# Feature Engineering

---

1. Import packages
2. Load data
3. Feature engineering

---

## 1. Import packages

In [1]:
```
import pandas as pd
```

---

## 2. Load data

In [2]:
```
df = pd.read_csv('clean_data_after_eda.csv')
df["date_activ"] = pd.to_datetime(df["date_activ"], format='%d-%m-%Y')
df["date_end"] = pd.to_datetime(df["date_end"], format='%d-%m-%Y')
df["date_modif_prod"] = pd.to_datetime(df["date_modif_prod"], format='%d-%m-%Y')
df["date_renewal"] = pd.to_datetime(df["date_renewal"], format='%d-%m-%Y')
```

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

Out[3]:

| | id | channel_sales | cons_12m | cons_gas_12m | cons_last_month | date_activ | date_end | date_mod |
|---|---|---|---|---|---|---|---|---|
| **0** | 24011ae4ebbe3035111d65fa7c15bc57 | foosdfpfkusacimwkcsosbicdxkicaua | 0 | 54946 | 0 | 2013-06-15 | 2016-06-15 | 201 |
| **1** | d29c2c54acc38ff3c0614d0a653813dd | MISSING | 4660 | 0 | 0 | 2009-08-21 | 2016-08-30 | 200 |
| **2** | 764c75f661154dac3a6c254cd082ea7d | foosdfpfkusacimwkcsosbicdxkicaua | 544 | 0 | 0 | 2010-04-16 | 2016-04-16 | 201 |

3 rows × 44 columns

# 3. Feature engineering

## Difference between off-peak prices in December and preceding January

Below is the code created by your colleague to calculate the feature described above. Use this code to re-create this feature and then think about ways to build on this feature to create features with a higher predictive power.

In [4]:
```
price_df = pd.read_csv('price_data (1).csv')
price_df["price_date"] = pd.to_datetime(price_df["price_date"], format='%d-%m-%Y')
price_df.head()
```

Out[4]:

| | id | price_date | price_off_peak_var | price_peak_var | price_mid_peak_var | price_off_peak_fix | price_peak_fix | price_mid |
|---|---|---|---|---|---|---|---|---|
| **0** | 038af19179925da21a25619c5a24b745 | 2015-01-01 | 0.151367 | 0.0 | 0.0 | 44.266931 | 0.0 | |
| **1** | 038af19179925da21a25619c5a24b745 | 2015-02-01 | 0.151367 | 0.0 | 0.0 | 44.266931 | 0.0 | |
| **2** | 038af19179925da21a25619c5a24b745 | 2015-03-01 | 0.151367 | 0.0 | 0.0 | 44.266931 | 0.0 | |
| **3** | 038af19179925da21a25619c5a24b745 | 2015-04-01 | 0.149626 | 0.0 | 0.0 | 44.266931 | 0.0 | |
| **4** | 038af19179925da21a25619c5a24b745 | 2015-05-01 | 0.149626 | 0.0 | 0.0 | 44.266931 | 0.0 | |

In [5]:
```
# Group off-peak prices by companies and month
monthly_price_by_id = price_df.groupby(['id', 'price_date']).agg({'price_off_peak_var': 'mean', 'price_off_peak_fix': 'mean'})

# Get january and december prices
jan_prices = monthly_price_by_id.groupby('id').first().reset_index()
dec_prices = monthly_price_by_id.groupby('id').last().reset_index()

# Calculate the difference
diff = pd.merge(dec_prices.rename(columns={'price_off_peak_var': 'dec_1', 'price_off_peak_fix': 'dec_2'}), jan_prices.drop(col
diff['offpeak_diff_dec_january_energy'] = diff['dec_1'] - diff['price_off_peak_var']
diff['offpeak_diff_dec_january_power'] = diff['dec_2'] - diff['price_off_peak_fix']
diff = diff[['id', 'offpeak_diff_dec_january_energy','offpeak_diff_dec_january_power']]
diff.head()
```

Out[5]:

| | id | offpeak_diff_dec_january_energy | offpeak_diff_dec_january_power |
|---|---|---|---|
| **0** | 0002203ffbb812588b632b9e628cc38d | -0.006192 | 0.162916 |
| **1** | 0004351ebdd665e6ee664792efc4fd13 | -0.004104 | 0.177779 |
| **2** | 0010bcc39e42b3c2131ed2ce55246e3c | 0.050443 | 1.500000 |
| **3** | 0010ee3855fdea87602a5b7aba8e42de | -0.010018 | 0.162916 |
| **4** | 00114d74e963e47177db89bc70108537 | -0.003994 | -0.000001 |

Now it is time to get creative and to conduct some of your own feature engineering! Have fun with it, explore different ideas and try to create as many as yo can!

In [6]:
```
diff.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16096 entries, 0 to 16095
Data columns (total 3 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              16096 non-null  object 
 1   offpeak_diff_dec_january_energy  16096 non-null  float64
 2   offpeak_diff_dec_january_power   16096 non-null  float64
dtypes: float64(2), object(1)
memory usage: 503.0+ KB
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14606 entries, 0 to 14605
Data columns (total 44 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  datetime64[ns]
 6   date_end                         14606 non-null  datetime64[ns]
 7   date_modif_prod                  14606 non-null  datetime64[ns]
 8   date_renewal                     14606 non-null  datetime64[ns]
 9   forecast_cons_12m                14606 non-null  float64
 10  forecast_cons_year               14606 non-null  int64
 11  forecast_discount_energy         14606 non-null  int64
 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  var_year_price_off_peak_var      14606 non-null  float64
 26  var_year_price_peak_var          14606 non-null  float64
 27  var_year_price_mid_peak_var      14606 non-null  float64
 28  var_year_price_off_peak_fix      14606 non-null  float64
 29  var_year_price_peak_fix          14606 non-null  float64
 30  var_year_price_mid_peak_fix      14606 non-null  float64
 31  var_year_price_off_peak          14606 non-null  float64
 32  var_year_price_peak              14606 non-null  float64
 33  var_year_price_mid_peak          14606 non-null  float64
 34  var_6m_price_off_peak_var        14606 non-null  float64
 35  var_6m_price_peak_var            14606 non-null  float64
 36  var_6m_price_mid_peak_var        14606 non-null  float64
```

```
 37  var_6m_price_off_peak_fix       14606 non-null  float64
 38  var_6m_price_peak_fix           14606 non-null  float64
 39  var_6m_price_mid_peak_fix       14606 non-null  float64
 40  var_6m_price_off_peak           14606 non-null  float64
 41  var_6m_price_peak               14606 non-null  float64
 42  var_6m_price_mid_peak           14606 non-null  float64
 43  churn                           14606 non-null  int64
dtypes: datetime64[ns](4), float64(28), int64(8), object(4)
memory usage: 4.9+ MB
```

## Extract same type of values

In [8]:
```
data = df.drop(['id','channel_sales', 'date_activ','date_end','date_modif_prod','date_renewal','has_gas', 'origin_up'], axis=1
X = data.drop('churn', axis=1)
Y = data['churn']
```

In [9]:
```
X.describe()
```

Out[9]:

| | cons_12m | cons_gas_12m | cons_last_month | forecast_cons_12m | forecast_cons_year | forecast_discount_energy | forecast_meter_rent_12m | forec |
|---|---|---|---|---|---|---|---|---|
| **count** | 1.460600e+04 | 1.460600e+04 | 14606.000000 | 14606.000000 | 14606.000000 | 14606.000000 | 14606.000000 | |
| **mean** | 1.592203e+05 | 2.809238e+04 | 16090.269752 | 1868.614880 | 1399.762906 | 0.966726 | 63.086871 | |
| **std** | 5.734653e+05 | 1.629731e+05 | 64364.196422 | 2387.571531 | 3247.786255 | 5.108289 | 66.165783 | |
| **min** | 0.000000e+00 | 0.000000e+00 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | |
| **25%** | 5.674750e+03 | 0.000000e+00 | 0.000000 | 494.995000 | 0.000000 | 0.000000 | 16.180000 | |
| **50%** | 1.411550e+04 | 0.000000e+00 | 792.500000 | 1112.875000 | 314.000000 | 0.000000 | 18.795000 | |
| **75%** | 4.076375e+04 | 0.000000e+00 | 3383.000000 | 2401.790000 | 1745.750000 | 0.000000 | 131.030000 | |
| **max** | 6.207104e+06 | 4.154590e+06 | 771203.000000 | 82902.830000 | 175375.000000 | 30.000000 | 599.310000 | |

8 rows × 35 columns

In [10]:
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14606 entries, 0 to 14605
Data columns (total 35 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   cons_12m                         14606 non-null  int64  
 1   cons_gas_12m                     14606 non-null  int64  
 2   cons_last_month                  14606 non-null  int64  
 3   forecast_cons_12m                14606 non-null  float64
 4   forecast_cons_year               14606 non-null  int64  
 5   forecast_discount_energy         14606 non-null  int64  
 6   forecast_meter_rent_12m          14606 non-null  float64
 7   forecast_price_energy_off_peak   14606 non-null  float64
 8   forecast_price_energy_peak       14606 non-null  float64
 9   forecast_price_pow_off_peak      14606 non-null  float64
 10  imp_cons                         14606 non-null  float64
 11  margin_gross_pow_ele             14606 non-null  float64
 12  margin_net_pow_ele               14606 non-null  float64
 13  nb_prod_act                      14606 non-null  int64  
 14  net_margin                       14606 non-null  float64
 15  num_years_antig                  14606 non-null  int64  
 16  pow_max                          14606 non-null  float64
 17  var_year_price_off_peak_var      14606 non-null  float64
 18  var_year_price_peak_var          14606 non-null  float64
 19  var_year_price_mid_peak_var      14606 non-null  float64
 20  var_year_price_off_peak_fix      14606 non-null  float64
 21  var_year_price_peak_fix          14606 non-null  float64
 22  var_year_price_mid_peak_fix      14606 non-null  float64
 23  var_year_price_off_peak          14606 non-null  float64
 24  var_year_price_peak              14606 non-null  float64
 25  var_year_price_mid_peak          14606 non-null  float64
 26  var_6m_price_off_peak_var        14606 non-null  float64
 27  var_6m_price_peak_var            14606 non-null  float64
 28  var_6m_price_mid_peak_var        14606 non-null  float64
 29  var_6m_price_off_peak_fix        14606 non-null  float64
 30  var_6m_price_peak_fix            14606 non-null  float64
 31  var_6m_price_mid_peak_fix        14606 non-null  float64
 32  var_6m_price_off_peak            14606 non-null  float64
 33  var_6m_price_peak                14606 non-null  float64
 34  var_6m_price_mid_peak            14606 non-null  float64
dtypes: float64(28), int64(7)
memory usage: 3.9 MB
```

In [11]:
```
Y
```

Out[11]:
```
0        1
1        0
2        0
3        0
4        0
        ..
14601    0
14602    1
14603    1
14604    0
14605    0
Name: churn, Length: 14606, dtype: int64
```

# Traning the model

In [12]:
```
from sklearn.model_selection import train_test_split
train_set_x, test_set_x, train_set_y, test_set_y = train_test_split(X, Y, test_size=0.3, random_state=42)
print(f'size of train_set: {len(train_set_x), len(train_set_y)} \nsize of test_set : {len(test_set_x), len(test_set_y)}\n')
```

```
size of train_set: (10224, 10224)
size of test_set : (4382, 4382)
```

In [13]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(train_set_x, train_set_y)
```

Out[13]:
```
RandomForestClassifier()
```

In [14]:
```
y_pred = model.predict(test_set_x)
y_pred
```

Out[14]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [15]:
```
dataframe= pd.DataFrame(y_pred)
dataframe
```

Out[15]:

| | 0 |
|---|---|
| **0** | 0 |
| **1** | 0 |
| **2** | 0 |
| **3** | 0 |
| **4** | 0 |
| **...** | ... |
| **4377** | 0 |
| **4378** | 0 |
| **4379** | 0 |
| **4380** | 0 |
| **4381** | 0 |

4382 rows × 1 columns

In [16]:
```
from sklearn import metrics
accu = metrics.accuracy_score(test_set_y, y_pred)
accu
```

Out[16]:
```
0.9020994979461433
```

In [ ]:
```

```