BCG Task 3

Arnav Singh

Sub-task 1: Think through what key drivers of churn could be for our client

Sub-task 2: Build the features in order to get ready to model

Import packages

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
import pandas as pd
import seaborn as sns
import datetime
import pickle
import warnings
warnings.filterwarnings("ignore")


sns.set(color_codes=True)
pd.set_option('display.max_columns',50)
```

Load data

```
date_cols=['date_activ','date_end','date_modif_prod','date_renewal']
train = pd.read_csv('train_clean.csv',parse_dates=date_cols)
train.head()
```

	id	channel_sales	cons_12m	cons_gas_12m
0	48ada52261e7cf58715202705a0451c9	lmkebamcaaclubfxadlmueccxoimlema	309275	0
1	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946
2	d29c2c54acc38ff3c0614d0a653813dd	NaN	4660	0
3	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0
4	bba03439a292a1e166f80264c16191cb	lmkebamcaaclubfxadlmueccxoimlema	1584	0

```
history = pd.read_csv('history_clean.csv',parse_dates=['price_date'])
history.head()
```

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

Feature engineering

First we need to do the Feature Selection We will create the average consumption of the year as one new feature

```
mean_year=history.groupby(['id']).mean().reset_index()

mean_6m_history[history['price_date']>'2015_06_01'].groupby(['id']).mean().reset_index()
```

```
mean_6m=history[history['price_date']>'2015-06-01'].groupby(['id']).mean().reset_index()

mean_3m=history[history['price_date']>'2015-10-01'].groupby(['id']).mean().reset_index()


#Combine the mean year in a single dataframe
mean_year=mean_year.rename(index=str,columns={'price_p1_var':'mean_year_price_p1_var',
                                              'price_p2_var':'mean_year_price_p2_var',
                                              'price_p3_var':'mean_year_price_p3_var',
                                              'price_p1_fix':'mean_year_price_p1_fix',
                                              'price_p2_fix':'mean_year_price_p2_fix',
                                              'price_p3_fix':'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']


mean_6m=mean_6m.rename(index=str,columns={'price_p1_var':'mean_6m_price_p1_var',
                                          'price_p2_var':'mean_6m_price_p2_var',
                                          'price_p3_var':'mean_6m_price_p3_var',
                                          'price_p1_fix':'mean_6m_price_p1_fix',
                                          'price_p2_fix':'mean_6m_price_p2_fix',
                                          'price_p3_fix':'mean_6m_price_p3_fix',})
mean_6m['mean_6m_price_p1']=mean_6m['mean_6m_price_p1_var']+mean_6m['mean_6m_price_p1_fix']
mean_6m['mean_6m_price_p2']=mean_6m['mean_6m_price_p2_var']+mean_6m['mean_6m_price_p2_fix']
mean_6m['mean_6m_price_p3']=mean_6m['mean_6m_price_p3_var']+mean_6m['mean_6m_price_p3_fix']


mean_3m=mean_3m.rename(index=str,columns={'price_p1_var':'mean_3m_price_p1_var',
                                          'price_p2_var':'mean_3m_price_p2_var',
                                          'price_p3_var':'mean_3m_price_p3_var',
                                          'price_p1_fix':'mean_3m_price_p1_fix',
                                          'price_p2_fix':'mean_3m_price_p2_fix',
                                          'price_p3_fix':'mean_3m_price_p3_fix',})
mean_3m['mean_3m_price_p1']=mean_3m['mean_3m_price_p1_var']+mean_3m['mean_3m_price_p1_fix']
mean_3m['mean_3m_price_p2']=mean_3m['mean_3m_price_p2_var']+mean_3m['mean_3m_price_p2_fix']
mean_3m['mean_3m_price_p3']=mean_3m['mean_3m_price_p3_var']+mean_3m['mean_3m_price_p3_fix']


history_new = pd.merge(mean_year,mean_6m, on='id',how='left')
history_new = pd.merge(mean_year,mean_3m, on='id',how='left')
history_new.head()
```

	id	mean_year_price_p1_var	mean_year_price_p2_var	mean_ye
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	

Datetime

```
#Extract contract duration
#we will define the duration=date_end-date_activ
train['contract_duration']=((train['date_end']-train['date_activ'])/ np.timedelta64(1,'M')).astype(int)
train.head()
```

	id	channel_sales	cons_12m	cons_gas_12m
0	48ada52261e7cf58715202705a0451c9	lmkebamcaaclubfxadlmueccxoimlema	309275	0
1	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946
2	d29c2c54acc38ff3c0614d0a653813dd	NaN	4660	0
3	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0
4	bba03439a292a1e166f80264c16191cb	lmkebamcaaclubfxadlmueccxoimlema	1584	0

```
#set the reference time to be 2016-01-01
#write a finction to caculate the month difference between datetime features
```

```
def calculatemonth(referencetime,dataframe,column):
    time_diff=referencetime-dataframe[column]
    months=(time_diff/np.timedelta64(1,'M')).astype(int)
    return months


referencetime=pd.to_datetime('2016-01-01')


train['activ_diff']=calculatemonth(referencetime,train,'date_activ')
train['end_diff']=calculatemonth(referencetime,train,'date_end')
train['modif_diff']=calculatemonth(referencetime,train,'date_modif_prod')
train['renewal_diff']=calculatemonth(referencetime,train,'date_renewal')
train.head()
```

	id	channel_sales	cons_12m	cons_gas_12m
0	48ada52261e7cf58715202705a0451c9	lmkebamcaaclubfxadlmueccxoimlema	309275	0
1	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946
2	d29c2c54acc38ff3c0614d0a653813dd	NaN	4660	0
3	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0
4	bba03439a292a1e166f80264c16191cb	lmkebamcaaclubfxadlmueccxoimlema	1584	0

```
#Remove the date columns
train.drop(columns=['date_activ','date_end','date_modif_prod','date_renewal'],axis=1,inplace=True)


train.head()
```

	id	channel_sales	cons_12m	cons_gas_12m
0	48ada52261e7cf58715202705a0451c9	lmkebamcaaclubfxadlmueccxoimlema	309275	0
1	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946
2	d29c2c54acc38ff3c0614d0a653813dd	NaN	4660	0
3	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0
4	bba03439a292a1e166f80264c16191cb	lmkebamcaaclubfxadlmueccxoimlema	1584	0

Categorical Data

Binary encoding

```
#For the column has_gas,replace t for 1 and f for 0
train['has_gas']=train['has_gas'].replace(['t','f'],[1,0])
```

one-hot encoding

```
train['channel_sales']=train['channel_sales'].fillna('null_values_channel')
train['channel_sales']=train['channel_sales'].apply(lambda x:x[:4])
categories_channel=pd.get_dummies(train[['channel_sales']])
categories_channel.drop(columns=['channel_sales_null'],inplace=True)
categories_channel.head()
```

	channel_sales_epum	channel_sales_ewpa	channel_sales_fixd	channel_sales_foos	cl
0	0	0	0	0	
1	0	0	0	1	
2	0	0	0	0	
3	0	0	0	1	
4	0	0	0	0	

```
#for the column origin_up, first fill the null value
train['origin_up']=train['origin_up'].fillna('null_values_origin')


train['origin_up']=train['origin_up'].apply(lambda x:x[:4])
categories_origin= pd.get_dummies(train[['origin_up']])
categories_origin.drop(columns=['origin_up_null'],inplace=True)
categories_origin.head()
```

	origin_up_ewxe	origin_up_kamk	origin_up_ldks	origin_up_lxid	origin_up_usap
0	0	0	1	0	0
1	0	0	0	1	0
2	0	1	0	0	0
3	0	1	0	0	0
4	0	1	0	0	0

```
#Use the common index to merge
train=pd.merge(train,categories_channel,left_index=True,right_index=True)
train=pd.merge(train,categories_origin,left_index=True,right_index=True)


train=train.drop(['channel_sales','origin_up'],axis=1)
train.head()
```

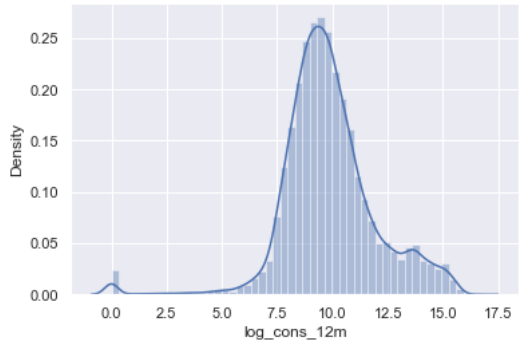
	id	cons_12m	cons_gas_12m	cons_last_month	forecast_cons_
0	48ada52261e7cf58715202705a0451c9	309275	0	10025	26520
1	24011ae4ebbe3035111d65fa7c15bc57	0	54946	0	0
2	d29c2c54acc38ff3c0614d0a653813dd	4660	0	0	189
3	764c75f661154dac3a6c254cd082ea7d	544	0	0	47
4	bba03439a292a1e166f80264c16191cb	1584	0	0	240

Numerical Data

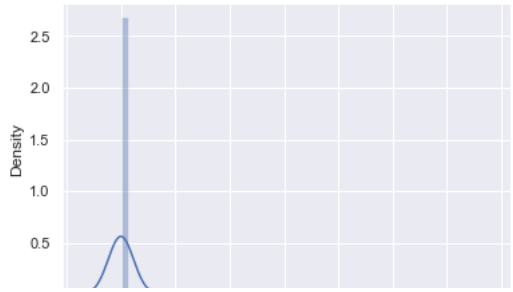
Distribution transformation

From the previous EDA we can see that some features are highly skewed, we need to transform the distribution to normal-like distribution

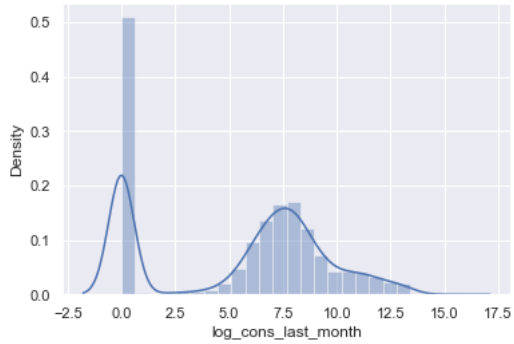
```
#First for the cons_12, remove the negative values and apply a log tranformation
train.loc[train.cons_12m<0,'cons_12m']=np.nan
train['cons_12m']=train['cons_12m'].dropna()
train['log_cons_12m']=train['cons_12m'].apply(lambda x:np.log(1+x))
sns.distplot(train['log_cons_12m']);
```



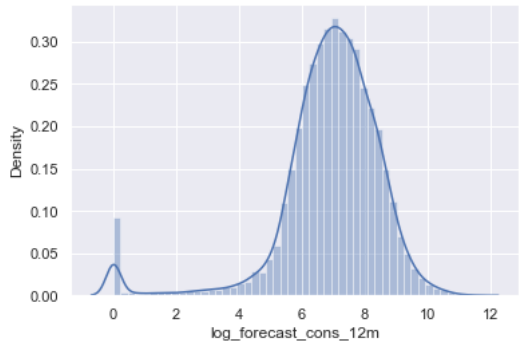

```
train.loc[train.cons_gas_12m<0,'cons_gas_12m']=np.nan
train['cons_gas_12m']=train['cons_gas_12m'].dropna()
train['log_cons_gas_12m']=train['cons_gas_12m'].apply(lambda x:np.log(1+x))
sns.distplot(train['log_cons_gas_12m']);
```



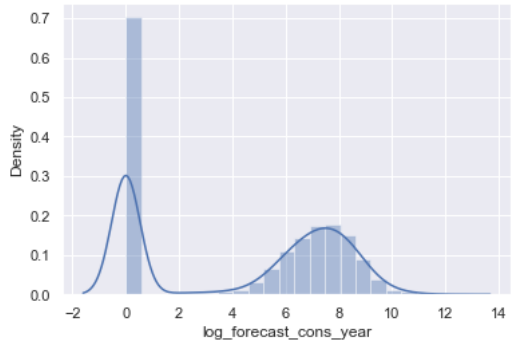

```
train.loc[train.cons_last_month<0,'cons_last_month']=np.nan
train['cons_last_month']=train['cons_last_month'].dropna()
train['log_cons_last_month']=train['cons_last_month'].apply(lambda x:np.log(1+x))
sns.distplot(train['log_cons_last_month']);
```




```
train.loc[train.forecast_cons_12m<0,'forecast_cons_12m']=np.nan
train['forecast_cons_12m']=train['forecast_cons_12m'].dropna()
train['log_forecast_cons_12m']=train['forecast_cons_12m'].apply(lambda x:np.log(1+x))
sns.distplot(train['log_forecast_cons_12m']);
```

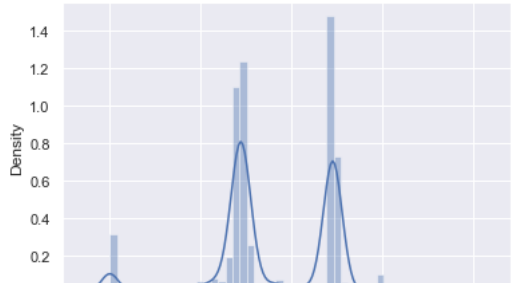



```
train.loc[train.forecast_cons_year<0,'forecast_cons_year']=np.nan
train['forecast_cons_year']=train['forecast_cons_year'].dropna()
train['log_forecast_cons_year']=train['forecast_cons_year'].apply(lambda x:np.log(1+x))
sns.distplot(train['log_forecast_cons_year']);
```

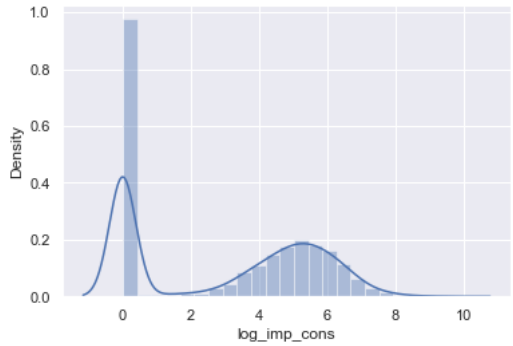



```
train.loc[train.forecast_meter_rent_12m<0,'forecast_meter_rent_12m']=np.nan
train['forecast_meter_rent_12m']=train['forecast_meter_rent_12m'].dropna()
train['log_forecast_meter_rent_12m']=train['forecast_meter_rent_12m'].apply(lambda x:np.log(1+x))
sns.distplot(train['log_forecast_meter_rent_12m'])
```

```
<AxesSubplot:xlabel='log_forecast_meter_rent_12m', ylabel='Density'>
```




```
train.loc[train.imp_cons<0,'imp_cons']=np.nan
train['imp_cons']=train['imp_cons'].dropna()
train['log_imp_cons']=train['imp_cons'].apply(lambda x:np.log(1+x))
sns.distplot(train['log_imp_cons']);
```



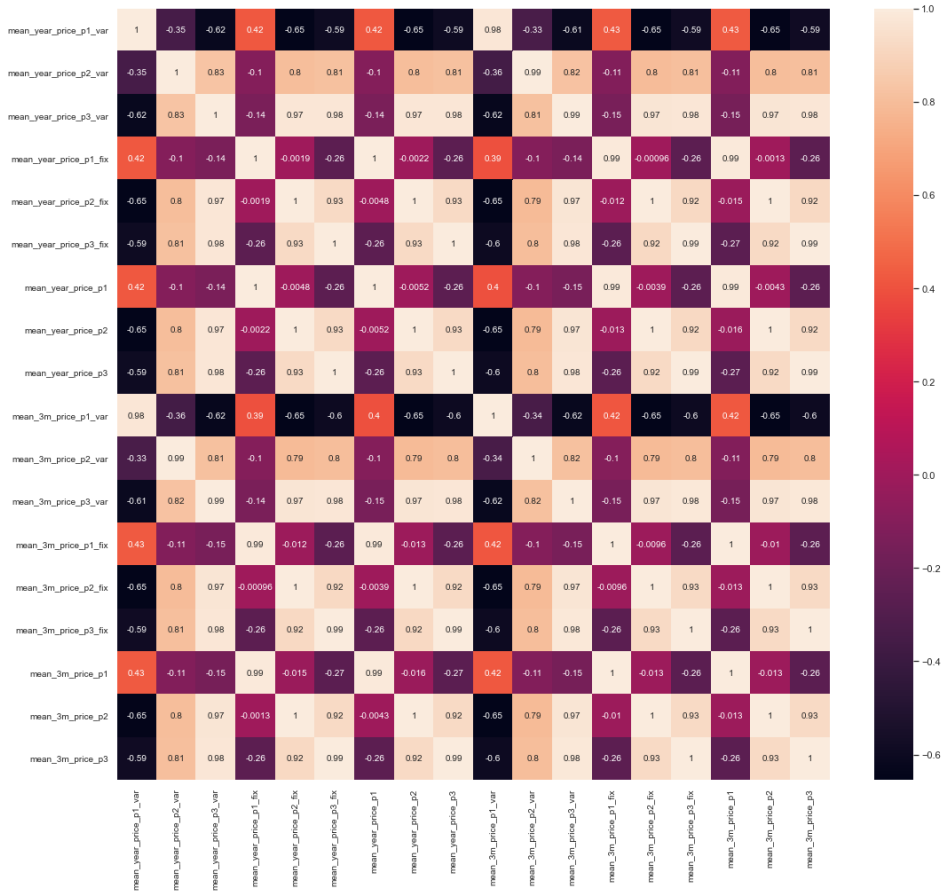

```
train=train.drop(['cons_12m','cons_gas_12m','cons_last_month','forecast_cons_12m','forecast_cons_year','forecast_meter_rent_12m','imp_cor
train.head()
```

	id	forecast_discount_energy	forecast_price_energy_p1	for
0	48ada52261e7cf58715202705a0451c9	0.0	0.095919	
1	24011ae4ebbe3035111d65fa7c15bc57	0.0	0.114481	
2	d29c2c54acc38ff3c0614d0a653813dd	0.0	0.145711	
3	764c75f661154dac3a6c254cd082ea7d	0.0	0.165794	
4	bba03439a292a1e166f80264c16191cb	0.0	0.146694	

High correlation variables

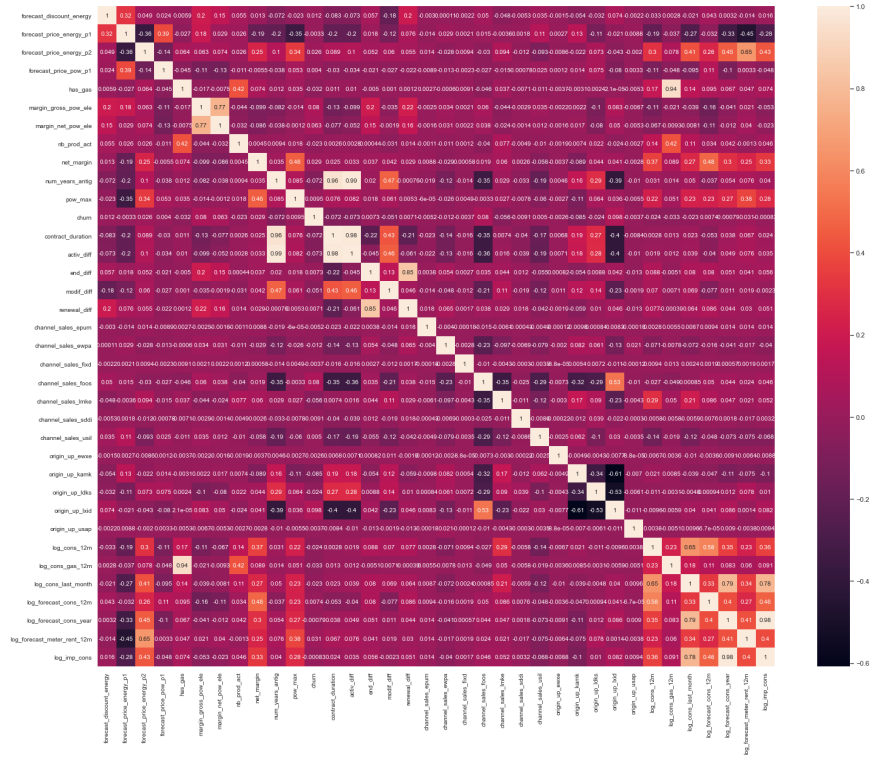
```
#Calculate correlation of variables
corr_hist=history_new.corr()
```

```
plt.figure(figsize=(18,16))
sns.heatmap(corr_hist,xticklabels=corr_hist.columns.values,
           yticklabels=corr_hist.columns.values,annot=True,annot_kws={'size':10})
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.show()
```




```
#Calculate correlation of variables
corr_train=train.corr()


plt.figure(figsize=(25,20))
sns.heatmap(corr_train,xticklabels=corr_train.columns.values,
           yticklabels=corr_train.columns.values,annot=True,annot_kws={'size':10})
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.show()
```



As expected, num_years_antig has a high correlation with activ_diff, we can remove the num_years_antig since they are provides the same information.

```
train.drop(columns=['num_years_antig'],inplace=True)
```

Removing Outliers

As the previous EDA we can see that there have several outliers in the dataset,for simplisity, I will replace these outliers with the mean.

```
#I will use IQR method to detect outliers
def remove_outliers(df,col):
    df.loc[df[col]<0,col]=df[col].mean()
    q1=df[col].quantile(.25)
    q3=df[col].quantile(.75)
    iqr=q3-q1
    upper_bound=q3+(iqr*1.5)
    lower_bound=q1-(iqr*1.5)
    df=df[(lower_bound<df[col])|(df[col]<upper_bound)]
```

```
remove_outliers(history_new,'mean_year_price_p1_var')
remove_outliers(history_new,'mean_year_price_p2_var')
remove_outliers(history_new,'mean_year_price_p3_var')
remove_outliers(history_new,'mean_year_price_p1_fix')
remove_outliers(history_new,'mean_year_price_p2_fix')
remove_outliers(history_new,'mean_year_price_p3_fix')
remove_outliers(history_new,'mean_year_price_p1')
remove_outliers(history_new,'mean_year_price_p2')
remove_outliers(history_new,'mean_year_price_p3')
remove_outliers(train,'log_cons_12m')
remove_outliers(train,'log_cons_gas_12m')
remove_outliers(train,'log_cons_last_month')
remove_outliers(train,'log_forecast_cons_12m')
remove_outliers(train,'log_forecast_meter_rent_12m')
remove_outliers(train,'log_forecast_cons_year')
remove_outliers(train,'log_imp_cons')
remove_outliers(train,'forecast_discount_energy')
remove_outliers(train,'forecast_price_energy_p1')
remove_outliers(train,'forecast_price_energy_p2')
remove_outliers(train,'forecast_price_pow_p1')
remove_outliers(train,'margin_gross_pow_ele')
remove_outliers(train,'margin_net_pow_ele')
remove_outliers(train,'net_margin')
remove_outliers(train,'pow_max')
remove_outliers(train,'forecast_price_energy_p1')
```

Merge Data Together

```
df = pd.merge(train, history_new, on='id', how = 'left')
df.head()
```

	id	forecast_discount_energy	forecast_price_energy_p1	for
0	48ada52261e7cf58715202705a0451c9	0.0	0.095919	
1	24011ae4ebbe3035111d65fa7c15bc57	0.0	0.114481	
2	d29c2c54acc38ff3c0614d0a653813dd	0.0	0.145711	
3	764c75f661154dac3a6c254cd082ea7d	0.0	0.165794	
4	bba03439a292a1e166f80264c16191cb	0.0	0.146694	

5 rows × 54 columns

```
df =df.dropna()
```

```
df.isnull().sum().sort_values(ascending = False)
```

```
mean_3m_price_p3               0
activ_diff                     0
origin_up_ewxe                 0
channel_sales_usil             0
channel_sales_sddi             0
channel_sales_lmke             0
channel_sales_foos             0
channel_sales_fixd             0
channel_sales_ewpa             0
channel_sales_epum             0
renewal_diff                   0
modif_diff                     0
end_diff                       0
contract_duration              0
mean_3m_price_p2               0
churn                          0
pow_max                        0
net_margin                     0
nb_prod_act                    0
margin_net_pow_ele             0
margin_gross_pow_ele           0
has_gas                        0
forecast_price_pow_p1          0
forecast_price_energy_p2       0
forecast_price_energy_p1       0
forecast_discount_energy       0
origin_up_kamk                 0
origin_up_ldks                 0
origin_up_lxid                 0
origin_up_usap                 0
mean_3m_price_p1               0
mean_3m_price_p3_fix           0
mean_3m_price_p2_fix           0
mean_3m_price_p1_fix           0
mean_3m_price_p3_var           0
mean_3m_price_p2_var           0
mean_3m_price_p1_var           0
mean_year_price_p3             0
mean_year_price_p2             0
mean_year_price_p1             0
mean_year_price_p3_fix         0
mean_year_price_p2_fix         0
mean_year_price_p1_fix         0
mean_year_price_p3_var         0
mean_year_price_p2_var         0
mean_year_price_p1_var         0
log_imp_cons                   0
log_forecast_meter_rent_12m    0
log_forecast_cons_year         0
log_forecast_cons_12m          0
log_cons_last_month            0
log_cons_gas_12m               0
log_cons_12m                   0
id                             0
dtype: int64
```

```
df=df.drop('id', axis = 1)
```

```
df.to_csv('feature_engineering.csv', index = False)
```

Colab paid products - Cancel contracts here