Exploratody Data Analysis

Import Library

```
# Core library
import pandas as pd
import numpy as np
import datetime
import warnings
import os

# Visualization library
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

Loading dataset with Pandas

We will start by loading `client_data.csv` and `price_data.csv` into each dataframe to work with python.

```
# Find file path
path = os.getcwd()

# Load dataset
client_df = pd.read_csv(os.path.join(path,'client_data.csv'))
price_df = pd.read_csv(os.path.join(path,'price_data.csv'))
```

We start by looking at the first 5 rows of dataframe using `head(5)`.

```
client_df.head(5)
```

```
                                  id                    channel_sales
\
0  24011ae4ebbe3035111d65fa7c15bc57  foosdfpfkusacimwkcsosbicdxkicaua

1  d29c2c54acc38ff3c0614d0a653813dd                           MISSING

2  764c75f661154dac3a6c254cd082ea7d  foosdfpfkusacimwkcsosbicdxkicaua

3  bba03439a292a1e166f80264c16191cb  lmkebamcaaclubfxadlmueccxoimlema

4  149d57cf92fc41cf94415803a877cb4b                           MISSING


   cons_12m  cons_gas_12m  cons_last_month  date_activ    date_end  \
```

```
0         0         54946                0  2013-06-15  2016-06-15
1      4660             0                0  2009-08-21  2016-08-30
2       544             0                0  2010-04-16  2016-04-16
3      1584             0                0  2010-03-30  2016-03-30
4      4425             0              526  2010-01-13  2016-03-07

  date_modif_prod date_renewal  forecast_cons_12m  ...  has_gas
imp_cons  \
0      2015-11-01   2015-06-23               0.00  ...        t
0.00
1      2009-08-21   2015-08-31             189.95  ...        f
0.00
2      2010-04-16   2015-04-17              47.96  ...        f
0.00
3      2010-03-30   2015-03-31             240.04  ...        f
0.00
4      2010-01-13   2015-03-09             445.75  ...        f
52.32

   margin_gross_pow_ele  margin_net_pow_ele  nb_prod_act
net_margin  \
0                 25.44               25.44            2      678.99

1                 16.38               16.38            1       18.89

2                 28.60               28.60            1        6.60

3                 30.22               30.22            1       25.46

4                 44.91               44.91            1       47.98


  num_years_antig                          origin_up  pow_max  churn
0               3  lxidpiddsbxsbosboudacockeimpuepw   43.648      1
1               6  kamkkxfxxuwbdslkwifmmcsiusiuosws   13.800      0
2               6  kamkkxfxxuwbdslkwifmmcsiusiuosws   13.856      0
3               6  kamkkxfxxuwbdslkwifmmcsiusiuosws   13.200      0
4               6  kamkkxfxxuwbdslkwifmmcsiusiuosws   19.800      0

[5 rows x 26 columns]
```

```
price_df.head(5)
```

```
                                 id  price_date  price_off_peak_var  \
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
```

```
   price_peak_var  price_mid_peak_var  price_off_peak_fix
price_peak_fix  \
0             0.0                 0.0           44.266931
0.0
1             0.0                 0.0           44.266931
0.0
2             0.0                 0.0           44.266931
0.0
3             0.0                 0.0           44.266931
0.0
4             0.0                 0.0           44.266931
0.0

   price_mid_peak_fix
0                 0.0
1                 0.0
2                 0.0
3                 0.0
4                 0.0
```

Descriptive statistic of data

Data types

We will use `shape` to look at the number of column and row of dataframe. We will also using `info()` to get structure of dataframe.

```
print(client_df.shape)
client_df.info()
```

```
(14606, 26)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14606 entries, 0 to 14605
Data columns (total 26 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              14606 non-null  object
 1   channel_sales                   14606 non-null  object
 2   cons_12m                        14606 non-null  int64
 3   cons_gas_12m                    14606 non-null  int64
 4   cons_last_month                 14606 non-null  int64
 5   date_activ                      14606 non-null  object
 6   date_end                        14606 non-null  object
 7   date_modif_prod                 14606 non-null  object
 8   date_renewal                    14606 non-null  object
 9   forecast_cons_12m               14606 non-null  float64
 10  forecast_cons_year              14606 non-null  int64
 11  forecast_discount_energy        14606 non-null  float64
 12  forecast_meter_rent_12m         14606 non-null  float64
```

```
 13  forecast_price_energy_off_peak  14606 non-null  float64
 14  forecast_price_energy_peak      14606 non-null  float64
 15  forecast_price_pow_off_peak     14606 non-null  float64
 16  has_gas                         14606 non-null  object
 17  imp_cons                        14606 non-null  float64
 18  margin_gross_pow_ele            14606 non-null  float64
 19  margin_net_pow_ele              14606 non-null  float64
 20  nb_prod_act                     14606 non-null  int64
 21  net_margin                      14606 non-null  float64
 22  num_years_antig                 14606 non-null  int64
 23  origin_up                       14606 non-null  object
 24  pow_max                         14606 non-null  float64
 25  churn                           14606 non-null  int64
dtypes: float64(11), int64(7), object(8)
memory usage: 2.9+ MB
```

```
print(price_df.shape)
price_df.info()
```

```
(193002, 8)
<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_off_peak_var  193002 non-null  float64
 3   price_peak_var      193002 non-null  float64
 4   price_mid_peak_var  193002 non-null  float64
 5   price_off_peak_fix  193002 non-null  float64
 6   price_peak_fix      193002 non-null  float64
 7   price_mid_peak_fix  193002 non-null  float64
dtypes: float64(6), object(2)
memory usage: 11.8+ MB
```

Statistics

We will start looking at statistic of the dataframes using `describe()`. To make it easier to read we will only look at 2 decimal using `round(2)`.

```
client_df.describe().round(2)
```

```
         cons_12m  cons_gas_12m  cons_last_month  forecast_cons_12m  \
count    14606.00      14606.00         14606.00           14606.00
mean    159220.29      28092.38         16090.27            1868.61
std     573465.26     162973.06         64364.20            2387.57
min          0.00          0.00             0.00               0.00
25%       5674.75          0.00             0.00             495.00
50%      14115.50          0.00           792.50            1112.88
75%      40763.75          0.00          3383.00            2401.79
```

```
max    6207104.00    4154590.00        771203.00           82902.83

       forecast_cons_year  forecast_discount_energy
forecast_meter_rent_12m  \
count            14606.00                  14606.00
14606.00
mean              1399.76                      0.97
63.09
std               3247.79                      5.11
66.17
min                  0.00                      0.00
0.00
25%                  0.00                      0.00
16.18
50%                314.00                      0.00
18.80
75%               1745.75                      0.00
131.03
max             175375.00                     30.00
599.31

       forecast_price_energy_off_peak  forecast_price_energy_peak  \
count                        14606.00                    14606.00
mean                             0.14                        0.05
std                              0.02                        0.05
min                              0.00                        0.00
25%                              0.12                        0.00
50%                              0.14                        0.08
75%                              0.15                        0.10
max                              0.27                        0.20

       forecast_price_pow_off_peak  imp_cons  margin_gross_pow_ele  \
count                     14606.00  14606.00              14606.00
mean                         43.13    152.79                 24.57
std                           4.49    341.37                 20.23
min                           0.00      0.00                  0.00
25%                          40.61      0.00                 14.28
50%                          44.31     37.39                 21.64
75%                          44.31    193.98                 29.88
max                          59.27  15042.79                374.64

       margin_net_pow_ele  nb_prod_act  net_margin  num_years_antig
pow_max  \
count            14606.00     14606.00    14606.00         14606.00
14606.00
mean                24.56         1.29      189.26             5.00
18.14
std                 20.23         0.71      311.80             1.61
13.53
min                  0.00         1.00        0.00             1.00
```

```
3.30
25%                 14.28          1.00       50.71              4.00
12.50
50%                 21.64          1.00      112.53              5.00
13.86
75%                 29.88          1.00      243.10              6.00
19.17
max                374.64         32.00    24570.65             13.00
320.00

          churn
count  14606.0
mean       0.1
std        0.3
min        0.0
25%        0.0
50%        0.0
75%        0.0
max        1.0
```

```
price_df.describe().round(2)
```

```
       price_off_peak_var  price_peak_var  price_mid_peak_var  \
count           193002.00       193002.00           193002.00
mean                 0.14            0.05                0.03
std                  0.03            0.05                0.04
min                  0.00            0.00                0.00
25%                  0.13            0.00                0.00
50%                  0.15            0.09                0.00
75%                  0.15            0.10                0.07
max                  0.28            0.23                0.11

       price_off_peak_fix  price_peak_fix  price_mid_peak_fix
count           193002.00       193002.00           193002.00
mean                43.33           10.62                6.41
std                  5.41           12.84                7.77
min                  0.00            0.00                0.00
25%                 40.73            0.00                0.00
50%                 44.27            0.00                0.00
75%                 44.44           24.34               16.23
max                 59.44           36.49               17.46
```

Finding NA/NULL data

To make sure the data is clean. We will look for any NULL/NA data in the dataframe using `isna()`. We will get total of missing data using `sum()`.

```
client_df.isna().sum()
```

```
id                              0
channel_sales                   0
```

```
cons_12m                          0
cons_gas_12m                      0
cons_last_month                   0
date_activ                        0
date_end                          0
date_modif_prod                   0
date_renewal                      0
forecast_cons_12m                 0
forecast_cons_year                0
forecast_discount_energy          0
forecast_meter_rent_12m           0
forecast_price_energy_off_peak    0
forecast_price_energy_peak        0
forecast_price_pow_off_peak       0
has_gas                           0
imp_cons                          0
margin_gross_pow_ele              0
margin_net_pow_ele                0
nb_prod_act                       0
net_margin                        0
num_years_antig                   0
origin_up                         0
pow_max                           0
churn                             0
dtype: int64
```

```
price_df.isna().sum()
```

```
id                    0
price_date            0
price_off_peak_var    0
price_peak_var        0
price_mid_peak_var    0
price_off_peak_fix    0
price_peak_fix        0
price_mid_peak_fix    0
dtype: int64
```

It's appear the data is clean. Note: there are some data label as `MISSING`. Will need to describe what to do with the team.

Data visualization

First we want use a bar chart to better visualize the data. In this case the stacked bar chart is the best if we are going to look into over all churn status. As we will use them multiple time create a function will save time.

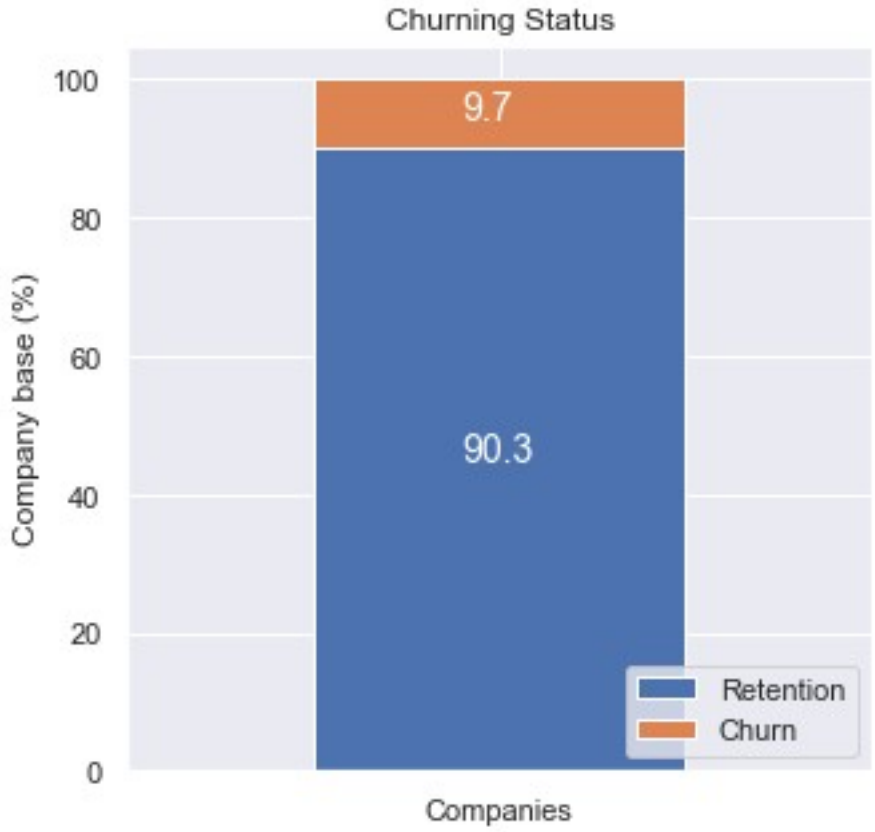
```
# Set plot style
sns.set(color_codes=True)
```

```
def plot_stacked_bars(dataframe, title_, size_=(18, 10), rot_=0,
legend_="upper right"):
    """
    Plot stacked bars with annotations
    """
    ax = dataframe.plot(
        kind="bar",
        stacked=True,
        figsize=size_,
        rot=rot_,
        title=title_
    )

    # Annotate bars
    annotate_stacked_bars(ax, textsize=14)
    # Rename legend
    plt.legend(["Retention", "Churn"], loc=legend_)
    # Labels
    plt.ylabel("Company base (%)")
    plt.show()

def annotate_stacked_bars(ax, pad=0.99, colour="white", textsize=13):
    """
    Add value annotations to the bars
    """

    # Iterate over the plotted rectanges/bars
    for p in ax.patches:

        # Calculate annotation
        value = str(round(p.get_height(),1))
        # If value is 0 do not annotate
        if value == '0.0':
            continue
        ax.annotate(
            value,
            ((p.get_x()+ p.get_width()/2)*pad-0.05, (p.get_y()
+p.get_height()/2)*pad),
            color=colour,
            size=textsize
        )

def plot_distribution(dataframe, column, ax, bins_=50):
    """
    Plot variable distirbution in a stacked histogram of churned or
retained company
    """
    # Create a temporal dataframe with the data to be plot
    temp = pd.DataFrame({"Retention": dataframe[dataframe["churn"]==0]
[column],
```

```
    "Churn":dataframe[dataframe["churn"]==1][column]})
    # Plot the histogram
    temp[["Retention","Churn"]].plot(kind='hist', bins=bins_, ax=ax,
stacked=True)
    # X-axis label
    ax.set_xlabel(column)
    # Change the x-axis to plain style
    ax.ticklabel_format(style='plain', axis='x')
```

Start by looking at churning status

```
churn = client_df[['id', 'churn']]
churn.columns = ['Companies', 'churn']
churn_total = churn.groupby(churn['churn']).count()
churn_percentage = churn_total / churn_total.sum() * 100
plot_stacked_bars(churn_percentage.transpose(), "Churning Status", (5,
5), legend_="lower right")
```



About 10% of total customer have churned.

Consumption

We will see if churn based on power consumption and activity or not.

```
consumption = client_df[['id', 'cons_12m', 'cons_gas_12m',
'cons_last_month', 'imp_cons', 'has_gas', 'churn']]
```

```
fig, axs = plt.subplots(nrows=1, figsize=(18, 5))

plot_distribution(consumption, 'cons_12m', axs)
```