BCG Feature Engineering task 3

July 19, 2021

0.0.1 Task 3

1 Feature Engineering

Uncovering signals within data

2 Import Relevant libraries

[1]:
```
import pandas as pd
import seaborn as sns
sns.set()
import datetime
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import os
```

[2]:
```
## setting max columnss to display
pd.set_option('display.max_columns', 120)
```

import new processed csv

3 Import Data

[3]:
```
main = pd.read_csv("BCG_main_01.csv")
train = pd.read_csv("BCG_train_01.csv")
```

[4]:
```
main.head(10)
```

[4]:
```
                                 id  price_date  energy_1st_per_pri  \
0  038af19179925da21a25619c5a24b745  2015-01-01            0.151367
1  038af19179925da21a25619c5a24b745  2015-02-01            0.151367
2  038af19179925da21a25619c5a24b745  2015-03-01            0.151367
3  038af19179925da21a25619c5a24b745  2015-04-01            0.149626
4  038af19179925da21a25619c5a24b745  2015-05-01            0.149626
5  038af19179925da21a25619c5a24b745  2015-06-01            0.149626
6  038af19179925da21a25619c5a24b745  2015-07-01            0.150321
```

```
7  038af19179925da21a25619c5a24b745  2015-08-01           0.145859
8  038af19179925da21a25619c5a24b745  2015-09-01           0.145859
9  038af19179925da21a25619c5a24b745  2015-10-01           0.145859

   energy_2nd_per_pri  energy_3rd_per_pri  power_1st_per_pri  \
0                 0.0                 0.0          44.266931
1                 0.0                 0.0          44.266931
2                 0.0                 0.0          44.266931
3                 0.0                 0.0          44.266931
4                 0.0                 0.0          44.266931
5                 0.0                 0.0          44.266930
6                 0.0                 0.0          44.444710
7                 0.0                 0.0          44.444710
8                 0.0                 0.0          44.444710
9                 0.0                 0.0          44.444710

   power_2nd_per_pri  power_3rd_per_pri  churn
0                0.0                0.0      0
1                0.0                0.0      0
2                0.0                0.0      0
3                0.0                0.0      0
4                0.0                0.0      0
5                0.0                0.0      0
6                0.0                0.0      0
7                0.0                0.0      0
8                0.0                0.0      0
9                0.0                0.0      0
```

[5]:
```
train.head(10)
```

[5]:
```
                                 id                      activity_new  \
0  48ada52261e7cf58715202705a0451c9  esoiiifxdlbkcsluxmfuacbdckommixw
1  d29c2c54acc38ff3c0614d0a653813dd                               NaN
2  764c75f661154dac3a6c254cd082ea7d                               NaN
3  bba03439a292a1e166f80264c16191cb                               NaN
4  568bb38a1afd7c0fc49c77b3789b59a3  sfisfxfcocfpcmckuekokxuseixdaoeu
5  149d57cf92fc41cf94415803a877cb4b                               NaN
6  1aa498825382410b098937d65c4ec26d                               NaN
7  7ab4bf4878d8f7661dfc20e9b8e18011  sscfoipxikopfskekuobeuxkxmwsuucb
8  01495c955be7ec5e7f3203406785aae0                               NaN
9  f53a254b1115634330c12c7fdbf7958a  cssldxpacdmuuaulamxdekcokibauube

   campaign_disc_ele  cons_12m  cons_gas_12m  cons_last_month  date_activ  \
0                NaN    309275             0            10025  2012-11-07
1                NaN      4660             0                0  2009-08-21
2                NaN       544             0                0  2010-04-16
3                NaN      1584             0                0  2010-03-30
```

```
4             NaN    121335             0             12400  2010-04-08
5             NaN      4425             0               526  2010-01-13
6             NaN      8302             0              1998  2011-12-09
7             NaN     45097             0                 0  2011-12-02
8             NaN     29552             0              1260  2010-04-21
9             NaN      2962             0                 0  2011-09-23

     date_end date_modif_prod date_renewal  forecast_cons_12m  \
0  2016-11-06      2012-11-07   2015-11-09           26520.30
1  2016-08-30      2009-08-21   2015-08-31             189.95
2  2016-04-16      2010-04-16   2015-04-17              47.96
3  2016-03-30      2010-03-30   2015-03-31             240.04
4  2016-04-08      2010-04-08   2015-04-12           10865.02
5  2016-03-07      2010-01-13   2015-03-09             445.75
6  2016-12-09      2015-11-01   2015-12-10             796.94
7  2016-12-02      2011-12-02   2015-12-03            8069.28
8  2016-04-21      2010-04-21   2015-04-22             864.73
9  2016-09-23      2011-09-23   2015-09-25             444.38

   forecast_cons_year  forecast_discount_energy  forecast_meter_rent_12m  \
0               10025                       0.0                   359.29
1                   0                       0.0                    16.27
2                   0                       0.0                    38.72
3                   0                       0.0                    19.83
4               12400                       0.0                   170.74
5                 526                       0.0                   131.73
6                1998                       0.0                    30.12
7                   0                       0.0                     0.00
8                 751                       0.0                   144.49
9                   0                       0.0                    15.85

   forecast_price_energy_p1  forecast_price_energy_p2  forecast_price_pow_p1  \
0                  0.095919                  0.088347              58.995952
1                  0.145711                  0.000000              44.311378
2                  0.165794                  0.087899              44.311378
3                  0.146694                  0.000000              44.311378
4                  0.110083                  0.093746              40.606701
5                  0.116900                  0.100015              40.606701
6                  0.164775                  0.086131              45.308378
7                  0.166178                  0.087538              44.311378
8                  0.115174                  0.098837              40.606701
9                  0.145711                  0.000000              44.311378

  has_gas  imp_cons  margin_gross_pow_ele  margin_net_pow_ele  nb_prod_act  \
0       f    831.80                 41.76               41.76            1
1       f      0.00                 16.38               16.38            1
2       f      0.00                 28.60               28.60            1
```

```
3       f     0.00                 30.22                30.22            1
4       f  1052.37                  3.18                 3.18            1
5       f    52.32                 44.91                44.91            1
6       f   181.21                 33.12                33.12            1
7       f     0.00                  4.04                 4.04            1
8       f    70.63                 53.92                53.92            1
9       f     0.00                 12.82                12.82            1

   net_margin  num_years_antig                       origin_up  pow_max  \
0     1732.36                3  ldkssxwpmemidmecebumciepifcamkci  180.000
1       18.89                6  kamkkxfxxuwbdslkwifmmcsiusiuosws   13.800
2        6.60                6  kamkkxfxxuwbdslkwifmmcsiusiuosws   13.856
3       25.46                6  kamkkxfxxuwbdslkwifmmcsiusiuosws   13.200
4      823.18                6  lxidpiddsbxsbosboudacockeimpuepw   75.000
5       47.98                6  kamkkxfxxuwbdslkwifmmcsiusiuosws   19.800
6      118.89                4  lxidpiddsbxsbosboudacockeimpuepw   13.200
7      346.63                4  lxidpiddsbxsbosboudacockeimpuepw   15.000
8      100.09                6  lxidpiddsbxsbosboudacockeimpuepw   26.400
9       42.59                4  kamkkxfxxuwbdslkwifmmcsiusiuosws   13.200

   churn
0      0
1      0
2      0
3      0
4      0
5      0
6      1
7      1
8      0
9      0
```

4 Feature Engineering

Since we have the consumption data for each of companies for year 2015,2016 We will create new features using the average of the year, the last six months and last three months

[6]:
```
mean_year = main.groupby(["id"]).mean().reset_index()
# df_group2 = df.groupby("ID", as_index=False).sum()
# mean_year = main.groupby(["id"], as_index = False).mean()
```

[7]:
```
mean_year.head(10)
```

[7]:
```
                                 id  energy_1st_per_pri  energy_2nd_per_pri  \
0  0002203ffbb812588b632b9e628cc38d            0.124338            0.103794
1  0004351ebdd665e6ee664792efc4fd13            0.146426            0.000000
2  0010bcc39e42b3c2131ed2ce55246e3c            0.181558            0.000000
```

```
3  0010ee3855fdea87602a5b7aba8e42de          0.118757          0.098292
4  00114d74e963e47177db89bc70108537          0.147926          0.000000
5  00126c87cf78d7604278f0a9adeb689e          0.119806          0.099417
6  0013f326a839a2f6ad87a1859952d227          0.126076          0.105542
7  00184e957277eeef733a7b563fdabd06          0.147637          0.000000
8  001987ed9dbdab4efa274a9c7233e1f4          0.122756          0.102290
9  0019baf3ed1242cd99b3cb592030446f          0.267449          0.000000

   energy_3rd_per_pri  power_1st_per_pri  power_2nd_per_pri  \
0            0.073160          40.701732          24.421038
1            0.000000          44.385450           0.000000
2            0.000000          45.319710           0.000000
3            0.069032          40.647427          24.388455
4            0.000000          44.266930           0.000000
5            0.070304          40.661003          24.396601
6            0.074921          40.728885          24.437330
7            0.000000          44.266930           0.000000
8            0.073031          40.647427          24.388455
9            0.000000          57.961930           0.000000

   power_3rd_per_pri  churn
0          16.280694      0
1           0.000000      0
2           0.000000      0
3          16.258971      0
4           0.000000      0
5          16.264402      0
6          16.291555      0
7           0.000000      0
8          16.258972      0
9           0.000000      1
```

[8]:

```python
mean_6m = main[main["price_date"] > "2015-06-01"].groupby(["id"]).mean().
    ↪reset_index()
```

[9]:

```python
mean_6m.head(10)
```

[9]:

```
                                 id  energy_1st_per_pri  energy_2nd_per_pri  \
0  0002203ffbb812588b632b9e628cc38d          0.121266          0.102368
1  0004351ebdd665e6ee664792efc4fd13          0.144687          0.000000
2  0010bcc39e42b3c2131ed2ce55246e3c          0.202024          0.000000
3  0010ee3855fdea87602a5b7aba8e42de          0.114428          0.096080
4  00114d74e963e47177db89bc70108537          0.146184          0.000000
5  00126c87cf78d7604278f0a9adeb689e          0.114428          0.096080
6  0013f326a839a2f6ad87a1859952d227          0.123007          0.104108
7  00184e957277eeef733a7b563fdabd06          0.145837          0.000000
8  001987ed9dbdab4efa274a9c7233e1f4          0.119535          0.101186
```

```
9  0019baf3ed1242cd99b3cb592030446f          0.275868           0.000000

   energy_3rd_per_pri  power_1st_per_pri  power_2nd_per_pri  \
0            0.073728          40.728885           24.43733
1            0.000000          44.444710            0.00000
2            0.000000          45.944710            0.00000
3            0.069418          40.728885           24.43733
4            0.000000          44.266930            0.00000
5            0.069418          40.728885           24.43733
6            0.075469          40.728885           24.43733
7            0.000000          44.266930            0.00000
8            0.074525          40.728885           24.43733
9            0.000000          59.206930            0.00000

   power_3rd_per_pri  churn
0          16.291555      0
1           0.000000      0
2           0.000000      0
3          16.291555      0
4           0.000000      0
5          16.291555      0
6          16.291555      0
7           0.000000      0
8          16.291555      0
9           0.000000      1
```

[10]:
```
mean_3m = main[main["price_date"] > "2015-10-01"].groupby(["id"]).mean().
↪reset_index()
```

[11]:
```
mean_3m.head(10)
```

[11]:
```
                                 id  energy_1st_per_pri  energy_2nd_per_pri  \
0  0002203ffbb812588b632b9e628cc38d            0.119906            0.101673
1  0004351ebdd665e6ee664792efc4fd13            0.143943            0.000000
2  0010bcc39e42b3c2131ed2ce55246e3c            0.201280            0.000000
3  0010ee3855fdea87602a5b7aba8e42de            0.113068            0.095385
4  00114d74e963e47177db89bc70108537            0.145440            0.000000
5  00126c87cf78d7604278f0a9adeb689e            0.113068            0.095385
6  0013f326a839a2f6ad87a1859952d227            0.121647            0.103413
7  00184e957277eeef733a7b563fdabd06            0.145093            0.000000
8  001987ed9dbdab4efa274a9c7233e1f4            0.118175            0.100491
9  0019baf3ed1242cd99b3cb592030446f            0.275124            0.000000

   energy_3rd_per_pri  power_1st_per_pri  power_2nd_per_pri  \
0            0.073719          40.728885           24.43733
1            0.000000          44.444710            0.00000
2            0.000000          45.944710            0.00000
```

```
3            0.069409         40.728885          24.43733
4            0.000000         44.266930           0.00000
5            0.069409         40.728885          24.43733
6            0.075460         40.728885          24.43733
7            0.000000         44.266930           0.00000
8            0.074516         40.728885          24.43733
9            0.000000         59.206930           0.00000

   power_3rd_per_pri  churn
0          16.291555      0
1           0.000000      0
2           0.000000      0
3          16.291555      0
4           0.000000      0
5          16.291555      0
6          16.291555      0
7           0.000000      0
8          16.291555      0
9           0.000000      1
```

[12]:
```
## combination of three
```

There's 2 errors happen here at first

TypeError: 'method' object is not subscriptable

Subscriptable objects are objects with a **getitem** method. These are data types such as lists, dictionaries, and tuples. The **getitem** method allows the Python interpreter to retrieve an individual item from a collection.

Not all objects are subscriptable. Methods, for instance, are not. This is because they do not implement the **getitem** method. This means you cannot use square bracket syntax to access the items in a method or to call a method.

Consider the following code snippet:

- cheeses = ["Edam", "Stilton", "English Cheddar", "Parmesan"]
- print(cheeses[0])

This code returns "Edam", the cheese at the index position 0. We cannot use square brackets to call a function or a method because functions and methods are not subscriptable objects.

TypeError: 'function' object is not subscriptable

Iterable objects such as lists and strings can be accessed using indexing notation. This lets you access an individual item, or range of items, from an iterable.

Consider the following code:

- grades = ["A", "A", "B"]
- print(grades[0])

The value at the index position 0 is A. Thus, our code returns “A”. This syntax does not work on a function. This is because a function is not an iterable object. Functions are only capable of returning an iterable object if they are called.

The “TypeError: ‘function’ object is not subscriptable” error occurs when you try to access a function as if it were an iterable object.

This error is common in two scenarios:

```
When you assign a function the same name as an iterable
When you try to access the values from a function as if the function were iterable
```

[13]:
```
train.dtypes
```

[13]:
```
id                           object
activity_new                 object
campaign_disc_ele           float64
cons_12m                      int64
cons_gas_12m                  int64
cons_last_month               int64
date_activ                   object
date_end                     object
date_modif_prod              object
date_renewal                 object
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
churn                         int64
dtype: object
```

[14]:
```
main.dtypes
```

[14]:
```
id                     object
price_date             object
energy_1st_per_pri    float64
energy_2nd_per_pri    float64
```

```
energy_3rd_per_pri    float64
power_1st_per_pri     float64
power_2nd_per_pri     float64
power_3rd_per_pri     float64
churn                   int64
dtype: object
```

[15]:
```python
mean_year=mean_year.rename(index=str, columns={"energy_1st_per_pri":␣
 ↪"mean_year_price_p1_var",
                                               "energy_2nd_per_pri":␣
 ↪"mean_year_price_p2_var",
                                               "energy_3rd_per_pri":␣
 ↪"mean_year_price_p3_var",
                                               "power_1st_per_pri":␣
 ↪"mean_year_price_p1_fix",
                                               "power_2nd_per_pri":␣
 ↪"mean_year_price_p2_fix",
                                               "power_3rd_per_pri":␣
 ↪"mean_year_price_p3_fix"})
mean_year["mean_year_price_p1"] =mean_year["mean_year_price_p1_var"]␣
 ↪+mean_year["mean_year_price_p1_fix"]
mean_year["mean_year_price_p2"] =mean_year["mean_year_price_p2_var"]␣
 ↪+mean_year["mean_year_price_p2_fix"]
mean_year["mean_year_price_p3"] =mean_year["mean_year_price_p3_var"]␣
 ↪+mean_year["mean_year_price_p3_fix"]
```

[16]:
```python
mean_year.head(5)
```

[16]:
```
                                 id  mean_year_price_p1_var  \
0  0002203ffbb812588b632b9e628cc38d                0.124338
1  0004351ebdd665e6ee664792efc4fd13                0.146426
2  0010bcc39e42b3c2131ed2ce55246e3c                0.181558
3  0010ee3855fdea87602a5b7aba8e42de                0.118757
4  00114d74e963e47177db89bc70108537                0.147926

   mean_year_price_p2_var  mean_year_price_p3_var  mean_year_price_p1_fix  \
0                0.103794                0.073160               40.701732
1                0.000000                0.000000               44.385450
2                0.000000                0.000000               45.319710
3                0.098292                0.069032               40.647427
4                0.000000                0.000000               44.266930

   mean_year_price_p2_fix  mean_year_price_p3_fix  churn  mean_year_price_p1  \
0               24.421038               16.280694      0           40.826071
1                0.000000                0.000000      0           44.531877
2                0.000000                0.000000      0           45.501268
3               24.388455               16.258971      0           40.766185
```

```
4              0.000000                0.000000      0          44.414856

   mean_year_price_p2  mean_year_price_p3
0           24.524832           16.353854
1            0.000000            0.000000
2            0.000000            0.000000
3           24.486748           16.328003
4            0.000000            0.000000
```

[17]:
```
train.head(2)
```

[17]:
```
                                 id                      activity_new  \
0  48ada52261e7cf58715202705a0451c9  esoiiifxdlbkcsluxmfuacbdckommixw
1  d29c2c54acc38ff3c0614d0a653813dd                               NaN

   campaign_disc_ele  cons_12m  cons_gas_12m  cons_last_month  date_activ  \
0                NaN    309275             0            10025  2012-11-07
1                NaN      4660             0                0  2009-08-21

     date_end date_modif_prod date_renewal  forecast_cons_12m  \
0  2016-11-06      2012-11-07   2015-11-09           26520.30
1  2016-08-30      2009-08-21   2015-08-31             189.95

   forecast_cons_year  forecast_discount_energy  forecast_meter_rent_12m  \
0               10025                       0.0                   359.29
1                   0                       0.0                    16.27

   forecast_price_energy_p1  forecast_price_energy_p2  forecast_price_pow_p1  \
0                  0.095919                  0.088347              58.995952
1                  0.145711                  0.000000              44.311378

  has_gas  imp_cons  margin_gross_pow_ele  margin_net_pow_ele  nb_prod_act  \
0       f     831.8                 41.76               41.76            1
1       f       0.0                 16.38               16.38            1

   net_margin  num_years_antig                      origin_up  pow_max  \
0     1732.36                3  ldkssxwpmemidmecebumciepifcamkci    180.0
1       18.89                6  kamkkxfxxuwbdslkwifmmcsiusiuosws     13.8

   churn
0      0
1      0
```

[18]:
```
train.dtypes
```

[18]:
```
id                          object
activity_new                object
```

```
campaign_disc_ele           float64
cons_12m                      int64
cons_gas_12m                  int64
cons_last_month               int64
date_activ                   object
date_end                     object
date_modif_prod              object
date_renewal                 object
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
churn                         int64
dtype: object
```

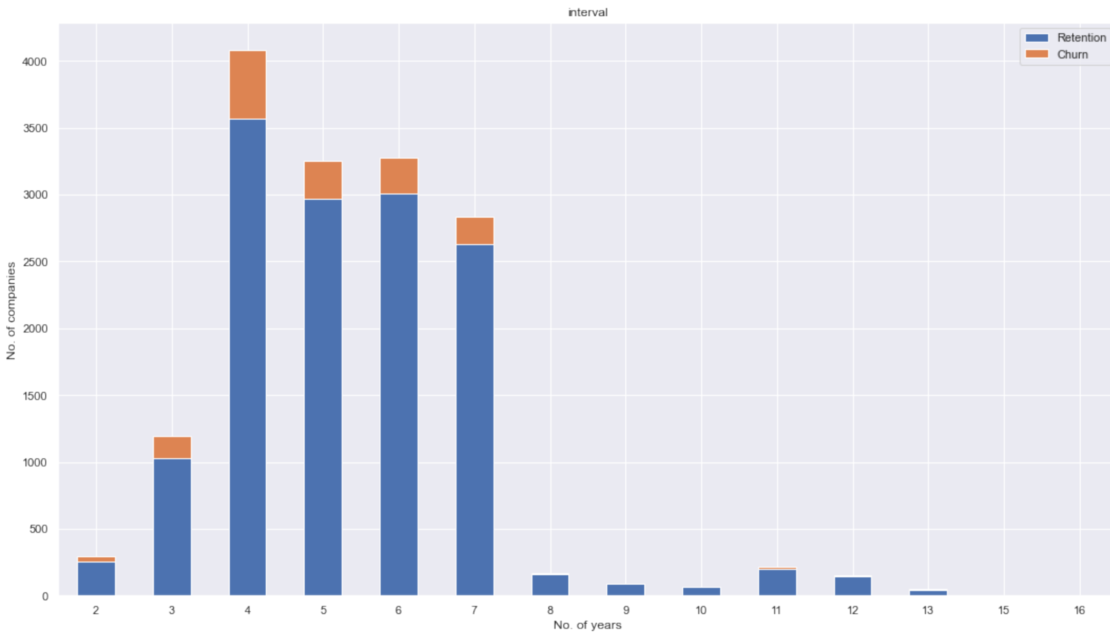
[19]:
```
train["date_activ"] = pd.to_datetime(train["date_activ"])
```

[20]:
```
train["date_end"] = pd.to_datetime(train["date_end"])
```

[21]:
```
train["interval"] = ((train["date_end"]-train["date_activ"])/np.timedelta64(1,␣
 ↪"Y")).astype(int)
```

[22]:
```
interval=train[["interval", "churn", "id"]].groupby(["interval",␣
 ↪"churn"])["id"].count().unstack(level=1)
interval_percentage = (interval.div(interval.sum(axis=1), axis=0)*100)
```

[23]:
```
interval.plot(kind="bar",figsize=(18,10),stacked=True,rot=0,title="interval")

# Rename legend
plt.legend(["Retention", "Churn"], loc="upper right")

# Labels
plt.ylabel("No. of companies")
plt.xlabel("No. of years")
plt.show()
```



We can clearly that churn is very low for companies which joined recently or that have made the contract a long time ago. With the higher number of churners within the 3-7 years of interval.

We will also transform the dates provided in such a way that we can make more sense out of those.

1. months_activ:
   - Number of months active until reference date (Jan 2016)
2. months_to_end:
   - Number of months of the contract left at reference date (Jan 2016)
3. months_modif_prod:
   - Number of months since last modification at reference date (Jan 2016)
4. months_renewal:
   - Number of months since last renewal at reference date (Jan 2016)

To create the month column we will follow a simple process: 1. Substract the reference date and the column date 2. Convert the timedelta in months 3. Convert to integer (we are not interested in having decimal months)

[24]:
```
def convert_months(reference_date, dataframe, column):
    """
    Input a column with timedeltas and return months

    """

    time_delta=REFERENCE_DATE-dataframe[column]
    months= (time_delta/np.timedelta64(1, "M")).astype(int)
    return months
```

[25]:
```
# Create reference date as provided on the exercise statement
REFERENCE_DATE=datetime.datetime(2016,1,1)
```

[26]:
```
dir(pd)
```

[26]:
```
['BooleanDtype',
 'Categorical',
 'CategoricalDtype',
 'CategoricalIndex',
 'DataFrame',
 'DateOffset',
 'DatetimeIndex',
 'DatetimeTZDtype',
 'ExcelFile',
 'ExcelWriter',
 'Float64Index',
 'Grouper',
 'HDFStore',
 'Index',
 'IndexSlice',
 'Int16Dtype',
 'Int32Dtype',
 'Int64Dtype',
 'Int64Index',
 'Int8Dtype',
 'Interval',
 'IntervalDtype',
 'IntervalIndex',
 'MultiIndex',
 'NA',
 'NaT',
 'NamedAgg',
 'Period',
 'PeriodDtype',
 'PeriodIndex',
 'RangeIndex',
 'Series',
 'SparseDtype',
 'StringDtype',
 'Timedelta',
 'TimedeltaIndex',
 'Timestamp',
 'UInt16Dtype',
 'UInt32Dtype',
 'UInt64Dtype',
 'UInt64Index',
 'UInt8Dtype',
```

```
'__builtins__',
'__cached__',
'__doc__',
'__docformat__',
'__file__',
'__getattr__',
'__git_version__',
'__loader__',
'__name__',
'__package__',
'__path__',
'__spec__',
'__version__',
'_config',
'_hashtable',
'_is_numpy_dev',
'_lib',
'_libs',
'_np_version_under1p16',
'_np_version_under1p17',
'_np_version_under1p18',
'_testing',
'_tslib',
'_typing',
'_version',
'api',
'array',
'arrays',
'bdate_range',
'compat',
'concat',
'core',
'crosstab',
'cut',
'date_range',
'describe_option',
'errors',
'eval',
'factorize',
'get_dummies',
'get_option',
'infer_freq',
'interval_range',
'io',
'isna',
'isnull',
'json_normalize',
```

```
'lreshape',
'melt',
'merge',
'merge_asof',
'merge_ordered',
'notna',
'notnull',
'offsets',
'option_context',
'options',
'pandas',
'period_range',
'pivot',
'pivot_table',
'plotting',
'qcut',
'read_clipboard',
'read_csv',
'read_excel',
'read_feather',
'read_fwf',
'read_gbq',
'read_hdf',
'read_html',
'read_json',
'read_orc',
'read_parquet',
'read_pickle',
'read_sas',
'read_spss',
'read_sql',
'read_sql_query',
'read_sql_table',
'read_stata',
'read_table',
'reset_option',
'set_eng_float_format',
'set_option',
'show_versions',
'test',
'testing',
'timedelta_range',
'to_datetime',
'to_numeric',
'to_pickle',
'to_timedelta',
'tseries',
```

```
 'unique',
 'util',
 'value_counts',
 'wide_to_long']
```

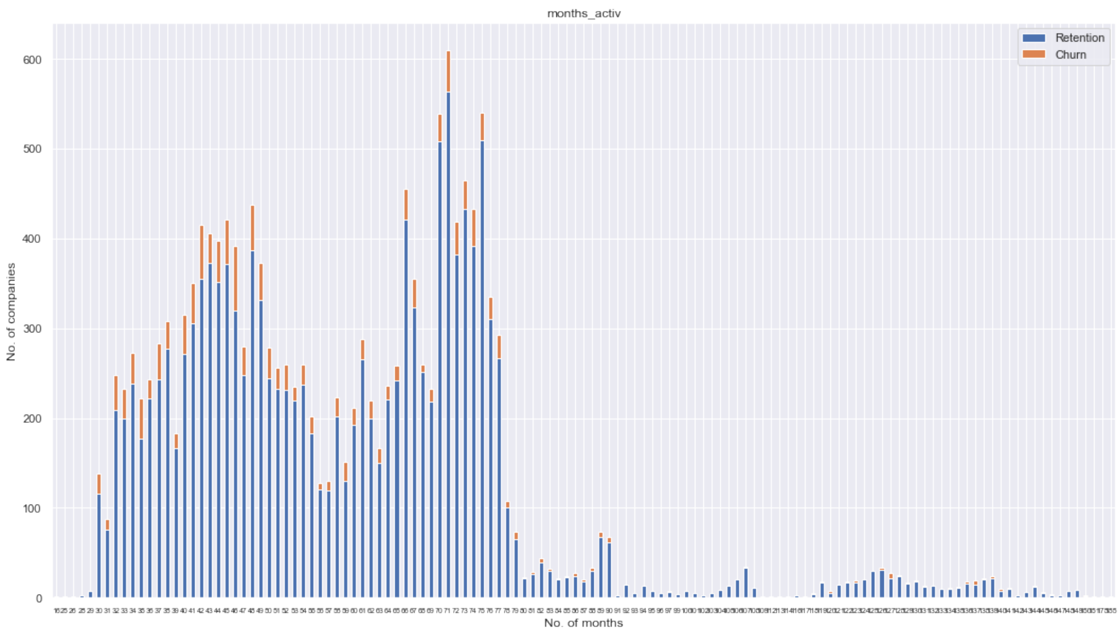
[27]:
```
train["date_modif_prod"] = pd.to_datetime(train["date_modif_prod"])
```

[28]:
```
train["date_renewal"] = pd.to_datetime(train["date_renewal"])
```

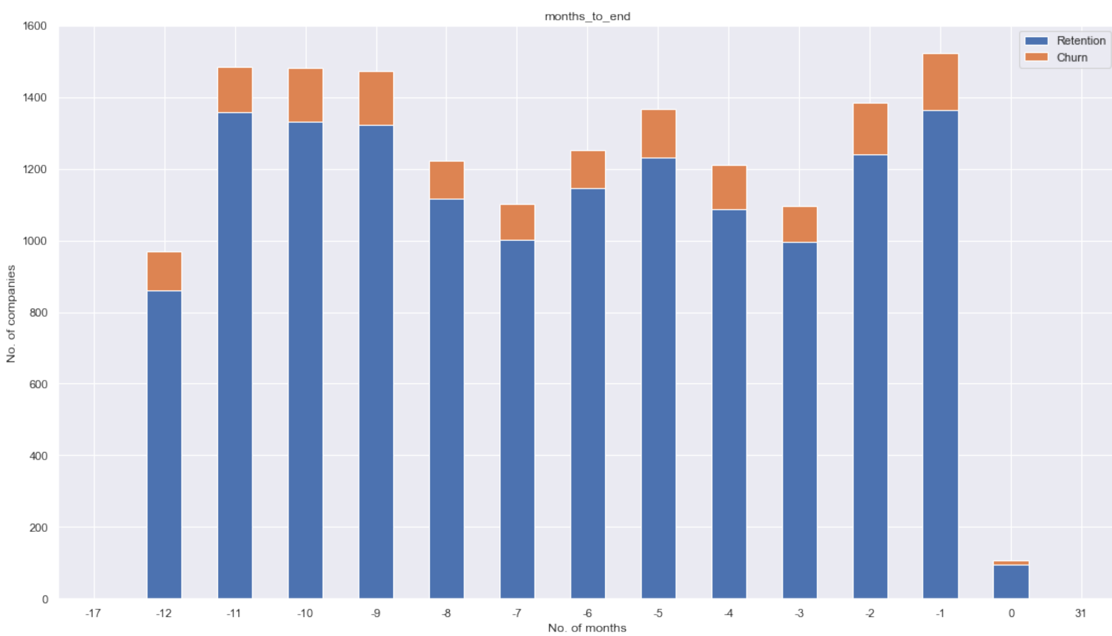
[29]:
```
train["months_activ"] = convert_months(REFERENCE_DATE, train, "date_activ")
train["months_to_end"] = convert_months(REFERENCE_DATE, train, "date_end")
train["months_modif_prod"] = convert_months(REFERENCE_DATE, train,␣
↪"date_modif_prod")
train["months_renewal"] = convert_months(REFERENCE_DATE, train, "date_renewal")
```

[30]:
```
def plot_churn_by_month(dataframe, column, fontsize_=11):

    ## Here we will Plot churn distribution by monthly variable
    temp=dataframe[[column, "churn", "id"]].groupby([column, "churn"])["id"].
↪count().unstack(level=1)
    temp.plot(kind="bar",
              figsize=(18,10),
              stacked=True,
              rot=0,
              title=column)

    # Rename legend
    plt.legend(["Retention", "Churn"], loc="upper right") ##to visualize in the␣
↪upper right a visualization of what the colors refers to

    # Labels
    plt.ylabel("No. of companies")
    plt.xlabel("No. of months")

    # Set xlabel fontsize
    plt.xticks(fontsize=fontsize_)
    plt.show()
```

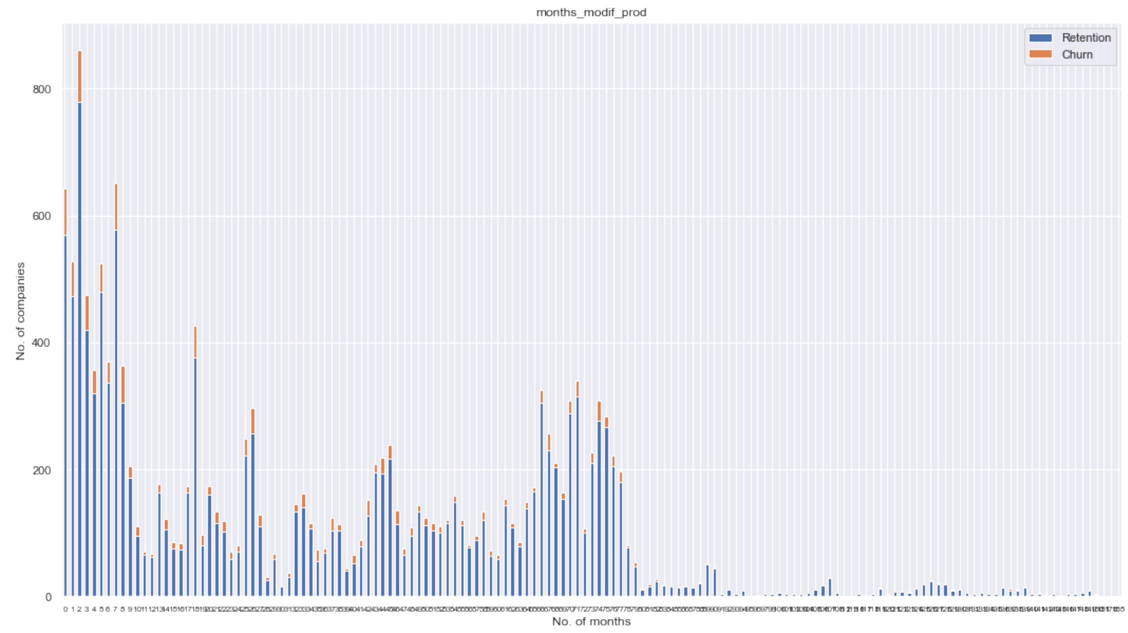
[31]:
```
plot_churn_by_month(train, "months_activ", 7)
```



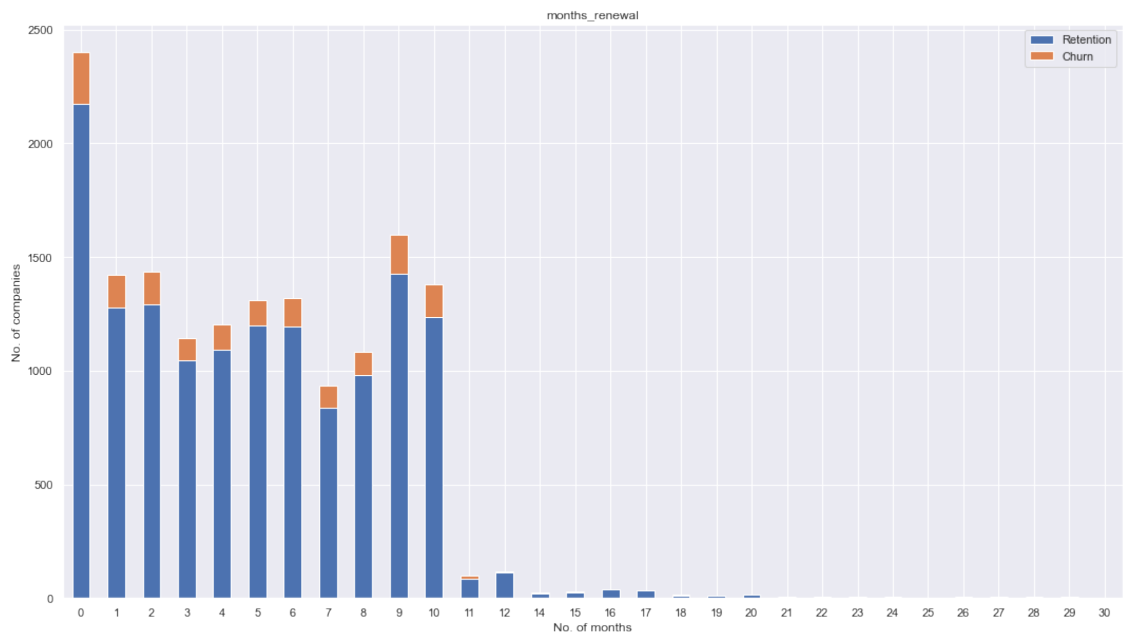
[32]:
```
plot_churn_by_month(train, "months_to_end")
```



[33]:
```
plot_churn_by_month(train, "months_modif_prod", 8)
```



[34]:
```
plot_churn_by_month(train, "months_renewal")
```



After we visualize date columns next we will remove date columns:-

[35]:
```
train.drop(columns=["date_activ", "date_end", "date_modif_prod",␣
 ↪"date_renewal"],inplace=True)
```

[36]:
```
train.head(2)
```

[36]:
```
                                 id                     activity_new  \
0  48ada52261e7cf58715202705a0451c9  esoiiifxdlbkcsluxmfuacbdckommixw
1  d29c2c54acc38ff3c0614d0a653813dd                              NaN

   campaign_disc_ele  cons_12m  cons_gas_12m  cons_last_month  \
0                NaN    309275             0            10025
1                NaN      4660             0                0

   forecast_cons_12m  forecast_cons_year  forecast_discount_energy  \
0           26520.30               10025                       0.0
1             189.95                   0                       0.0

   forecast_meter_rent_12m  forecast_price_energy_p1  \
0                   359.29                  0.095919
1                    16.27                  0.145711

   forecast_price_energy_p2  forecast_price_pow_p1 has_gas  imp_cons  \
0                  0.088347              58.995952       f     831.8
1                  0.000000              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                 16.38               16.38            1       18.89

   num_years_antig                         origin_up  pow_max  churn  \
0                3  ldkssxwpmemidmecebumciepifcamkci    180.0      0
1                6  kamkkxfxxuwbdslkwifmmcsiusiuosws     13.8      0

   interval  months_activ  months_to_end  months_modif_prod  months_renewal
0         3            37            -10                 37               1
1         7            76             -7                 76               4
```

Next we will try to refer boolean values in has_gas columns to 0 or 1

[37]:
```
train["has_gas"]=train["has_gas"].replace(["t", "f"],[1,0])
# di = {"f": 0, "t": 1}
# train["has_gas"] = train.replace({"has_gas":di})
```

[38]:
```
train.head(2)
```

[38]:
```
                                 id                     activity_new  \
0  48ada52261e7cf58715202705a0451c9  esoiiifxdlbkcsluxmfuacbdckommixw
1  d29c2c54acc38ff3c0614d0a653813dd                              NaN

   campaign_disc_ele  cons_12m  cons_gas_12m  cons_last_month  \
0                NaN    309275             0            10025
```

```
1                NaN      4660             0                0

   forecast_cons_12m  forecast_cons_year  forecast_discount_energy  \
0           26520.30               10025                       0.0
1             189.95                   0                       0.0

   forecast_meter_rent_12m  forecast_price_energy_p1  \
0                   359.29                  0.095919
1                    16.27                  0.145711

   forecast_price_energy_p2  forecast_price_pow_p1  has_gas  imp_cons  \
0                  0.088347              58.995952        0     831.8
1                  0.000000              44.311378        0       0.0

   margin_gross_pow_ele  margin_net_pow_ele  nb_prod_act  net_margin  \
0                 41.76               41.76            1     1732.36
1                 16.38               16.38            1       18.89

   num_years_antig                         origin_up  pow_max  churn  \
0                3  ldkssxwpmemidmecebumciepifcamkci    180.0      0
1                6  kamkkxfxxuwbdslkwifmmcsiusiuosws     13.8      0

   interval  months_activ  months_to_end  months_modif_prod  months_renewal
0         3            37            -10                 37               1
1         7            76             -7                 76               4
```

4.0.1 Categorical data and dummy variables

When training our model we cannot use string data as such, so we will need to encode it into numerical data.

The easiest method is mappingeach category to an integer (label encoding) but this will not work because the model will misunderstand the data to be in some kind of order or hierarchy, 0 < 1 < 2 < 3 …

For that reason we will use a method with dummy variables or one-hot encoder

[39]:
```python
print(train.columns)
```

```
Index(['id', 'activity_new', 'campaign_disc_ele', 'cons_12m', 'cons_gas_12m',
       'cons_last_month', 'forecast_cons_12m', 'forecast_cons_year',
       'forecast_discount_energy', 'forecast_meter_rent_12m',
       'forecast_price_energy_p1', 'forecast_price_energy_p2',
       'forecast_price_pow_p1', 'has_gas', 'imp_cons', 'margin_gross_pow_ele',
       'margin_net_pow_ele', 'nb_prod_act', 'net_margin', 'num_years_antig',
       'origin_up', 'pow_max', 'churn', 'interval', 'months_activ',
       'months_to_end', 'months_modif_prod', 'months_renewal'],
      dtype='object')
```

[40]:
```python
train['campaign_disc_ele'].unique()
```

[40]:
```
array([nan])
```

[41]:
```python
train['origin_up'].unique()
```

[41]:
```
array(['ldkssxwpmemidmecebumciepifcamkci',
       'kamkkxfxxuwbdslkwifmmcsiusiuosws',
       'lxidpiddsbxsbosboudacockeimpuepw',
       'usapbepcfoloekilkwsdiboslwaxobdp',
       'ewxeelcelemmiwuafmddpobolfuxioce'], dtype=object)
```

[42]:
```python
train['imp_cons'].unique()
```

[42]:
```
array([ 831.8 ,    0.  , 1052.37, …,   67.03,   46.98,   18.05])
```

[43]:
```python
train['has_gas'].unique()
```

[43]:
```
array([0, 1], dtype=int64)
```

[44]:
```python
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15674 entries, 0 to 15673
Data columns (total 28 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   id                        15674 non-null  object
 1   activity_new              6369 non-null   object
 2   campaign_disc_ele         0 non-null      float64
 3   cons_12m                  15674 non-null  int64
 4   cons_gas_12m              15674 non-null  int64
 5   cons_last_month           15674 non-null  int64
 6   forecast_cons_12m         15674 non-null  float64
 7   forecast_cons_year        15674 non-null  int64
 8   forecast_discount_energy  15674 non-null  float64
 9   forecast_meter_rent_12m   15674 non-null  float64
 10  forecast_price_energy_p1  15674 non-null  float64
 11  forecast_price_energy_p2  15674 non-null  float64
 12  forecast_price_pow_p1     15674 non-null  float64
 13  has_gas                   15674 non-null  int64
 14  imp_cons                  15674 non-null  float64
 15  margin_gross_pow_ele      15674 non-null  float64
 16  margin_net_pow_ele        15674 non-null  float64
 17  nb_prod_act               15674 non-null  int64
 18  net_margin                15674 non-null  float64
 19  num_years_antig           15674 non-null  int64
 20  origin_up                 15674 non-null  object
```

```
 21  pow_max                    15674 non-null  float64
 22  churn                      15674 non-null  int64
 23  interval                   15674 non-null  int32
 24  months_activ               15674 non-null  int32
 25  months_to_end              15674 non-null  int32
 26  months_modif_prod          15674 non-null  int32
 27  months_renewal             15674 non-null  int32
dtypes: float64(12), int32(5), int64(8), object(3)
memory usage: 3.0+ MB
```

[45]:
```
## we will drop "campaign_disc_ele" column
```

[46]:
```
train_1 = train.drop(columns=['campaign_disc_ele'])
```

[47]:
```
train_1
```

[47]:
```
                                     id                      activity_new  \
0      48ada52261e7cf58715202705a0451c9  esoiiifxdlbkcsluxmfuacbdckommixw
1      d29c2c54acc38ff3c0614d0a653813dd                               NaN
2      764c75f661154dac3a6c254cd082ea7d                               NaN
3      bba03439a292a1e166f80264c16191cb                               NaN
4      568bb38a1afd7c0fc49c77b3789b59a3  sfisfxfcocfpcmckuekokxuseixdaoeu
…                                   …                                 …
15669  18463073fb097fc0ac5d3e040f356987                               NaN
15670  d0a6f71671571ed83b2645d23af6de00                               NaN
15671  10e6828ddd62cbcf687cb74928c4c2d2                               NaN
15672  1cf20fd6206d7678d5bcafd28c53b4db                               NaN
15673  563dde550fd624d7352f3de77c0cdfcd                               NaN

       cons_12m  cons_gas_12m  cons_last_month  forecast_cons_12m  \
0        309275             0            10025           26520.30
1          4660             0                0             189.95
2           544             0                0              47.96
3          1584             0                0             240.04
4        121335             0            12400           10865.02
…             …           …                …                 …
15669     32270         47940                0            4648.01
15670      7223             0              181             631.69
15671      1844             0              179             190.39
15672       131             0                0              19.34
15673      8730             0                0             762.41

       forecast_cons_year  forecast_discount_energy  forecast_meter_rent_12m  \
0                   10025                       0.0                   359.29
1                       0                       0.0                    16.27
2                       0                       0.0                    38.72
3                       0                       0.0                    19.83
4                   12400                       0.0                   170.74
```

```
…                   …                      …                      …
15669               0                    0.0                  18.57
15670             181                    0.0                 144.03
15671             179                    0.0                 129.60
15672               0                    0.0                   7.18
15673               0                    0.0                   1.07

       forecast_price_energy_p1  forecast_price_energy_p2  \
0                      0.095919                  0.088347
1                      0.145711                  0.000000
2                      0.165794                  0.087899
3                      0.146694                  0.000000
4                      0.110083                  0.093746
…                           …                       …
15669                  0.138305                  0.000000
15670                  0.100167                  0.091892
15671                  0.116900                  0.100015
15672                  0.145711                  0.000000
15673                  0.167086                  0.088454

       forecast_price_pow_p1  has_gas  imp_cons  margin_gross_pow_ele  \
0                  58.995952        0    831.80                 41.76
1                  44.311378        0      0.00                 16.38
2                  44.311378        0      0.00                 28.60
3                  44.311378        0      0.00                 30.22
4                  40.606701        0   1052.37                  3.18
…                        …      …        …                    …
15669              44.311378        1      0.00                 27.88
15670              58.995952        0     15.94                  0.00
15671              40.606701        0     18.05                 39.84
15672              44.311378        0      0.00                 13.08
15673              45.311378        0      0.00                 11.84

       margin_net_pow_ele  nb_prod_act  net_margin  num_years_antig  \
0                   41.76            1     1732.36                3
1                   16.38            1       18.89                6
2                   28.60            1        6.60                6
3                   30.22            1       25.46                6
4                    3.18            1      823.18                6
…                     …          …         …              …
15669               27.88            2      381.77                4
15670                0.00            1       90.34                3
15671               39.84            1       20.38                4
15672               13.08            1        0.96                3
15673               11.84            1       96.34                6

                              origin_up  pow_max  churn  interval  \
```

```
0      ldkssxwpmemidmecebumciepifcamkci  180.000     0         3
1      kamkkxfxxuwbdslkwifmmcsiusiuosws   13.800     0         7
2      kamkkxfxxuwbdslkwifmmcsiusiuosws   13.856     0         6
3      kamkkxfxxuwbdslkwifmmcsiusiuosws   13.200     0         6
4      lxidpiddsbxsbosboudacockeimpuepw   75.000     0         6
…                                    …       …    …      …
15669  lxidpiddsbxsbosboudacockeimpuepw   15.000     0         3
15670  lxidpiddsbxsbosboudacockeimpuepw    6.000     1         4
15671  lxidpiddsbxsbosboudacockeimpuepw   15.935     1         3
15672  lxidpiddsbxsbosboudacockeimpuepw   11.000     0         4
15673  ldkssxwpmemidmecebumciepifcamkci   10.392     0         6

       months_activ  months_to_end  months_modif_prod  months_renewal
0                37            -10                 37               1
1                76             -7                 76               4
2                68             -3                 68               8
3                69             -2                 69               9
4                68             -3                 68               8
…             …            …                …             …
15669            43             -4                  7              19
15670            40             -7                 40               4
15671            46             -1                 46              10
15672            40             -7                 40               4
15673            72            -11                 72               0

[15674 rows x 27 columns]
```

[48]:
```python
train_1['activity_new'].unique()
```

[48]:
```
array(['esoiiifxdlbkcsluxmfuacbdckommixw', nan,
       'sfisfxfcocfpcmckuekokxuseixdaoeu',
       'sscfoipxikopfskekuobeuxkxmwsuucb',
       'cssldxpacdmuuaulamxdekcokibauube',
       'ppcxfxbffsxaakxamcdpexdoxulfwwae',
       'kkklcdamwfafdcfwofuscwfwadblfmce',
       'apdekpcbwosbxepsfxclislboipuxpop',
       'lckfspcixfmlwudlisbaiacuioicciam',
       'daobdssbkieoukwxbopxiiospudkopwl',
       'cccpsslxcemdlomsaffxsecccbxpdkax',
       'kwuslieomapmswolewpobpplkaooaaew',
       'ifppdlcfssupdcsdcclkoubulccouwml',
       'mwmuuaeloxbawummwfwcmxckmsfibpwk',
       'almlfkoedpwfdmmsebsdwueskducuiok',
       'wlxfbefauebfbauopppswxppaafdkoap',
       'cwkwaxadbfukekuspislmbipbkxdudla',
       'cluecxlameloamldmasudocsbmaoamdw',
       'pmccdfmdxpmdeodsasdeboedildifmlp',
```

```
'paoauaefwcbedmiowwmokakuisslckbd',
'bapcuxcousodpaabofsesslupodaapcx',
'dupxuibdflmskeieweeofcaluuuiioix',
'fmwdwsxillemwbbwelxsampiuwwpcdcb',
'xbsbaipfluioualwapemiublmepsbuoo',
'sbimkkplabbebeauuidmbiopofmamcau',
'xlwlakkoiebpwueucmdpsslcfiaxeuec',
'awoxilkbfwkemdseewlpbolidusdaeim',
'wwsobfkddbekdkdawkeiemomebebebii',
'ckfxocssowaeipxueikxcmaxdmcduxsa',
'cwswmoddxdpulwofkbfdwilbwuodkeme',
'idpmdduoieixxoklkufmspokimxcxeid',
'wxemiwkumpibllwklfbcooafckufkdlm',
'mpicaaibskkfmxoblmwwwuuwpkecacil',
'ebadppbpcufaidikpolbbxxfuelueofp',
'ipdldckuswupeifllfbwccfpeafludfi',
'ucwesoppcsdxffsplelumwixoapxpidx',
'mksmfeexfwuuwsbpamfmxbikklcdwkbb',
'dmklwapxmxxalfwupxepeiuoooduaueb',
'ipiiicokaeexiaebwmkecbdummcdmccu',
'wkcccadoacxlcoukpxawpopxsfpculmd',
'sffadmsbuamddwapeumdfibkmpkdicmc',
'ilkfsaapsxpkcpswblldddfmpamwelpxi',
'kfewlppusowokloemwafomsspodpfkff',
'akokxbmlwukcmwlimosloemdplieuuwm',
'sumdxiaiudmaioicexmiwuudlblkissm',
'fmbupoclkdkboesbfklilouawfasmdfc',
'pxxlamsdbssumpslpkduwskxodummews',
'piaosodsowlfpxipbiudiiwuikoeiisd',
'epmwweimsesebmlpseufxpckcxmmuxol',
'mloddpisebmawpocbcspfiwieodsmded',
'fibkpxbliefxfmeielcidsckcxkpofaa',
'saxlifeumaobawxpemwuopbwwldlucff',
'apmpdisoaulbesoawkkekkcpokeaeucl',
'bxopwkbwdewxssbmkwcummkaakbwafxf',
'bwpaswkpcilmlklklcapcwwumwaodaoo',
'lefskxfcmdbaxwoxdsusxfbdwcmfbeif',
'useklixsdbmxwcalbkkbxdfoilisbkcs',
'pmedwkpuckbppeoecxiccwxluwkxdkpe',
'dbiuixkxodixflmixcmbmmemluwsxaie',
'kcxodwiuawoifbipbbxmpxleslblfbwb',
'lfulxaxsclsidbsxksecakbplcesecml',
'fkmblacmaapkaoauabpwpuweokkeiali',
'ibfoefbbekcwubufxslcoewkfswxolua',
'pffpiboilxxdeluedfxssmaklbdplfmi',
'cwofmuicebbcmiaaxufmfimpowpacobu',
'xwwsfoileuxkwbcxupadudcfoecmmdda',
```

```
'ixuciffexbsibwibpcwdmfwcoixkfscw',
'lkeudbeowbapkpfodoxacpwdpaeuwxcx',
'liekubiseoiflbsuiubbfcwpfsaxbofk',
'ckadsdebplpkplelfspfoiucmxkeppus',
'eoakmfoebekdxxupfdbookikcdkkclco',
'ibkiiwcxiccxpoedpweiuxwbxbuewbxm',
'acpmlkfcadicfcpslmoxcdakikieeeso',
'wdkxkxomwacxkwubwisleblebfekluib',
'dbapibbxclxuuosukfokcodpbplkeclc',
'ppbmluelablufxsafiemmpxufupiwaik',
'pkcmeicamloxowfbfleaxcxiucapulee',
'bilukaxxaslukscimbduakwseilcxupx',
'dwamuluiuaiowuxmesuuilkbobidcmfo',
'lpolmawamsuskeebxlsamfdwcuecsuul',
'xclapbmiasllcspuclsumusikimpckek',
'axsupumdipebmlbiwolspmkdouoiddbc',
'fskfsbkdioupwobbsaoospkxaafmwobl',
'xfefpeffxidcliucwaufxsxalwodlmfo',
'lwdukaafdapweskskfuemwppusodiwfd',
'clbkplmouokdpxiwxebwculxxsdiuwap',
'fsoixmxdfowweuxusdclkapubxluefbi',
'obowdwuxkxoamcesspxmebxlsbmuiocm',
'buamfwafmxwxwflcxscksfkapuelecop',
'iabskbxembdweacmalxplabbxupsaadc',
'fcbfabofwcdaosksieduepeeusawfdsi',
'spcildxusfwkiacbxokefewaoalakiee',
'dfcsaaowsemmabpepocaeaaecfwppxxk',
'fcoesawwkbuwfswmpimwkiplsumkoiei',
'cawpikmdedxaabcllupdpfubuomcddxw',
'ixxfakuxuscleimexkbxmeblekcsccbw',
'uapfospaxexfspkkskumakxcdlwuiuul',
'cfdsselwimsklimddecfifseabdkxfcs',
'xlplcsecoeklisfpdmalaxcwekdekdua',
'eckbfdkkkfoxpeffpacbikwpeicksulu',
'focpboxmeucdmbaiseimfpiubpcmmcss',
'wdkbuxwfkbefwplcoudfalpfafdfpfax',
'sxublbwoeuckkocekklxkllcdxxaisop',
'afeccskfmobewicibxofslkxecsuekfi',
'dascbpmpfmfuexmfocmafmmamckebsxe',
'fdcwobsbkeeisaofmssomixkmefxslpk',
'ddxbfdebxlflclxdwkkdesieuudlmeoe',
'ulfmpofsfppwdswdepemulsumksexsbm',
'duidibwdmowiudemasiwabceucckesdw',
'kfombxdwubpkokafeecduxbwkkuadfwc',
'sudafembxmksswlmaeawmfpdkfpmwmok',
'oclfuccbxapuklpeowbabcpawcbwxesk',
'kmxccaddbdpaaolkbidlobeefsbbcxca',
```

```
'cxdlpsmkulssdwsoskdmisdmdbcuebww',
'scdupipedpwefxmbadpfddxkwpxxeloc',
'ppwwsldwidxcfoieopwwsxaixpkbaswl',
'odxludcoupskdmbbfbbwaaxwidexafau',
'kwiixecplmewlscxsdiudmolbcfmdbxs',
'dplebxkdmsmpsiealoubbwiwlebpcidm',
'imfclodmbmabakpawdmwfssefabcemoe',
'fwfifipcmomuacibieekffilsbxsfcsk',
'mcexdumuplwdoisamslcdcmmliwmwbwp',
'ebebpkkmmosllfpxkeamimcmfpppfxbu',
'owuppmeskiukobiwdkfxoufexesaewil',
'iimppdwbwecsmxcpliaesdiasxccpwie',
'wfiuolfffsekuoimxdsasfwcmwssewoi',
'balskueexlmuccwdffilikwxasupasxf',
'okocbusoacxuklxbuswbklixodopwblk',
'fcokoocmubsiclsbbefulmfiplksskbf',
'mobaudseeepfkosbeppubbomlsxicpdm',
'bdwcuablpkfxxipoksbuuueckdukdifb',
'uiouuawillpcssldoeemcddcpfseebsw',
'lakksuxlbesxeskmemsppkiebumeimkm',
'axxmxawdwbioollfeexcadkmmksiibcs',
'fcxomxuocpmxmfukiusfbfaccmiwfopx',
'xwpmxlfwakscmxoiipbcckoabfouxebx',
'ddkpdekmbfdffwdmabkiiilolsxswccl',
'cmdfecexccsmowuoksewsukfcwlplamd',
'axekkipoplpalkpikkkfdumlapcufmlb',
'kdoboksimosaxdfpooiiisaxkmxaskxo',
'acefxcckbdxakciukwuwepweawbkwmii',
'eeaebbffbxdouffsoluusplfcxiadblb',
'pudsxpkefiudxxfcumemocbpuklxiufa',
'kkpddsilciodwwwffucmkflilcpfaumo',
'peaesobbauddfswfsudwudcwmeoddoiw',
'llcslipokulfubiffkomaedkecamdoib',
'akifpcasepewecemeiapksusfxomwwbc',
'ekcdpfisfuemmpcauicoawcowfwlbpdu',
'okxcpskmecbumwcifxfxdofxocupwwom',
'awmiwfbewabdaduimwoefiluxdcdwsxb',
'aiwacxxawfolwlocxdmuwecicpclwpdo',
'usculwxxobpfompufcclxielfiosluce',
'xscbuwcbpwsilaeadffielubxpfpmpxw',
'ocskiadudoffubcmbomoslkcddxwfsuf',
'eldwcmdwfekwdubwxpuaklxdmkucsdki',
'xufxifdsuplwioblexsdedieaowlkcox',
'iwmoskaicewfewukldfwcdwlxcwwoeom',
'cwleuplwopmllxkbabaoeopmxxmfaiod',
'alkuukubieaxcobeeowowmokpbilomax',
'edxmolisbfbwlpmccduowkxpkiiooess',
```

```
'mxokpaemsfodeossocaxbsdsiwfbewux',
'xkfpfmcwobuumawmkxleudppfwiwwbmb',
'plflscsfepabwxekcdlecbilsbxakwsd',
'mpibpwodmampwkmexuskeubkcxfpifkc',
'libuewofdiwukcoeempcibcwcwepldap',
'dbaboadcmfudklmkkueepfamikbeaiwo',
'pbpfffswspwswuxudcdibsmdkpokflmi',
'mbiecfsdmkwubbksoapxsficcmioesue',
'upolusukxpckksmddueklcxoumssbuwd',
'ifxluxelbupfwopcpulesflaaffmkipp',
'clafkuosbouwewlexoewscfebeilpfok',
'cpsbiipoacmouecemlddaxxdllacksaw',
'duiwascsdupcmdfkspbukuuaklsawmmc',
'xpmaxkfbpsbuliopadcpsolcbdboopbf',
'ffmciapbdkcwwiwpuakakmiexskcmxfc',
'oxkddxsmiwadlcxkulieewkokxiucsfx',
'uaxxxwkppmwfciofupisxsdeauikeppw',
'mfwxlsadmxwikwwoalweepalsmplxlbp',
'bdbcaommfeelfuofobfauflkiolollwk',
'feccpaplkaopdipbicfuiacpcieicauo',
'amucfxoxeidufkwdiscepiisxeipuawb',
'cfeluxakapclbcismpfoefdmmplddekc',
'amfioduwmscbccofekcfcxpxokiiadpx',
'xffaaufximocoapsmwpfwpmbbifdbdea',
'uuukommlocffflaemeeppkappcbdpmlw',
'ieseoxduxoakdspubslseeokdkxiueeo',
'kleicdldcamuaislmkwowllimpblacpf',
'scoewuuiibsmdpixxxxdxbpebkddwxbf',
'mafuxfeoabiaxwbwmmefmefbpfpbklul',
'mummxffocmpaikpkxeekmbdxklcikwuc',
'widaskucxiudllboubewwofmciikxswl',
'mekpscciupdkafxuaauboomoiowmouob',
'umxeuawseikkkxpxsbbafiuwowufsmxs',
'udfcdxexxmdfakwaafplfiplaidudpex',
'ckmosmiuspskosddifpslwllwfkscxid',
'kackmuikblemiplimmldfpdsobmuwkac',
'fuoxmsaoxbbpiwxfacdswabokfdxxwff',
'bsdfdwimlxbckukbdpidkbilwkebxbib',
'wcxsffwcewaafuxlpaaxdffuipdfuula',
'bweosldmwklxfssebduslduxppifcckp',
'ommcdoxsluxpfksewskappuxifxxdaaf',
'fcdfsumaxdslpwpxekaxasfuffeakxca',
'xuebxddpsuiwcoubouucmwblddafkmec',
'mkauefdplcsocbdeeopxiiuoumpawpds',
'usfobxkbxdpsipadfucolkwuboammmcl',
'luwommfoxofwxdpxlpxaipmxpeskxupu',
'pposwlfwppaicikfaoeoaekwabfdfmpe',
```

```
'lblmkbemolxxlkicccsucmoapesxsplx',
'cxduoxssmucwpkpsxmiflmicsifosexk',
'oiimbufsmucwemdlmdpwoimsmmalaicf',
'swxildmwpxuuwuuoesmobpewaakakssi',
'fupwmsbfffpmcasudlsdmxckmuxsuaku',
'lplipswbpdloacuapedmspmesfkbeocu',
'oukkkbkoucsipomauwmbiclafakiceap',
'omdcsbkeibpplsubwbuwsmxewfwdiexc',
'pdefimwadmfwlxxffosumsoeallpupdd',
'kpullxmfmuukcibwibmibwpacpmuabuf',
'ksukxublefmwkfkclwamiwecuoplbsec',
'updsxswiffpfixmssiwcfociadowkbsc',
'feadeidokwlullcdkcefdafcwkilxauu',
'swbpoldwkuimcalxooacooepdkodcwic',
'ubklfxkiwxukoeuwxaiccsoufloucoip',
'pwufpsaalmuafkmpbmeosmpubdwumafu',
'iwueusllaxumufdpwoaewwoipdfsmmsm',
'wioskwbepkflkibibiepiowcoxoksxes',
'xlpufbwemoedxpsssmkbipwwicsadebw',
'xcswxcciuaxpacidfbixxmwkdmmskxkc',
'eupfpibdulbskcamamwsllxwskxixpuc',
'pkopxblbpasbfkkxdomwbbisssfabldi',
'bdcmbkmcaussbssoseifimsfkpcuaksk',
'dsbxbsuowflfmiimikwcdpepuoideeac',
'fswbowspcfidbaiuuwuicmmxdffupofp',
'bupfiimslwmifcsdllukmaeaxmbasium',
'kllldxcildwkssbmoabmsdffmawsafsf',
'keweuwsssmcfoafcwouubwsobmiocaca',
'xkxoxulxsowwwupmlobllkuamakibxps',
'skumedsiselixilifkdabadskamexiul',
'fxwoeaeiuwupxmeoxdmbfeefexbiuufi',
'fcfeucllkebuploesusiailfleaffdxf',
'abdiclmommkccxulxufwiiofdakbmfum',
'sodadseolpdpoopxaekobldpoixlmeem',
'ubaiusclasemueafdkuxwdemkfciboow',
'olpwwmwiidfiusocbaidbsckxbpoipcu',
'sbolemmfddlosupuwbcawusmbwmdmdfw',
'aumipeuxxkfeepiikplpcoifakioeeel',
'dxmfpsflslufmxlmwdmbkikffowmfmum',
'xkabcpelpkepsbpmlcamekocosdckaoo',
'pfcocskbxlmofswiflsbcefcpufbopuo',
'ambaaxsxxwfuspsuabupewfpbbksmcoo',
'kbubsdsdixweumcepwbpfoeweflseaeu',
'wixkdsawloxffiwmwswkcudoewxmawou',
'ckeukxpofbwoacsdeimeoxeuxdblpkwl',
'flfasacakomelffwefsodlxedkoxfels',
'exespdalufcdobebbdlcbmbficidkolw',
```

```
'kmlwkmxoocpieebifumobckeafmidpxf',
'bifsofauudeufolakwmpwwdxxcxeflas',
'abmaakxelmowmifxxfciwioaulelacpw',
'eamiapdokbfumefocubefudcowecllla',
'xcsomlbewdeuffxafeaeclodxwwpsfcf',
'bdwasdkxspekskwxikumuebmfufflifx',
'uliuakkedkcxlkudmkdfefppluxdcefp',
'cwcwxxdfpcxwfouoaeoofsluifdoxxpe',
'cfmsssiwbsbimcdaipfesuuapasxssib',
'bddmmiluukefokufkfbsbkwufodoeakc',
'exmccxcauwolkacaceedipbcmodfedfl',
'aixiudexbxdwwbpiakidekipbubxiamk',
'psediulbpfpisimfdbadbpdfmsfmkwbd',
'wuxsowlmdewcxwfafdwecwsfdbouldex',
'posfbkwcpabfkpkfepfwepwifpubxlpf',
'klbickwapkflumbxxksciecxacfbukpi',
'lwuamescopimpiefllxsdbopkbwpxucm',
'pluuboemkxalfambklsfckokxlwalpbo',
'lasmxduedduxxkmwliaudpdsfcacewpo',
'mxpxpdpbofpubucubxeepxlsxlmckwbl',
'beplffiwdfsmiuodulsfscelscscbdix',
'kmkacdccelocksmlpallpcwpiicoewsw',
'ooiladiddxkcikildpdfsbwwblcwxxkm',
'xpwokbdseslumlsislulloalddkioslw',
'klpmksubowwaicwpcmpiioblpxiwkais',
'bkwukbmxeuobswdbbxdblsbfmfcasamo',
'mdxeuxlfoulmupwmlbdumfampkfaswip',
'sofuiceubmokawuieelmcikmspsdbkmu',
'opemwsspmskladcbclpkmilallxmiook',
'mielwdslmxmiapbblalokbkiufwfiaup',
'duukumowsmmdaaadwsdwwieefdmiebpm',
'fckccskocemwpbxpupsmwalfcibxssbc',
'pkdwiifdckesdafkuoalwpxwkuwpxubb',
'kfsxcafkaldkkafdobmlckomemmfeupi',
'ciixbauekwabolfbbbsswfupoiioowsd',
'mcxebempwbbsiaoexabmefpmfmduwkfp',
'aseamwliciokfemccpxaouxwcubucxux',
'dwdflbsopucwoxdmccmulwiiefiiabel',
'axspbsdxabbfiplilxkxudsbbiwslocc',
'bplwscwaolwimidobccpixabiflaamld',
'wsboswwcbdfwmufackeodiuooiodoksd',
'pudpxobkpuxbalsmeimfkwocseoamsoi',
'oluwfsaokeeksmxfwkksoikfwkluuacd',
'aamfdbbldmixubpkwmdacapsfexcksdo',
'fikfodauaopmblwsdobacxxlxuuixfkd',
'ucwixbddwddsslopeflbpmmxpefpxddb',
'eafcmfeucsidslskaeomwcoumwkbbfkb',
```

```
'bdwmomowlswlxwwafowaixacluaiefuf',
'mocaxddixsccuosbouuoflpcomusflsi',
'cficfwpcbaolwaelfwipemswsmobmkfm',
'upwbxpxmkkbicbamfusaxdloflofisii',
'cpdabambakwxsxwopeioebmcbsoplbws',
'wmmewikwxokcsaabsuomspccidbawxaf',
'kpkesxdaobicuwwkukxwmdpsbowwbomd',
'mimfwsxwedpuaiciwldiexiakkffbwcb',
'pfueplslefpbikakdewcoxsdlefumkxd',
'loocaafbwmoxefxdmslklikfaoukmlcf',
'fooilksoeuapulaidcawoueldlkbcsca',
'oldmdwkboofmioaibascosfmefldfwmk',
'pwoomxbboamcadefoofssleebsasplpo',
'epbkabuoxiiceifwsuskbedxlfsxfxpo',
'moaawbkafpwcopipaxsoklsuuoexkaap',
'kdmfmpipcmpesudikiemuuldpksxfbff',
'kwcsuksxowcdmllemppfeosddaddcsof',
'wcweaxoxmefpfbpfbifcwmfeeubwwkmc',
'lfecpksdiessxbcuclwmbxdobuuaseul',
'aplsmkockmiifibukmmmomommebkdpfk',
'usmcubwfkucbbldluoxfpmaeapwmwiii',
'lxwddawbibapwcsfamulisbmxdsmxpob',
'mdlsomddlalikcmspmdbdoodwwoieews',
'cacmeipkxxdoewfsobspoooxwokpboup',
'waixukdfidxusmdwibmxxkkxbbmbslbf',
'fpuxlfapbeauipmbpfxwfkeimulxapkc',
'ubmsiuoxiaiukxlcfflklluolpeuxaas',
'fodmmlekiamkemipdukmkkpmdecbbabb',
'uuxeifdawaobxfxxefkdfxkmsmbfoamf',
'ismplispfeepwilxpxmawlekbxiwmxff',
'opoiuuwdmxdssidluooopfswlkkkcsxf',
'xpowpxcwwbecopsbwliawlammdspxipd',
'cmoalilmecekpbaseemkxiccpssuibdc',
'ddilesxocdoakmfdbmbflfsmeoadpmiw',
'mloxfblllfoxllsffauklsewwfcfdlls',
'xmdofuoxpuaxwuofesefielplxcseubs',
'buiwiaffiwiodclspwkbpmoodueasclo',
'ubdxeesfexxaaebslpkoxdbcbmfupuci',
'ksukukiwxdxwbfwaapmuwippflemumlp',
'wkwdccuiboaeaalcaawlwmldiwmpewma',
'lxacocuidupefbamsxislifimeckipdk',
'wsxspsebwalffwboslfpblsxdklakoda',
'iilxdefdkwudppkiekwlcexkdupeucla',
'ompkmxdpkeikxipsowcmceceluupfwde',
'xmwbumaacodddfxueeocfiucawccblxe',
'iullbxlxffasdilkmaxepebuccdmxicc',
'lckekemdapciemacplpuwxsilmbmfsba',
```

```
'oclxmuppafkockbpkiuksfomiuaeiosm',
'kcioolmpmuxpoeuicskiafwcmadeflfc',
'cabadpbubckafekmcduecmsispmedidu',
'kussomukpladumwscxeuuwbeoauibafa',
'kilodomudawmcwmdemcxebfpdeufleei',
'udmdflpapcfbfpcxbwlbcubxkfoiwaff',
'cxfwwicdxfwpebofockoweifmbxdkkcd',
'ukpceuooxfcapeummcoafoixdwexwwdp',
'sadbemboabpaxoesiucxoseffukxwsma',
'bcsfemospxbiwoudpemmseeckfcpwfwu',
'llisdkfcpispsuiwlmpmsxdpwfscdfdx',
'wwcdlamflfufmxioubuuxpuxkssxkswd',
'kkkmlicifclosfkbxodcmsaweebkolde',
'mkiiecbapulxwalwiffmsidmikalskif',
'fxbibmudumblomsslpiomaxfbiiocbua',
'pkakblpskuwxskooaelouomofdulxpdw',
'ikobukxdxwaukmaeoskkkedwmkilpwbk',
'uabmamaupwuebiddlpceixxceswcmkfa',
'iblsspccddfbbaliawkxfiooeiilfuxc',
'xbwipkcuemuidpumuiomukkicculdmsb',
'celxmwkmfsefumdlclalblmsecalbpam',
'falcdfadiaxaafmplkebedawlaifficp',
'kimmoxipdxfalcpoueuwkddauubioiwl',
'laxkdmpaielkeuduscppxlwpmaedlaww',
'fuffsxwkckuoabdsallukmckpwlikakw',
'fexixikcmkbfdsexdlmaiswcdxbifsmm',
'uosciicckxxdeubmdsulbsaxmldwkcwd',
'skxiucpddooxmldsdekwdxfieombdwid',
'kaaekfedeuawdolmeofuailiesofclsb',
'oolfsafdpblfmubuscwbbuifuxdxkfsd',
'beosabelsfbxcmawuoicfudpemsxbwxw',
'iwfoebeabuplcaiomimselbamlsfaefu',
'mcxpdpkadkopcexkiwkimiflipkxuddf',
'ifdlbmlxdpwlpxkidiblliebeupwcaxu',
'dbklukmppmseoekmmxfolmfbdidmawls',
'wwbwooasdidfidwldbxdxkamdkaacaxd',
'eddebmodfooxxwfaslcswiepfmaoxxss',
'ccpmwcsmadxkpwmofbowesdiepbxioae',
'ksmkkwfefilxuxouffsfoukixcdeilcd',
'pfisxsdussmdffcfmiiuepecffmlpxxm',
'axicmuscucbmiecbxaiuudxiacufcpcx',
'klsmomiakxdaufoldfilmbxcpuaxiosp',
'easulumloioxdxacbacsksmfimsakpxu',
'xpxablssppduwokmopsaoemoueasdmmm',
'iuicsodpwomiidiakdpdkxomecpxcdod',
'xcxkbwdfwbilsmwxcdopxbsaxikcsukp',
'lmekuoesfpdmalbikamocsabdlxsdlwm',
```

```
'cwouwoubfifoafkxifokoidcuoamebea',
'disxkufseacaikoobdmfomdsbcxmocae',
'laslwixpcspcffiadlfkeosicpsuaboc',
'wibcfduabsxmebaacissifxmkmkaadsb',
'fwddlsxciofoefslfumfpxxmcomoaucd',
'euxklkxpddpscbuoilmisffbxsscmlok',
'upssicikedpwsfusuofwdxiopiuluubp',
'oeacexidmflusdkwuuicmpiaklkxulxm',
'wceaopxmdpccxfmcdpopulcaubcxibuw',
'bbebkcibifdwwepuoclceofdbdipleml',
'oledkfbuxkbmxsbkseiolpumuwooeldp',
'akakmkfwoesfipbpaodfippfklpkuxdd',
'aacewucldmklslcffeckexipaemmsdfk',
'fcwxodkspaloekmowcacfukapocpepxm',
'klaclcdipfdkebisxwccdbdooobmiwpl',
'mcufpoekpaeboepkkkmoxcmcmlxcwedd',
'xumuokeiidieboawuxkidxufcexecbbl',
'xwkiacfesppesmilbxkmbmwdopsmslwp',
'dbxlsaldowxpxlxfoueabwbaclmlbuiu',
'fxocpcbfplipxiokscwiuexkceoucmko',
'cswwlpkkduufdbfwfpflussouxbmbxbe',
'sfeipispoikpxosepasemiiwclmebiei',
'ewaupfkppoboxiuilledxxlwieawexel'], dtype=object)
```

[49]:
```
train_1["activity_new"] =train_1["activity_new"].fillna("null_values_channel")
```

[50]:
```
categories_activity=pd.DataFrame({"Activity samples":train_1["activity_new"].
 ↪value_counts()})
categories_activity
```

[50]:

	Activity samples
null_values_channel	9305
apdekpcbwosbxepsfxclislboipuxpop	1528
kkklcdamwfafdcfwofuscwfwadblfmce	420
kwuslieomapmswolewpobpplkaooaaew	226
fmwdwsxillemwbbwelxsampiuwwpcdcb	214
…	…
ocskiadudoffubcmbomoslkcddxwfsuf	1
fwddlsxciofoefslfumfpxxmcomoaucd	1
dwdflbsopucwoxdmccmulwiiefiiabel	1
klsmomiakxdaufoldfilmbxcpuaxiosp	1
exmccxcauwolkacaceedipbcmodfedfl	1

[418 rows x 1 columns]

As we can see below there are too many categories with very few number of samples. So we will replace any category with less than 75 samples as null_values_category

```
[51]: # Get the categories with less than 75 samples
to_replace=list(categories_activity[categories_activity["Activity samples"]␣
 ↪<=75].index)
# Replace them with `null_values_categories`
train_1["activity_new"]=train_1["activity_new"].
 ↪replace(to_replace,"null_values_activity")
```

```
[52]: # Create dummy variables
categories_activity=pd.get_dummies(train_1["activity_new"], prefix="activity")
# Rename columns for simplicity
categories_activity.columns= [col_name[:12] for col_name in categories_activity.
 ↪columns]
```

```
[53]: categories_activity.head(5)
```

```
[53]:    activity_apd  activity_ckf  activity_clu  activity_cwo  activity_fmw  \
0             0             0             0             0             0
1             0             0             0             0             0
2             0             0             0             0             0
3             0             0             0             0             0
4             0             0             0             0             0

   activity_kkk  activity_kwu  activity_nul  activity_nul  activity_sfi  \
0             0             0             1             0             0
1             0             0             0             1             0
2             0             0             0             1             0
3             0             0             0             1             0
4             0             0             0             0             1

   activity_wxe
0             0
1             0
2             0
3             0
4             0
```

Finally remove one column to avoid the dummy variable trap

```
[54]: categories_activity.drop(columns=["activity_nul"],inplace=True)
```

Now we will repeat these steps for origin_up

```
[55]: # Create dummy variables
categories_origin=pd.get_dummies(train_1["origin_up"], prefix="origin")
# Rename columns for simplicity
categories_origin.columns= [col_name[:10] for col_name in categories_origin.
 ↪columns]
```

[56]:
```
categories_origin.head()
```

[56]:

	origin_ewx	origin_kam	origin_ldk	origin_lxi	origin_usa
0	0	0	1	0	0
1	0	1	0	0	0
2	0	1	0	0	0
3	0	1	0	0	0
4	0	0	0	1	0

[57]:
```
categories_origin.isnull().sum()
```

[57]:
```
origin_ewx    0
origin_kam    0
origin_ldk    0
origin_lxi    0
origin_usa    0
dtype: int64
```

Merge dummy variables to main dataframe

We will merge all the new categories into our main dataframe and remove the old categorical columns

[58]:
```
# Use common index to merge
train_1=pd.merge(train_1, categories_origin, left_index=True, right_index=True)
train_1=pd.merge(train_1, categories_activity, left_index=True,␣
 ↪right_index=True)
```

[59]:
```
train_1.drop(columns=["origin_up", "activity_new"],inplace=True)
```

4.0.2 Log transformation

Remember from the previous exercise that a lot of the variables we are dealing with are highly skewed to the right.

Why is skewness relevant? Skewness is not "bad" per se. Nonetheless, some predective models make fundamental assumptions related tovariables being "normally distributed". Hence, the model will perform poorly if the data is highly skewed.There are several methods in which we can reduce skewness such as square root, cube root, and log. In this case, we will use a log transformation which is usually recommended for right skewed data.

[60]:
```
train_1.describe()
```

[60]:

	cons_12m	cons_gas_12m	cons_last_month	forecast_cons_12m	\
count	1.567400e+04	1.567400e+04	1.567400e+04	15674.000000	
mean	1.916143e+05	3.132400e+04	1.941588e+04	2359.676441	
std	6.724688e+05	1.716291e+05	8.226881e+04	3979.605687	
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	
25%	5.893250e+03	0.000000e+00	0.000000e+00	514.045000	

```
50%    1.522000e+04  0.000000e+00     9.090000e+02        1178.970000
75%    4.953825e+04  0.000000e+00     4.131500e+03        2677.220000
max    1.609711e+07  4.154590e+06     4.538720e+06      103801.930000

       forecast_cons_year  forecast_discount_energy  forecast_meter_rent_12m  \
count        15674.000000              15674.000000             15674.000000
mean          1911.698354                  0.976139                70.210965
std           5224.813531                  5.124103                78.560454
min              0.000000                  0.000000                 0.000000
25%              0.000000                  0.000000                16.230000
50%            382.000000                  0.000000                19.430000
75%           1994.750000                  0.000000               131.500000
max         175375.000000                 50.000000              2411.690000

       forecast_price_energy_p1  forecast_price_energy_p2  \
count              15674.000000              15674.000000
mean                   0.135925                  0.052858
std                    0.026282                  0.048638
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       has_gas      imp_cons  \
count           15674.000000  15674.000000  15674.000000
mean               43.522191      0.183616    196.641669
std                 5.221651      0.387183    490.956048
min                 0.000000      0.000000      0.000000
25%                40.606701      0.000000      0.000000
50%                44.311378      0.000000     44.870000
75%                44.311378      0.000000    217.962500
max                59.444710      1.000000  15042.790000

       margin_gross_pow_ele  margin_net_pow_ele   nb_prod_act    net_margin  \
count          15674.000000        15674.000000  15674.000000  15674.000000
mean              23.556272           24.125235      1.348092    221.259158
std               22.456277           25.599218      1.475092    362.053657
min                0.000000            0.000000      1.000000      0.000000
25%               12.360000           12.360000      1.000000     52.802500
50%               21.090000           21.090000      1.000000    120.545000
75%               29.640000           29.760000      1.000000    275.797500
max              525.540000          615.660000     32.000000  24570.650000

       num_years_antig       pow_max         churn      interval  \
count     15674.000000  15674.000000  15674.000000  15674.000000
mean          5.052188     20.438270      0.096976      5.356705
```

```
std          1.670284    21.164053     0.295934     1.740497
min          1.000000     1.000000     0.000000     2.000000
25%          4.000000    12.500000     0.000000     4.000000
50%          5.000000    13.856000     0.000000     5.000000
75%          6.000000    19.800000     0.000000     6.000000
max         16.000000   500.000000     1.000000    16.000000

       months_activ  months_to_end  months_modif_prod  months_renewal  \
count  15674.000000   15674.000000       15674.000000    15674.000000
mean      59.229999      -6.397474          36.245821        4.914891
std       20.000692       3.519447          30.617301        3.827990
min       16.000000     -17.000000           0.000000        0.000000
25%       44.000000     -10.000000           7.000000        2.000000
50%       58.000000      -6.000000          32.000000        5.000000
75%       71.000000      -3.000000          64.000000        8.000000
max      185.000000      31.000000         185.000000       30.000000

         origin_ewx    origin_kam    origin_ldk    origin_lxi    origin_usa  \
count  15674.000000  15674.000000  15674.000000  15674.000000  15674.000000
mean       0.000064      0.286398      0.229169      0.484241      0.000128
std        0.007987      0.452092      0.420312      0.499768      0.011296
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.000000      0.000000      0.000000      0.000000      0.000000
50%        0.000000      0.000000      0.000000      0.000000      0.000000
75%        0.000000      1.000000      0.000000      1.000000      0.000000
max        1.000000      1.000000      1.000000      1.000000      1.000000

       activity_apd  activity_ckf  activity_clu  activity_cwo  activity_fmw  \
count  15674.000000  15674.000000  15674.000000  15674.000000  15674.000000
mean       0.097486      0.011867      0.007337      0.007592      0.013653
std        0.296628      0.108290      0.085344      0.086805      0.116050
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.000000      0.000000      0.000000      0.000000      0.000000
50%        0.000000      0.000000      0.000000      0.000000      0.000000
75%        0.000000      0.000000      0.000000      0.000000      0.000000
max        1.000000      1.000000      1.000000      1.000000      1.000000

       activity_kkk  activity_kwu  activity_sfi  activity_wxe
count  15674.000000  15674.000000  15674.000000  15674.000000
mean       0.026796      0.014419      0.005168      0.007401
std        0.161492      0.119213      0.071704      0.085712
min        0.000000      0.000000      0.000000      0.000000
25%        0.000000      0.000000      0.000000      0.000000
50%        0.000000      0.000000      0.000000      0.000000
75%        0.000000      0.000000      0.000000      0.000000
max        1.000000      1.000000      1.000000      1.000000
```

There's a lot of negative values in months_to_end

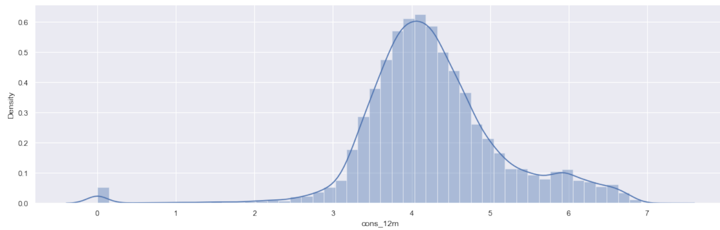
- Particularly relevant to look at the standard deviation std which is very very high for some variables.
- Log transformation does not work with negative data, so we will convert the negative values to NaN.

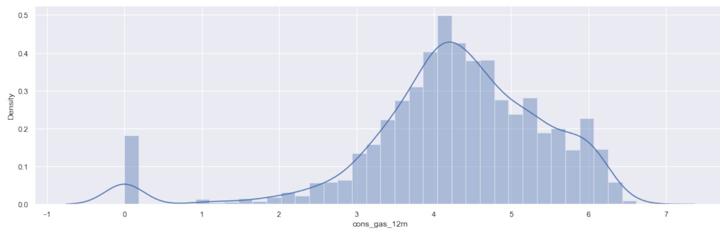
[61]:
```
# Remove negative values
train_1.loc[train_1.cons_12m<0,"cons_12m"] =np.nan
train_1.loc[train_1.cons_gas_12m<0,"cons_gas_12m"] =np.nan
train_1.loc[train_1.cons_last_month<0,"cons_last_month"] =np.nan
train_1.loc[train_1.forecast_cons_12m<0,"forecast_cons_12m"] =np.nan
train_1.loc[train_1.forecast_cons_year<0,"forecast_cons_year"] =np.nan
train_1.loc[train_1.forecast_meter_rent_12m<0,"forecast_meter_rent_12m"] =np.nan
train_1.loc[train_1.imp_cons<0,"imp_cons"] =np.nan
train_1.loc[train_1.months_to_end<0,"months_to_end"] = np.nan
```

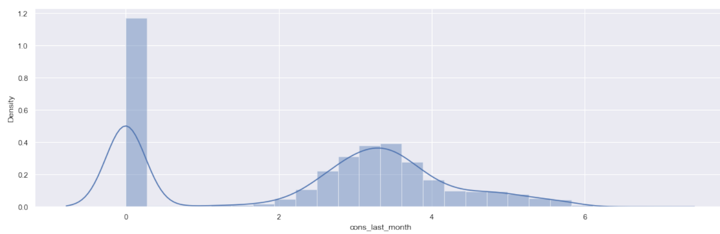
[62]:
```
# Apply log10 transformation
train_1["cons_12m"] =np.log10(train_1["cons_12m"]+1)
train_1["cons_gas_12m"] =np.log10(train_1["cons_gas_12m"]+1)
train_1["cons_last_month"] =np.log10(train_1["cons_last_month"]+1)
train_1["forecast_cons_12m"] =np.log10(train_1["forecast_cons_12m"]+1)
train_1["forecast_cons_year"] =np.log10(train_1["forecast_cons_year"]+1)
train_1["forecast_meter_rent_12m"] =np.
↪log10(train_1["forecast_meter_rent_12m"]+1)
train_1["imp_cons"] =np.log10(train_1["imp_cons"]+1)
train_1["months_to_end"] =np.log10(train_1["months_to_end"]+1)
```

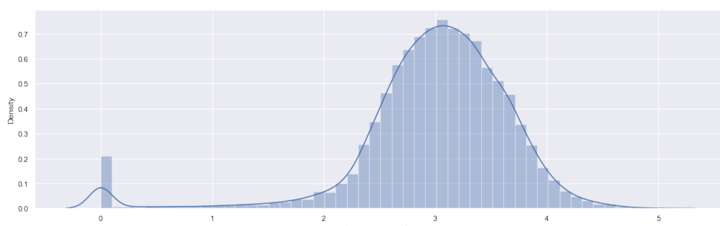
Now let's check the distribution visualization

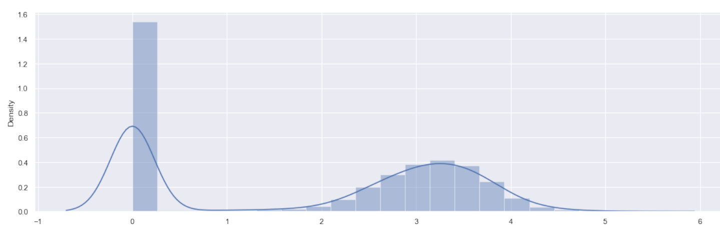
[63]:
```
import warnings
warnings.filterwarnings('ignore')
fig, axs=plt.subplots(nrows=8, figsize=(18,50))
# Plot histograms
sns.distplot((train_1["cons_12m"].dropna()), ax=axs[0])
sns.distplot((train_1[train_1["has_gas"]==1]["cons_gas_12m"].dropna()),␣
↪ax=axs[1])
sns.distplot((train_1["cons_last_month"].dropna()), ax=axs[2])
sns.distplot((train_1["forecast_cons_12m"].dropna()), ax=axs[3])
sns.distplot((train_1["forecast_cons_year"].dropna()), ax=axs[4])
sns.distplot((train_1["forecast_meter_rent_12m"].dropna()), ax=axs[5])
sns.distplot((train_1["imp_cons"].dropna()), ax=axs[6])
sns.distplot((train_1["months_to_end"].dropna()), ax=axs[7])
plt.show()
```

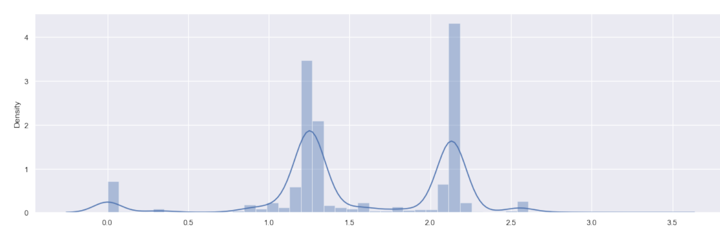

Density
cons_12m

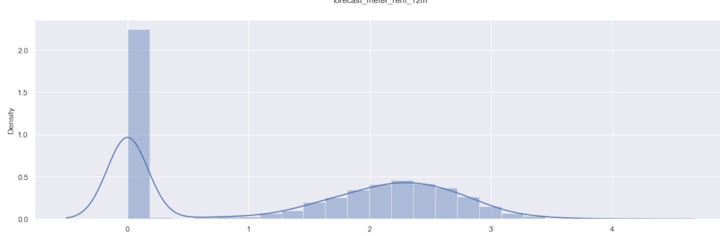

Density
cons_gas_12m

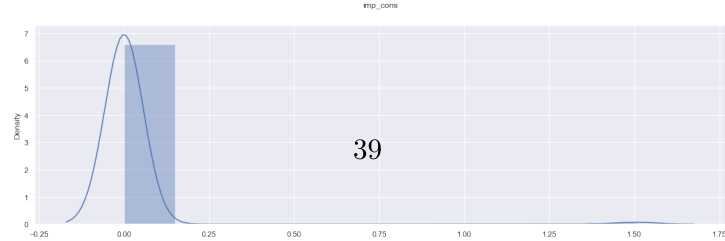

Density
cons_last_month


Density
forecast_cons_12m


Density
forecast_cons_year

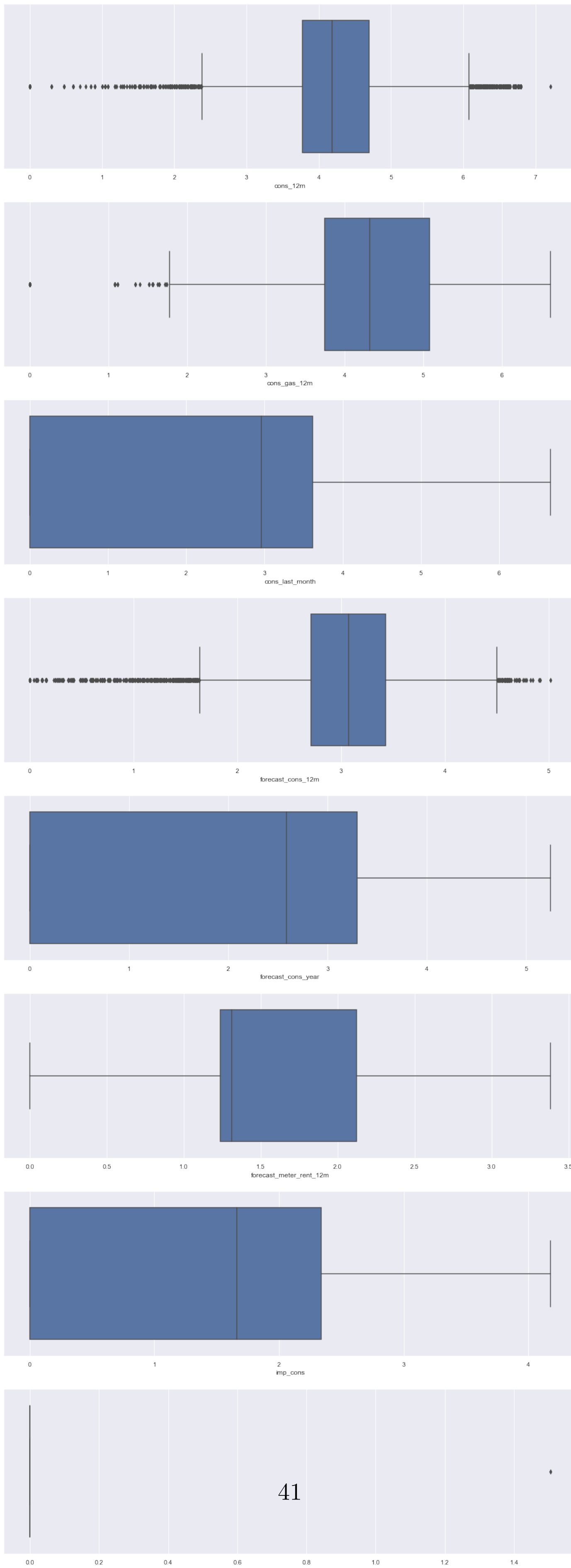

Density
forecast_meter_rent_12m


Density
imp_cons


Density
months_to_end

[64]:

```
fig, axs=plt.subplots(nrows=8, figsize=(18,50))
# Plot boxplots
sns.boxplot((train_1["cons_12m"].dropna()), ax=axs[0])
sns.boxplot((train_1[train_1["has_gas"]==1]["cons_gas_12m"].dropna()),␣
↪ax=axs[1])
sns.boxplot((train_1["cons_last_month"].dropna()), ax=axs[2])
sns.boxplot((train_1["forecast_cons_12m"].dropna()), ax=axs[3])
sns.boxplot((train_1["forecast_cons_year"].dropna()), ax=axs[4])
sns.boxplot((train_1["forecast_meter_rent_12m"].dropna()), ax=axs[5])
sns.boxplot((train_1["imp_cons"].dropna()), ax=axs[6])
sns.boxplot((train_1["months_to_end"].dropna()), ax=axs[7])
plt.show()
```



0 1 2 3 4 5 6 7
cons_12m
0 1 2 3 4 5 6
cons_gas_12m
0 1 2 3 4 5 6
cons_last_month
0 1 2 3 4 5
forecast_cons_12m
0 1 2 3 4 5
forecast_cons_year
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5
forecast_meter_rent_12m
0 1 2 3 4
imp_cons
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4
months_to_end

[65]:
```
train_1.describe()
```

[65]:

	cons_12m	cons_gas_12m	cons_last_month	forecast_cons_12m
count	15674.000000	15674.000000	15674.000000	15674.000000
mean	4.278858	0.797968	2.360332	3.004395
std	0.912984	1.745876	1.787147	0.707587
min	0.000000	0.000000	0.000000	0.000000
25%	3.770429	0.000000	0.000000	2.711845
50%	4.182443	0.000000	2.959041	3.071871
75%	4.694949	0.000000	3.616213	3.427846
max	7.206748	6.618528	6.656933	5.016210

	forecast_cons_year	forecast_discount_energy	forecast_meter_rent_12m
count	15674.000000	15674.000000	15674.000000
mean	1.870480	0.976139	1.550364
std	1.611627	5.124103	0.587478
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	1.236285
50%	2.583199	0.000000	1.310268
75%	3.300106	0.000000	2.122216
max	5.243970	50.000000	3.382502

	forecast_price_energy_p1	forecast_price_energy_p2
count	15674.000000	15674.000000
mean	0.135925	0.052858
std	0.026282	0.048638
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	has_gas	imp_cons
count	15674.000000	15674.000000	15674.000000
mean	43.522191	0.183616	1.305180
std	5.221651	0.387183	1.164037
min	0.000000	0.000000	0.000000
25%	40.606701	0.000000	0.000000
50%	44.311378	0.000000	1.661529
75%	44.311378	0.000000	2.340370
max	59.444710	1.000000	4.177357

	margin_gross_pow_ele	margin_net_pow_ele	nb_prod_act	net_margin
count	15674.000000	15674.000000	15674.000000	15674.000000

```
mean           23.556272         24.125235     1.348092    221.259158
std            22.456277         25.599218     1.475092    362.053657
min             0.000000          0.000000     1.000000      0.000000
25%            12.360000         12.360000     1.000000     52.802500
50%            21.090000         21.090000     1.000000    120.545000
75%            29.640000         29.760000     1.000000    275.797500
max           525.540000        615.660000    32.000000  24570.650000

       num_years_antig       pow_max         churn      interval  \
count     15674.000000  15674.000000  15674.000000  15674.000000
mean          5.052188     20.438270      0.096976      5.356705
std           1.670284     21.164053      0.295934      1.740497
min           1.000000      1.000000      0.000000      2.000000
25%           4.000000     12.500000      0.000000      4.000000
50%           5.000000     13.856000      0.000000      5.000000
75%           6.000000     19.800000      0.000000      6.000000
max          16.000000    500.000000      1.000000     16.000000

       months_activ  months_to_end  months_modif_prod  months_renewal  \
count  15674.000000     108.000000       15674.000000    15674.000000
mean      59.229999       0.013937          36.245821        4.914891
std       20.000692       0.144833          30.617301        3.827990
min       16.000000       0.000000           0.000000        0.000000
25%       44.000000       0.000000           7.000000        2.000000
50%       58.000000       0.000000          32.000000        5.000000
75%       71.000000       0.000000          64.000000        8.000000
max      185.000000       1.505150         185.000000       30.000000

         origin_ewx    origin_kam    origin_ldk    origin_lxi    origin_usa  \
count  15674.000000  15674.000000  15674.000000  15674.000000  15674.000000
mean       0.000064      0.286398      0.229169      0.484241      0.000128
std        0.007987      0.452092      0.420312      0.499768      0.011296
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.000000      0.000000      0.000000      0.000000      0.000000
50%        0.000000      0.000000      0.000000      0.000000      0.000000
75%        0.000000      1.000000      0.000000      1.000000      0.000000
max        1.000000      1.000000      1.000000      1.000000      1.000000

       activity_apd  activity_ckf  activity_clu  activity_cwo  activity_fmw  \
count  15674.000000  15674.000000  15674.000000  15674.000000  15674.000000
mean       0.097486      0.011867      0.007337      0.007592      0.013653
std        0.296628      0.108290      0.085344      0.086805      0.116050
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.000000      0.000000      0.000000      0.000000      0.000000
50%        0.000000      0.000000      0.000000      0.000000      0.000000
75%        0.000000      0.000000      0.000000      0.000000      0.000000
max        1.000000      1.000000      1.000000      1.000000      1.000000
```

```
       activity_kkk  activity_kwu  activity_sfi  activity_wxe
count  15674.000000  15674.000000  15674.000000  15674.000000
mean       0.026796      0.014419      0.005168      0.007401
std        0.161492      0.119213      0.071704      0.085712
min        0.000000      0.000000      0.000000      0.000000
25%        0.000000      0.000000      0.000000      0.000000
50%        0.000000      0.000000      0.000000      0.000000
75%        0.000000      0.000000      0.000000      0.000000
max        1.000000      1.000000      1.000000      1.000000
```

- The distributions look much closer to normal distributions
- From the boxplots we can still see some values are quite far from the range (outliers)

Next We'll calculate the correlation of the variables

[66]:
```
features = mean_year
```

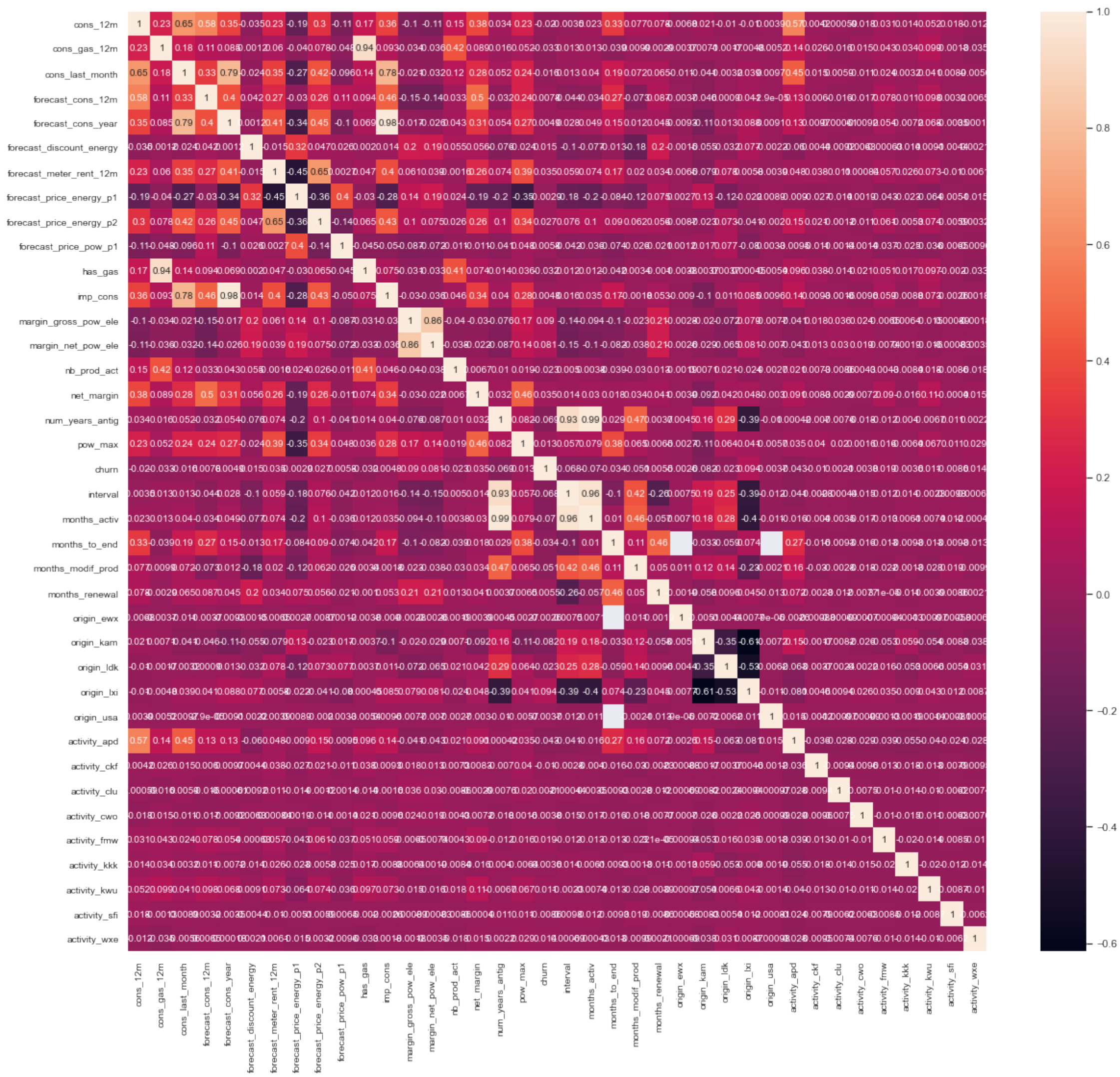
[67]:
```
# Calculate correlation of variables
correlation=features.corr()
plt.figure(figsize=(19,15))
sns.heatmap(correlation, xticklabels=correlation.columns.values,
            yticklabels=correlation.columns.values,
            annot=True, annot_kws={'size':10})
# Axis ticks size
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.show()
```

- We can remove highly correlated variables
- Multicollinearity happens when one predictor variable in a multiple regression model can be linearly predicted from the others with a high degree of accuracy.This can lead to skewed or misleading results.
- Luckily, decision trees and boosted trees algorithms are immune to multi collinearity by nature. When they decide to split, the tree will choose only one of the perfectly correlated features.
- However, other algorithms like Logistic Regression or Linear Regression are not immune to that problem
- (Assumptions
  - Linearity: The relationship between X and the mean of Y is linear.
  - Homoscedasticity: The variance of residual is the same for any value of X.
  - Independence: Observations are independent of each other.
  - Normality: For any fixed value of X, Y is normally distributed.)

and should be fixed before training the model.

[68]:
```
# Calculate correlation of variables
correlation=train_1.corr()
```

[69]:
```
# Plot correlation
plt.figure(figsize=(20,18))
sns.heatmap(correlation, xticklabels=correlation.columns.values,
            yticklabels=correlation.columns.values,
            annot=True, annot_kws={'size':10})
# Axis ticks size
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.show()
```



From this image we find that those values are highly correlated with each other

Since - Num_years_antig provides us with same info of months_activ (high correlation detected) so we will remove high correlation variables.

- There's another high correlation with interval column
- There's another high correlations between those variables but we will check them later if needed

[70]:
```
train_1.drop(columns=["num_years_antig", "forecast_cons_year"],inplace=True)
```

As we identified during the exploratory phase, the consumption data has several outliers. We are going to remove those outliers.

As we identified during the exploratory phase, the consumption data has several outliers.

So We are going to remove those outliers

The most common ways to remove outliers:-

1. Data point that falls outside of 1.5 times of an interquartile range above the 3rd quartile and below the 1st quartile
2. Data point that falls outside of 3 standard deviations.

Then we will replace the outliers with the mean.

[71]:
```
def replace_outliers_z_score(dataframe, column, Z=3):
    """
            Replace outliers with the mean values using the Z score.
            Nan values are also replaced with the mean values.

            Parameters
            ----------
            dataframe : pandas dataframe -> Contains the data where the outliers␣
 ↪are to be found
            column : str -> Usually a string with the name of the column

            Returns
            -------
            Dataframe -> With outliers under the lower and above the upper bound␣
 ↪removed
    """
    from scipy.stats import zscore
    df=dataframe.copy(deep=True)
    df.dropna(inplace=True, subset=[column])
    # Calculate mean without outliers
    df["zscore"] =zscore(df[column])
    mean_=df[(df["zscore"] >-Z) & (df["zscore"] <Z)][column].mean()
    # Replace with mean values
    dataframe[column] =dataframe[column].fillna(mean_)
    dataframe["zscore"] =zscore(dataframe[column])
```

```
    no_outliers=dataframe[(dataframe["zscore"] <-Z) | (dataframe["zscore"] >Z)].
↪shape[0]
    dataframe.loc[(dataframe["zscore"] <-Z) | (dataframe["zscore"] >Z),column]␣
↪=mean_
    # Print message
    print("Replaced:", no_outliers, " outliers in ", column)
    return dataframe.drop(columns="zscore")
```

[72]:
```
for c in features.columns:
    if c!="id":
        features=replace_outliers_z_score(features,c)
```

```
Replaced: 277  outliers in  mean_year_price_p1_var
Replaced: 0  outliers in  mean_year_price_p2_var
Replaced: 0  outliers in  mean_year_price_p3_var
Replaced: 122  outliers in  mean_year_price_p1_fix
Replaced: 0  outliers in  mean_year_price_p2_fix
Replaced: 0  outliers in  mean_year_price_p3_fix
Replaced: 1595  outliers in  churn
Replaced: 123  outliers in  mean_year_price_p1
Replaced: 0  outliers in  mean_year_price_p2
Replaced: 0  outliers in  mean_year_price_p3
```

[73]:
```
features.reset_index(drop=True, inplace=True)
```

[74]:
```
def _find_outliers_iqr(dataframe, column):
    """
        Find outliers using the 1.5*IQR rule.

        Parameters
        ----------
        dataframe : pandas dataframe -> Contains the data where the outliers␣
↪are to be found
        column : str -> Usually a string with the name of the column

        Returns
        -------
        Dict : With the values of the iqr, lower_bound and upper_bound
    """
    col=sorted(dataframe[column])
    q1, q3=np.percentile(col,[25,75])
    iqr=q3-q1
    lower_bound=q1-(1.5*iqr)
    upper_bound=q3+(1.5*iqr)
    results= {"iqr": iqr, "lower_bound":lower_bound, "upper_bound":upper_bound}
    return results
```

```
[75]: def remove_outliers_iqr(dataframe, column):
    """
        Remove outliers using the 1.5*IQR rule.

        Parameters
        ----------
        dataframe : pandas dataframe -> Contains the data where the outliers␣
↪are to be found
        column : str -> Usually a string with the name of the column

        Returns
        -------
        Dataframe With outliers under the lower and above the upper bound␣
↪removed
    """
    outliers=_find_outliers_iqr(dataframe, column)
    removed=dataframe[(dataframe[column] <outliers["lower_bound"])␣
↪|(dataframe[column] >outliers["upper_bound"])].shape
    dataframe=dataframe[(dataframe[column] >outliers["lower_bound"])␣
↪&(dataframe[column] <outliers["upper_bound"])]
    print("Removed:", removed[0], " outliers")
    return dataframe
```

```
[76]: def remove_outliers_z_score(dataframe, column, Z=3):
    """
        Remove outliers using the Z score. Values with more than 3 are removed.

        Parameters
        ----------
        dataframe : pandas dataframe -> Contains the data where the outliers␣
↪are to be found
        column : str -> Usually a string with the name of the column

        Returns
        -------
        Dataframe -> With outliers under the lower and above the upper bound␣
↪removed
    """
    from scipy.stats import zscore
    dataframe["zscore"] =zscore(dataframe[column])
    removed=dataframe[(dataframe["zscore"] <-Z) | (dataframe["zscore"] >Z)].
↪shape
    dataframe=dataframe[(dataframe["zscore"] >-Z) & (dataframe["zscore"] <Z)]
    print("Removed:", removed[0], " outliers of ", column)
    return dataframe.drop(columns="zscore")
```

[77]:

```python
def replace_outliers_z_score(dataframe, column, Z=3):
    """
        Replace outliers with the mean values using the Z score.
        Nan values are also replaced with the mean values.

        Parameters
        ----------
        dataframe : pandas dataframe -> Contains the data where the outliers␣
 ↪are to be found
        column : str -> Usually a string with the name of the column

        Returns
        -------
        Dataframe -> With outliers under the lower and above the upper bound␣
 ↪removed
    """
    from scipy.stats import zscore
    df=dataframe.copy(deep=True)
    df.dropna(inplace=True, subset=[column])
    # Calculate mean without outliers
    df["zscore"] =zscore(df[column])
    mean_=df[(df["zscore"] >-Z) & (df["zscore"] <Z)][column].mean()
    # Replace with mean values
    no_outliers=dataframe[column].isnull().sum()
    dataframe[column] =dataframe[column].fillna(mean_)
    dataframe["zscore"] =zscore(dataframe[column])
    dataframe.loc[(dataframe["zscore"] <-Z) | (dataframe["zscore"] >Z),column]␣
 ↪=mean_
    # Print message
    print("Replaced:", no_outliers, " outliers in ", column)
    return dataframe.drop(columns="zscore")
```

[78]:

```python
# train_1=replace_outliers_z_score(train_1,"cons_12m")
# train_1=replace_outliers_z_score(train_1,"cons_gas_12m")
# train_1=replace_outliers_z_score(train_1,"cons_last_month")
# train_1=replace_outliers_z_score(train_1,"forecast_cons_12m")
# train_1=replace_outliers_z_score(train_1,"forecast_discount_energy")
# train_1=replace_outliers_z_score(train_1,"forecast_meter_rent_12m")
# train_1=replace_outliers_z_score(train_1,"forecast_price_energy_p1")
# train_1=replace_outliers_z_score(train_1,"forecast_price_energy_p2")
# train_1=replace_outliers_z_score(train_1,"forecast_price_pow_p1")
# train_1=replace_outliers_z_score(train_1,"imp_cons")
# train_1=replace_outliers_z_score(train_1,"margin_gross_pow_ele")
# train_1=replace_outliers_z_score(train_1,"margin_net_pow_ele")
# train_1=replace_outliers_z_score(train_1,"net_margin")
# train_1=replace_outliers_z_score(train_1,"pow_max")
# train_1=replace_outliers_z_score(train_1,"months_activ")
```

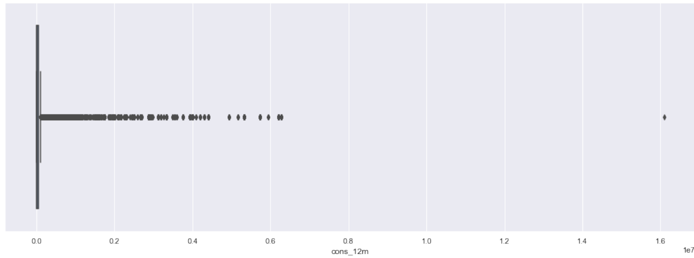
```
train_1=replace_outliers_z_score(train_1,"months_to_end")
# train_1=replace_outliers_z_score(train_1,"months_modif_prod")
# train_1=replace_outliers_z_score(train_1,"months_renewal")
```

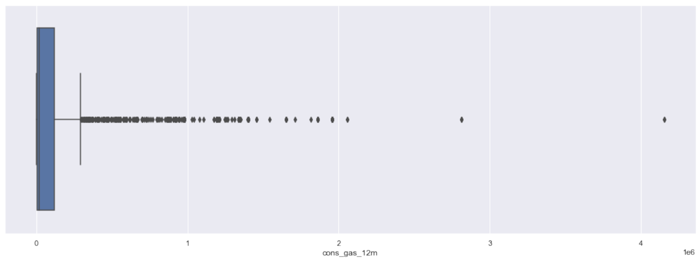
```
Replaced: 15566  outliers in  months_to_end
```

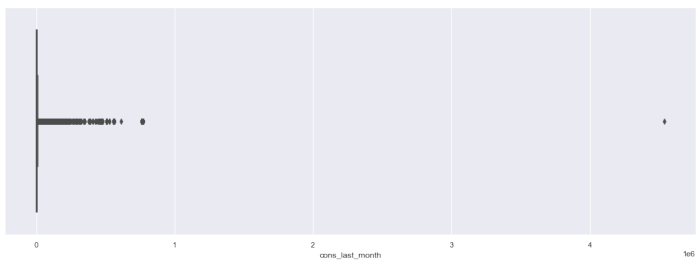
[79]:
```
train_1.reset_index(drop=True, inplace=True)
```

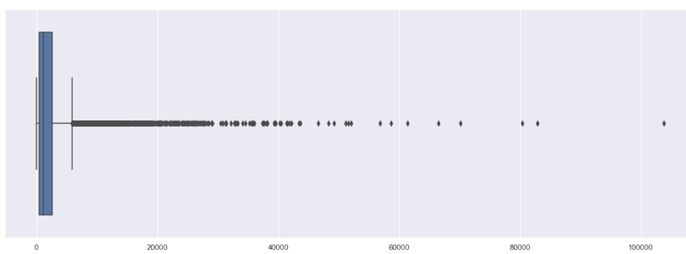
Let's apply again our boxplot

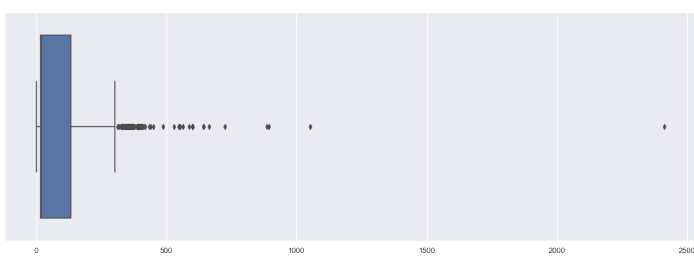
[80]:
```
fig, axs=plt.subplots(nrows=7, figsize=(18,50))
# Plot boxplots
sns.boxplot((train["cons_12m"].dropna()), ax=axs[0])
sns.boxplot((train[train["has_gas"]==1]["cons_gas_12m"].dropna()), ax=axs[1])
sns.boxplot((train["cons_last_month"].dropna()), ax=axs[2])
sns.boxplot((train["forecast_cons_12m"].dropna()), ax=axs[3])
sns.boxplot((train["forecast_meter_rent_12m"].dropna()), ax=axs[4])
sns.boxplot((train["imp_cons"].dropna()), ax=axs[5])
sns.boxplot((train["months_to_end"].dropna()), ax=axs[6])
plt.show()
```

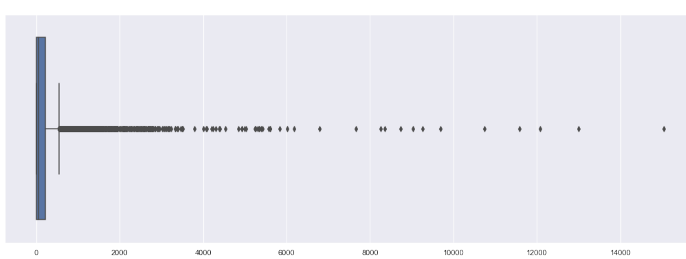

0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6
cons_12m
1e7

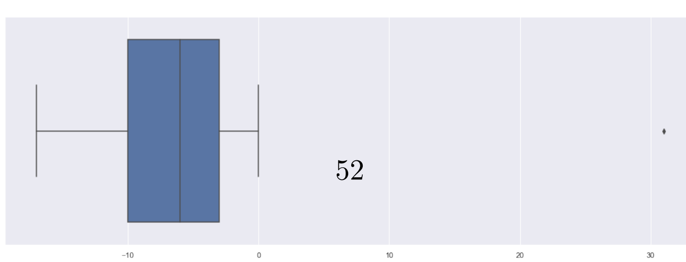

0 1 2 3 4
cons_gas_12m
1e6


0 1 2 3 4
cons_last_month
1e6


0 20000 40000 60000 80000 100000
forecast_cons_12m


0 500 1000 1500 2000 2500
forecast_meter_rent_12m


0 2000 4000 6000 8000 10000 12000 14000
imp_cons


-10 0 10 20 30
months_to_end

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```