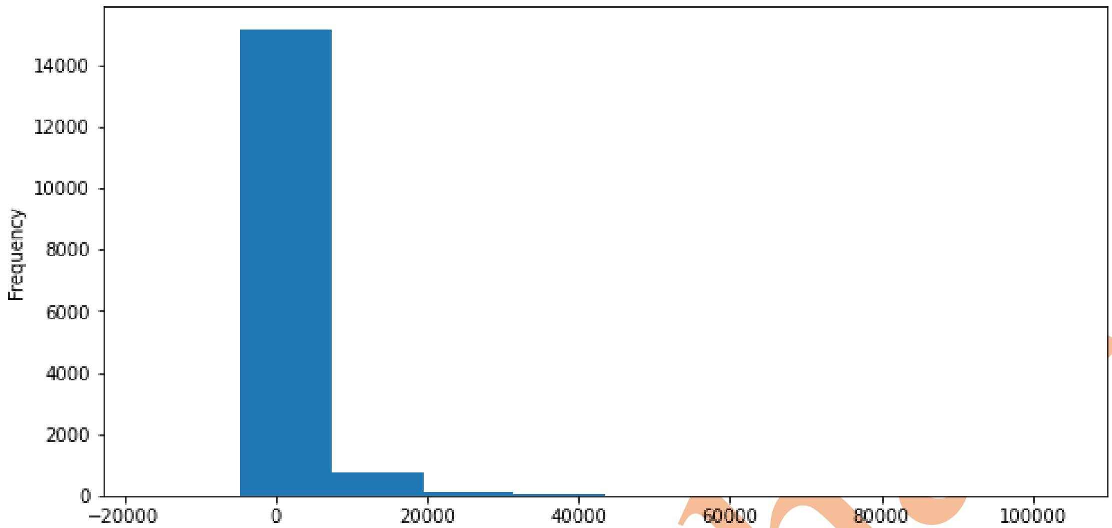

```
[410]: merge["cons_gas_12m"].plot.hist()
```

[410]: <AxesSubplot:ylabel='Frequency'>



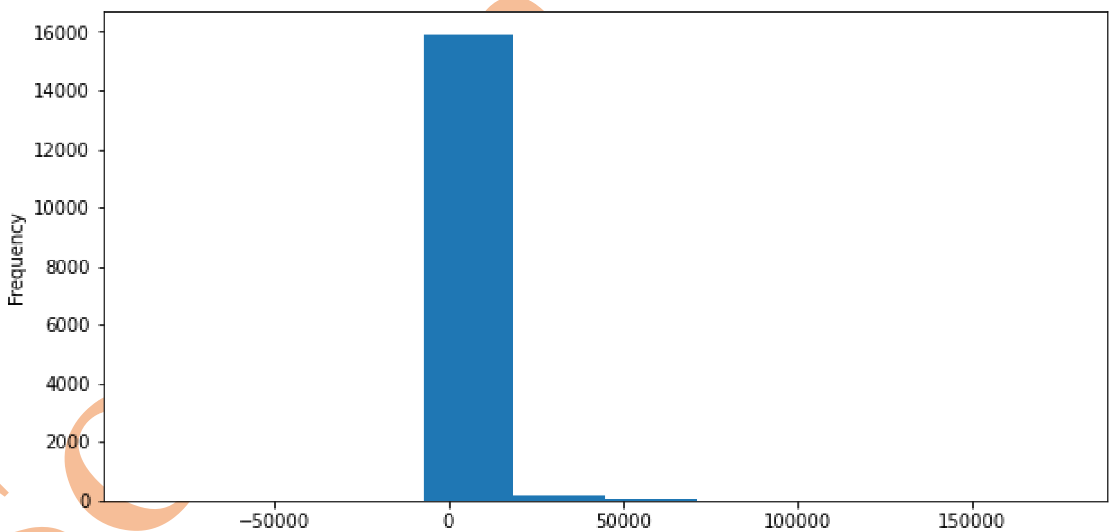

```
[411]: merge["forecast_cons_12m"].plot.hist()
```

[411]: <AxesSubplot:ylabel='Frequency'>



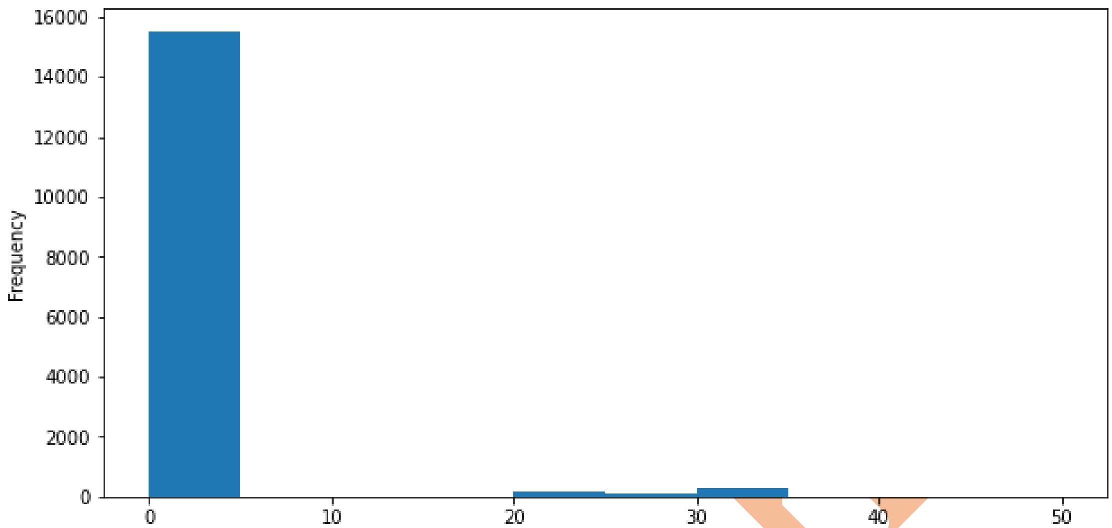

```
[412]: merge["forecast_cons_year"].plot.hist()
```

[412]: <AxesSubplot:ylabel='Frequency'>



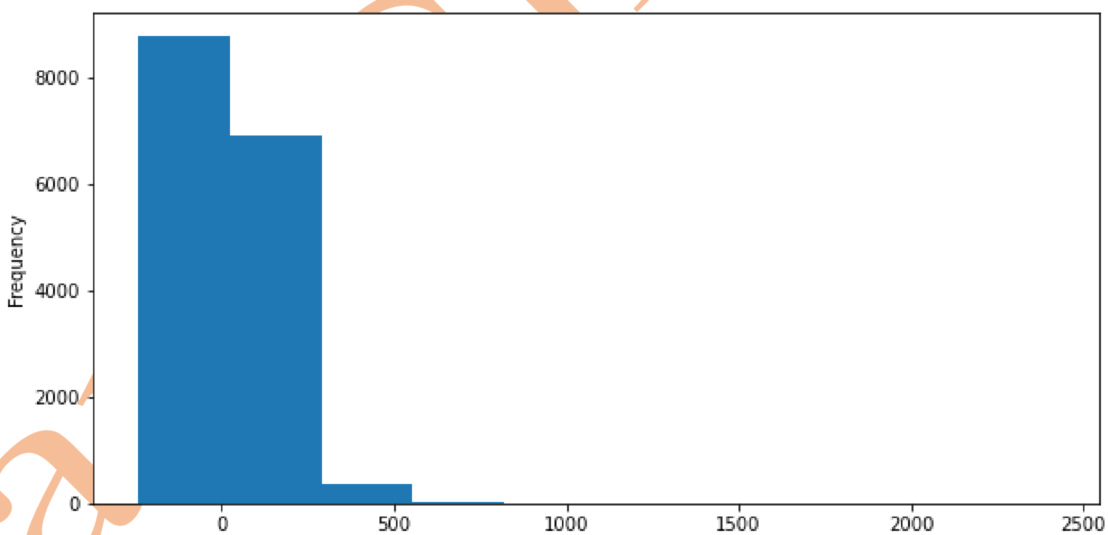

```
[413]: merge["forecast_discount_energy"].plot.hist()
```

[413]: <AxesSubplot:ylabel='Frequency'>



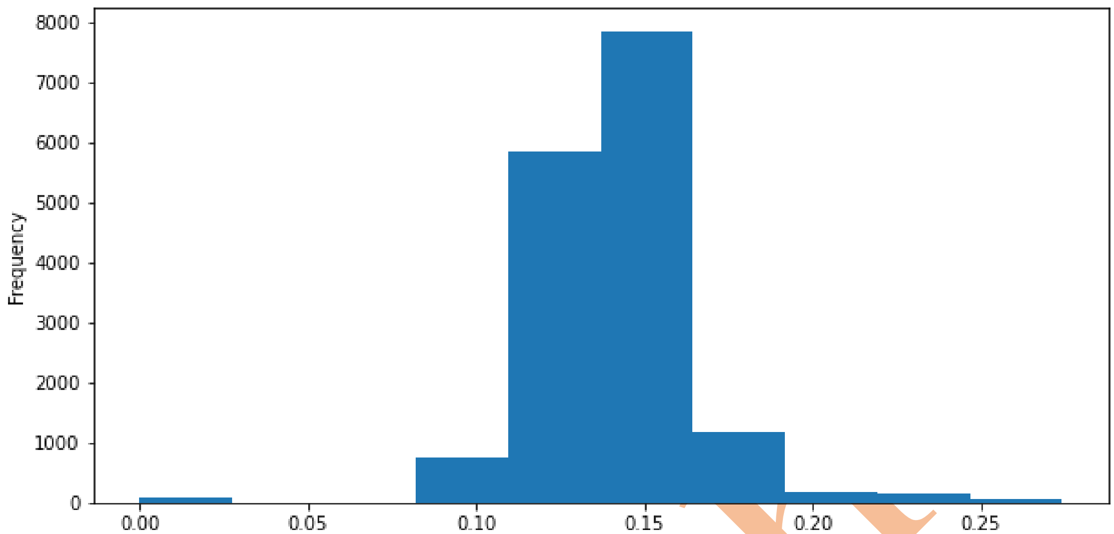
[414]:
```
merge["forecast_meter_rent_12m"].plot.hist()
```

[414]: <AxesSubplot:ylabel='Frequency'>



[415]:
```
merge["forecast_price_energy_p1"].plot.hist()
```

<AxesSubplot:ylabel='Frequency'>

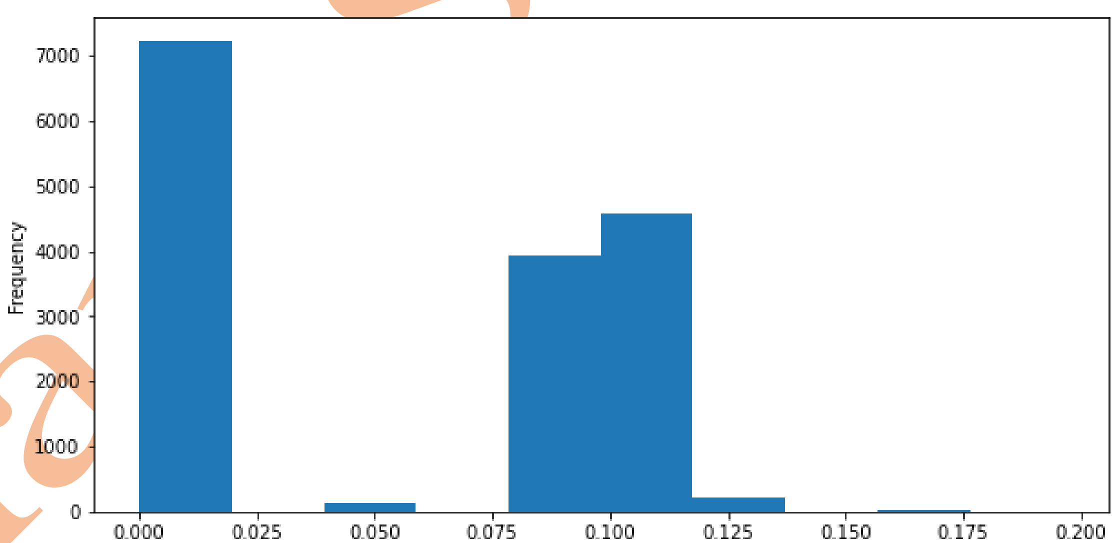


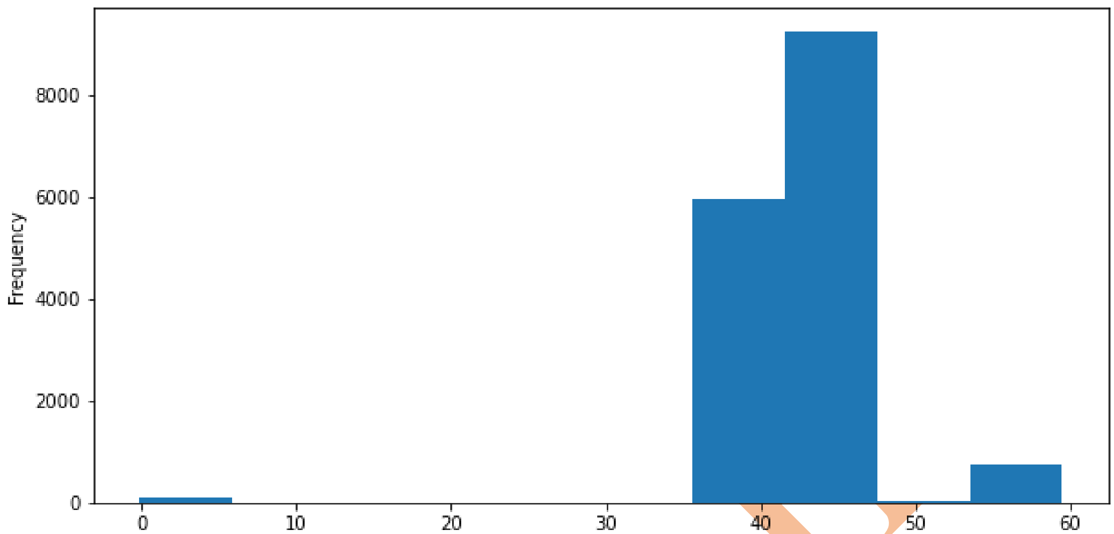
[415]:

[416]:

```
merge["forecast_price_energy_p2"].plot.hist()
```

[416]: <AxesSubplot:ylabel='Frequency'>



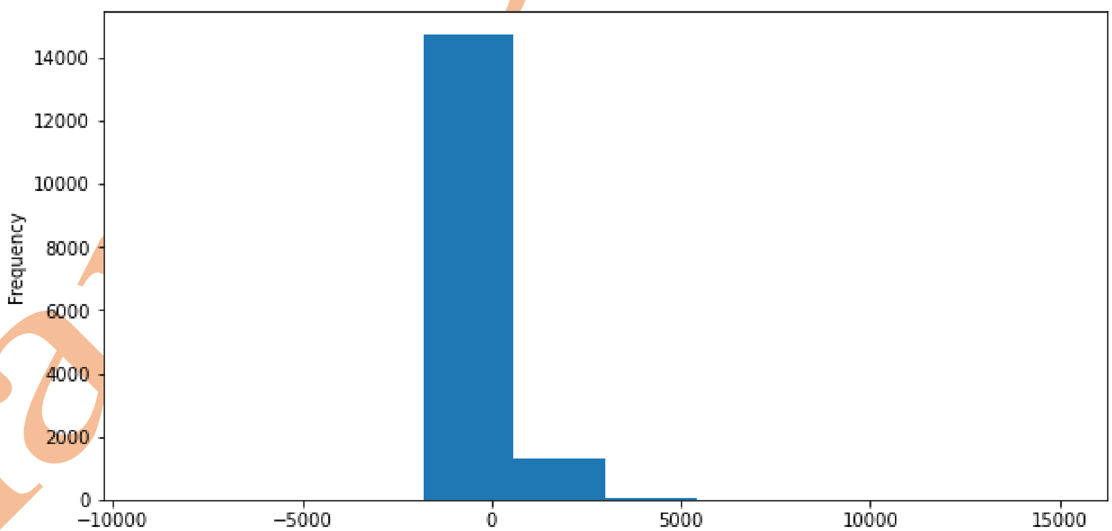



```
[417]: merge["forecast_price_pow_p1"].plot.hist()
```

[417]: <AxesSubplot:ylabel='Frequency'>

```
[418]: merge["imp_cons"].plot.hist()
```

[418]: <AxesSubplot:ylabel='Frequency'>

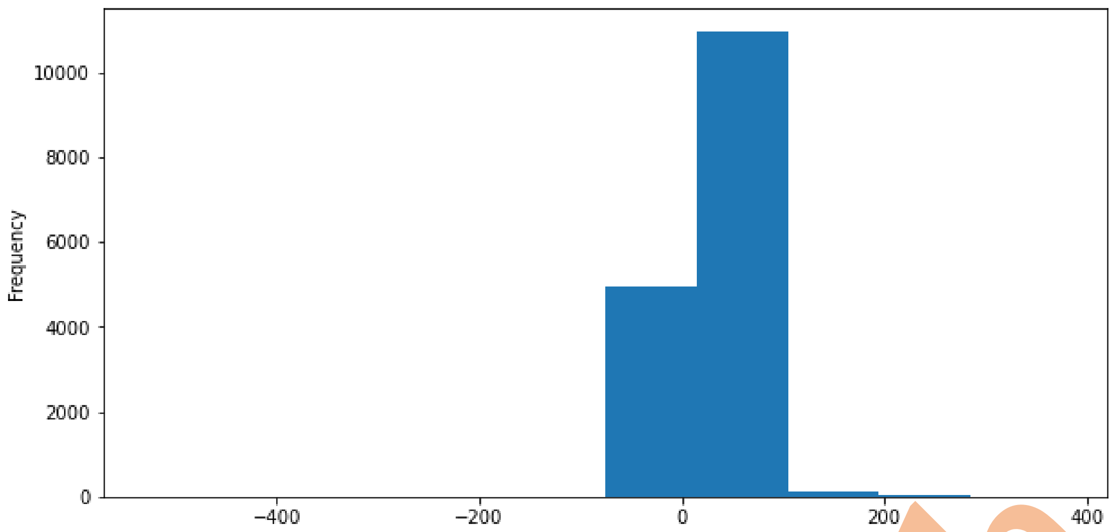


<AxesSubplot:ylabel='Frequency'>

[419]:

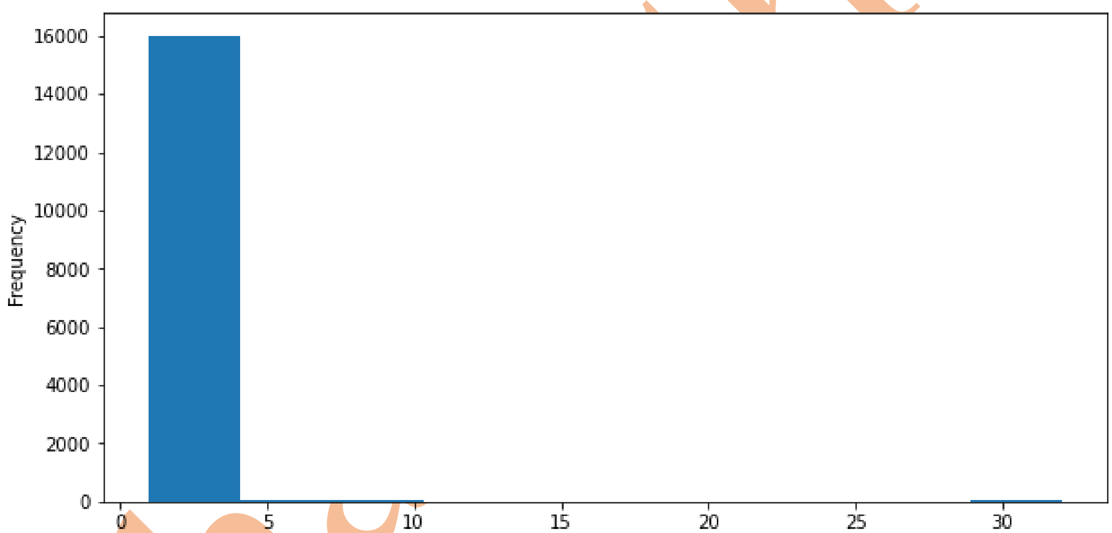
```
merge["margin_gross_pow_ele"].plot.hist()
```

[419]:



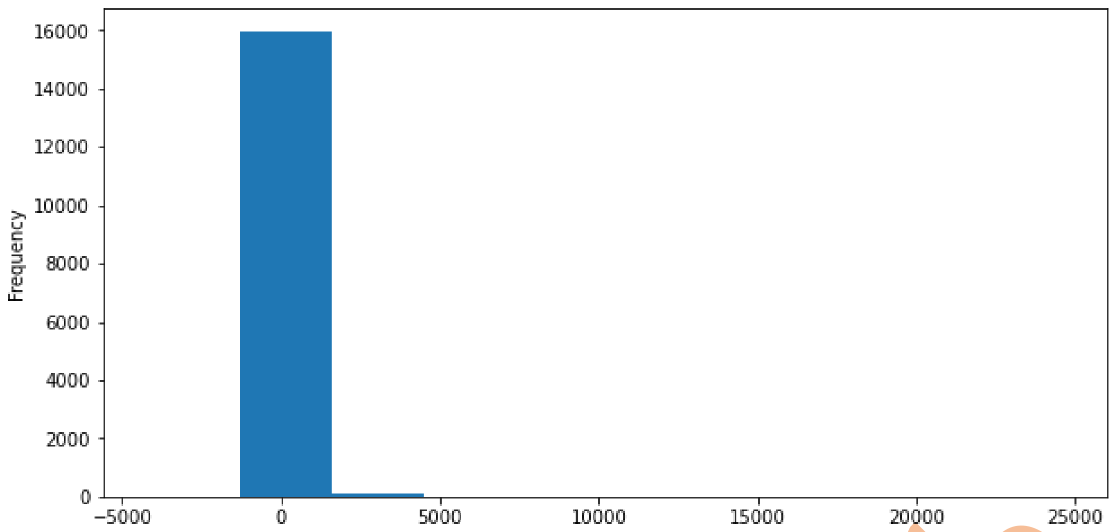
[420]:
```
merge["nb_prod_act"].plot.hist()
```

[420]: <AxesSubplot:ylabel='Frequency'>



[421]:
```
merge["net_margin"].plot.hist()
```

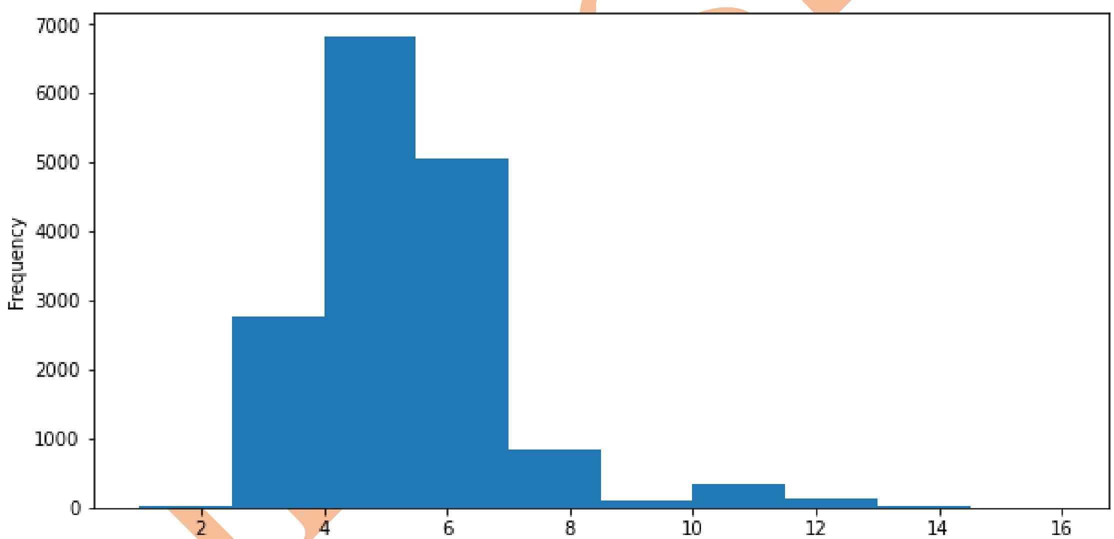
<AxesSubplot:ylabel='Frequency'>

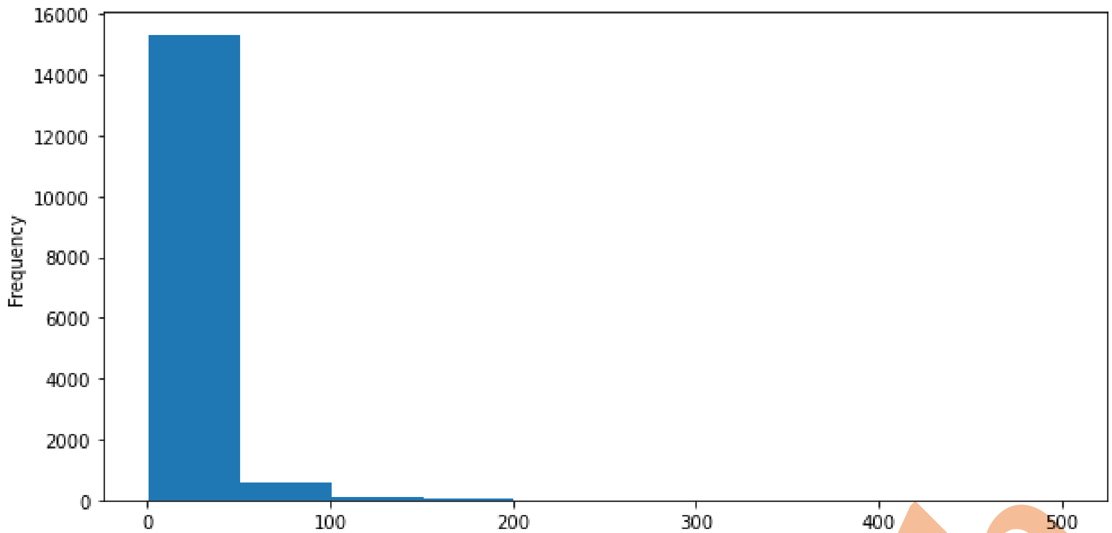


[421]:

[422]:
```
merge["num_years_antig"].plot.hist()
```

[422]: <AxesSubplot:ylabel='Frequency'>






```
[423]: merge["pow_max"].plot.hist()
```

[423]:

It can be seen that most of the data are rightly skewed

11.1 Checking the company with the highest consumption

```
[424]: consumption = merge[["id","cons_12m",
 "cons_gas_12m","cons_last_month",␣ ↪"imp_cons", "has_gas", "churn"]]
```

```
[425]: total_cons_12m = pd.DataFrame(consumption.groupby(["id",
"churn"])["cons_12m"].
 ↪agg(["sum"]))
 total_cons_12m.sort_values(ascending=False,
 by="sum").head()
```

```
[425]:sum id     churn
 2c2abbe8998364dd500e41588d41f45f     stayed
 16097108   b880901f75613c801886354abf24f30a
 stayed                                6286272
 3cbf266f90f0419636aa9e748fa0e7f0     stayed
 <AxesSubplot:ylabel='Frequency'>
```

```
6286272    f3baf732b3a86a45f5aec2d4578070c0
stayed                              6286272
4130bb214991c2ec4504b96d527624ca     stayed
6286272
```

It can be seen that ,company ‘2c2abbe8998364dd500e41588d41f45f’ has the highest consumption of energy.

[426]:
```
merge.to_csv('merge.csv')
```