# Feature Engineering for Energy clients

In this assignment, our objective is to perform feature engineering to modify variables to ensure better analysis for Churn Rate. Variables would be converted into appropriate format, redundant variables would be removed and new variables would be created as per needs basis.

In [66]:
```
#Getting the packages
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
import pandas as pd
import numpy as np
import seaborn as sns
from datetime import datetime
import matplotlib.pyplot as plt
from scipy import stats
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
%matplotlib inline
sns.set(color_codes=True)
```

In [4]:
```
#getting the dataset
df = pd.read_csv('C:\\Users\\sujoydutta\\Downloads\\clean_data_after_eda.csv')
df.head()
```

Out[4]:

| | id | channel_sales | cons_12m | cons_gas_12m | cons_last_mon |
|---|---|---|---|---|---|
| **0** | 24011ae4ebbe3035111d65fa7c15bc57 | foosdfpfkusacimwkcsosbicdxkicaua | 0 | 54946 | |
| **1** | d29c2c54acc38ff3c0614d0a653813dd | MISSING | 4660 | 0 | |
| **2** | 764c75f661154dac3a6c254cd082ea7d | foosdfpfkusacimwkcsosbicdxkicaua | 544 | 0 | |
| **3** | bba03439a292a1e166f80264c16191cb | lmkebamcaaclubfxadlmueccxoimlema | 1584 | 0 | |
| **4** | 149d57cf92fc41cf94415803a877cb4b | MISSING | 4425 | 0 | 5 |

5 rows × 44 columns

In [5]:
```
#getting the information on the dataset
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  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  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: float64(29), int64(7), object(8)
memory usage: 4.9+ MB
```

In [8]:

```
#getting secondary dataset
prices = pd.read_csv('C:\\Users\\sujoydutta\\Downloads\\price_data.csv')
prices.head()
```

Out[8]:

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

In [9]:

```
#getting the information on the dataset
prices.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193002 entries, 0 to 193001
Data columns (total 8 columns):
 #   Column              Non-Null Count   Dtype
```

```
 ---  ------             --------------  -----
 0   id                 193002 non-null  object
 1   price_date         193002 non-null  object
 2   price_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
```

In [12]:

```
#modifying the date columns
df["date_activ"] = pd.to_datetime(df["date_activ"], format='%Y-%m-%d')
df["date_end"] = pd.to_datetime(df["date_end"], format='%Y-%m-%d')
df["date_modif_prod"] = pd.to_datetime(df["date_modif_prod"], format='%Y-%m-%d')
df["date_renewal"] = pd.to_datetime(df["date_renewal"], format='%Y-%m-%d')
prices["price_date"] = pd.to_datetime(prices["price_date"], format='%Y-%m-%d')
```

In [14]:

```
#examining df
df.head(3)
```

Out[14]:

| | id | price_date | price_off_peak_var | price_peak_var | price_mid_peak_var | price_off |
|---|---|---|---|---|---|---|
| 0 | 038af19179925da21a25619c5a24b745 | 2015-01-01 | 0.151367 | 0.0 | 0.0 | 4 |
| 1 | 038af19179925da21a25619c5a24b745 | 2015-02-01 | 0.151367 | 0.0 | 0.0 | 4 |
| 2 | 038af19179925da21a25619c5a24b745 | 2015-03-01 | 0.151367 | 0.0 | 0.0 | 4 |

In [15]:

```
#examining prices
prices.head(3)
```

Out[15]:

| | id | price_date | price_off_peak_var | price_peak_var | price_mid_peak_var | price_off |
|---|---|---|---|---|---|---|
| 0 | 038af19179925da21a25619c5a24b745 | 2015-01-01 | 0.151367 | 0.0 | 0.0 | 4 |
| 1 | 038af19179925da21a25619c5a24b745 | 2015-02-01 | 0.151367 | 0.0 | 0.0 | 4 |
| 2 | 038af19179925da21a25619c5a24b745 | 2015-03-01 | 0.151367 | 0.0 | 0.0 | 4 |

In [17]:

```
# Getting Difference between off-peak prices in December and preceding January
monthly_price_by_id = prices.groupby(['id', 'price_date']).agg({'price_off_peak_var': 'm


jan_prices = monthly_price_by_id.groupby('id').first().reset_index()
dec_prices = monthly_price_by_id.groupby('id').last().reset_index()


diff = pd.merge(dec_prices.rename(columns={'price_off_peak_var': 'dec_1', 'price_off_pea
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[17]:

| | 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 |

In [18]:
```
#merging with main dataset
df = pd.merge(df, diff, on='id')
df.head()
```

Out[18]:

| | id | channel_sales | cons_12m | cons_gas_12m | cons_last_mon |
|---|---|---|---|---|---|
| 0 | 24011ae4ebbe3035111d65fa7c15bc57 | foosdfpfkusacimwkcsosbicdxkicaua | 0 | 54946 | |
| 1 | d29c2c54acc38ff3c0614d0a653813dd | MISSING | 4660 | 0 | |
| 2 | 764c75f661154dac3a6c254cd082ea7d | foosdfpfkusacimwkcsosbicdxkicaua | 544 | 0 | |
| 3 | bba03439a292a1e166f80264c16191cb | lmkebamcaaclubfxadlmueccxoimlema | 1584 | 0 | |
| 4 | 149d57cf92fc41cf94415803a877cb4b | MISSING | 4425 | 0 | 5 |

5 rows × 46 columns

In [19]:
```
# Aggregating average prices per period by company
mean_prices_by_month = prices.groupby(['id', 'price_date']).agg({
    'price_off_peak_var': 'mean',
    'price_peak_var': 'mean',
    'price_mid_peak_var': 'mean',
    'price_off_peak_fix': 'mean',
    'price_peak_fix': 'mean',
    'price_mid_peak_fix': 'mean'
}).reset_index()
```

In [20]:
```
# Calculating the mean difference between consecutive periods
mean_prices_by_month['off_peak_peak_var_mean_diff'] = mean_prices_by_month['price_off_pe
mean_prices_by_month['peak_mid_peak_var_mean_diff'] = mean_prices_by_month['price_peak_v
mean_prices_by_month['off_peak_mid_peak_var_mean_diff'] = mean_prices_by_month['price_of
mean_prices_by_month['off_peak_peak_fix_mean_diff'] = mean_prices_by_month['price_off_pe
mean_prices_by_month['peak_mid_peak_fix_mean_diff'] = mean_prices_by_month['price_peak_f
mean_prices_by_month['off_peak_mid_peak_fix_mean_diff'] = mean_prices_by_month['price_of
```

In [21]:
```
# Calculating the maximum monthly difference across time periods
max_diff_across_periods_months = mean_prices_by_month.groupby(['id']).agg({
    'off_peak_peak_var_mean_diff': 'max',
    'peak_mid_peak_var_mean_diff': 'max',
    'off_peak_mid_peak_var_mean_diff': 'max',
    'off_peak_peak_fix_mean_diff': 'max',
    'peak_mid_peak_fix_mean_diff': 'max',
    'off_peak_mid_peak_fix_mean_diff': 'max'
}).reset_index().rename(
    columns={
        'off_peak_peak_var_mean_diff': 'off_peak_peak_var_max_monthly_diff',
        'peak_mid_peak_var_mean_diff': 'peak_mid_peak_var_max_monthly_diff',
        'off_peak_mid_peak_var_mean_diff': 'off_peak_mid_peak_var_max_monthly_diff',
        'off_peak_peak_fix_mean_diff': 'off_peak_peak_fix_max_monthly_diff',
        'peak_mid_peak_fix_mean_diff': 'peak_mid_peak_fix_max_monthly_diff',
```

```
            'off_peak_mid_peak_fix_mean_diff': 'off_peak_mid_peak_fix_max_monthly_diff'
    }
)
```

In [22]:
```
#Examining the dataset
max_diff_across_periods_months.head()
```

Out[22]:

| | id | off_peak_peak_var_max_monthly_diff | peak_mid_peak_var_max_monthly_diff |
|---|---|---|---|
| **0** | 0002203ffbb812588b632b9e628cc38d | 0.022225 | 0.033743 |
| **1** | 0004351ebdd665e6ee664792efc4fd13 | 0.148405 | 0.000000 |
| **2** | 0010bcc39e42b3c2131ed2ce55246e3c | 0.205742 | 0.000000 |
| **3** | 0010ee3855fdea87602a5b7aba8e42de | 0.022581 | 0.031859 |
| **4** | 00114d74e963e47177db89bc70108537 | 0.149902 | 0.000000 |

In [26]:
```
#merging with main dataframe
columns = [
    'id',
    'off_peak_peak_var_max_monthly_diff',
    'peak_mid_peak_var_max_monthly_diff',
    'off_peak_mid_peak_var_max_monthly_diff',
    'off_peak_peak_fix_max_monthly_diff',
    'peak_mid_peak_fix_max_monthly_diff',
    'off_peak_mid_peak_fix_max_monthly_diff'
]

df = pd.merge(df, max_diff_across_periods_months[columns], on='id')
df.head()
```

Out[26]:

| | id | channel_sales | cons_12m | cons_gas_12m | cons_last_mon |
|---|---|---|---|---|---|
| **0** | 24011ae4ebbe3035111d65fa7c15bc57 | foosdfpfkusacimwkcsosbicdxkicaua | 0 | 54946 | |
| **1** | d29c2c54acc38ff3c0614d0a653813dd | MISSING | 4660 | 0 | |
| **2** | 764c75f661154dac3a6c254cd082ea7d | foosdfpfkusacimwkcsosbicdxkicaua | 544 | 0 | |
| **3** | bba03439a292a1e166f80264c16191cb | lmkebamcaaclubfxadlmueccxoimlema | 1584 | 0 | |
| **4** | 149d57cf92fc41cf94415803a877cb4b | MISSING | 4425 | 0 | 5 |

5 rows × 52 columns

In [29]:
```
#creating new column customer tenure in days
df['customer_tenure_days'] = (pd.to_datetime(df['date_end']) - pd.to_datetime(df['date_a
df['customer_tenure_days']
```

Out[29]:
```
0        1096
1        2566
2        2192
3        2192
4        2245
         ...
14601    1445
14602    1461
14603    1460
```

```
14604    1461
14605    2556
Name: customer_tenure_days, Length: 14606, dtype: int64
```

In [30]:
```
#creating new column time to renew in days
df['time_to_renewal_days'] = (pd.to_datetime(df['date_renewal']) - pd.to_datetime(df['da
df['time_to_renewal_days']
```

Out[30]:
```
0        -131
1        2201
2        1827
3        1827
4        1881
         ...
14601    -347
14602    1096
14603    1097
14604    1096
14605    2194
Name: time_to_renewal_days, Length: 14606, dtype: int64
```

In [31]:
```
#creating new column mean monthly consumption
df['avg_monthly_consumption'] = df['cons_12m'] / 12
df['avg_monthly_consumption']
```

Out[31]:
```
0           0.000000
1         388.333333
2          45.333333
3         132.000000
4         368.750000
            ...
14601    2689.166667
14602     601.916667
14603     153.666667
14604      10.916667
14605     727.500000
Name: avg_monthly_consumption, Length: 14606, dtype: float64
```

In [32]:
```
#creating new column Forecast vs actual consumption error
df['forecast_error'] = df['forecast_cons_12m'] - df['cons_12m']
df['forecast_error']
```

Out[32]:
```
0            0.00
1        -4470.05
2         -496.04
3        -1343.96
4        -3979.25
           ...
14601   -27621.99
14602    -6591.31
14603    -1653.61
14604     -111.66
14605    -7967.59
Name: forecast_error, Length: 14606, dtype: float64
```

In [33]:
```
#creating new column consumption variance
df['consumption_change'] = df['cons_last_month'] - df['avg_monthly_consumption']
df['consumption_change']
```

Out[33]:
```
0           0.000000
1        -388.333333
2         -45.333333
3        -132.000000
4         157.250000
            ...
14601   -2689.166667
14602    -420.916667
```

```
14603      25.333333
14604     -10.916667
14605    -727.500000
Name: consumption_change, Length: 14606, dtype: float64
```

In [34]:
```
#Modifying has gas column to binary values
df['has_gas'] = df['has_gas'].replace(['t', 'f'], [1, 0])
df['has_gas']
```

Out[34]:
```
0        1
1        0
2        0
3        0
4        0
        ..
14601    1
14602    0
14603    0
14604    0
14605    0
Name: has_gas, Length: 14606, dtype: int64
```

In [37]:
```
#seeing product diversity
df['product_diversity'] = df['nb_prod_act'] + df['has_gas']
df['product_diversity']
```

Out[37]:
```
0        3
1        1
2        1
3        1
4        1
        ..
14601    3
14602    1
14603    1
14604    1
14605    1
Name: product_diversity, Length: 14606, dtype: int64
```

In [38]:
```
# Dropping irrelevant columns
columns_to_drop = [
    'date_modif_prod',
    'pow_max',
    'forecast_discount_energy',
    'forecast_price_pow_off_peak',
    'imp_cons',
    'var_year_price_off_peak_var', 'var_year_price_peak_var', 'var_year_price_mid_peak_v
    'var_year_price_off_peak_fix', 'var_year_price_peak_fix', 'var_year_price_mid_peak_f
    'var_year_price_off_peak', 'var_year_price_peak', 'var_year_price_mid_peak',
    'var_6m_price_off_peak_var', 'var_6m_price_peak_var', 'var_6m_price_mid_peak_var',
    'var_6m_price_off_peak_fix', 'var_6m_price_peak_fix', 'var_6m_price_mid_peak_fix',
    'var_6m_price_off_peak', 'var_6m_price_peak', 'var_6m_price_mid_peak',
    'offpeak_diff_dec_january_energy', 'offpeak_diff_dec_january_power',
    'off_peak_peak_var_max_monthly_diff', 'peak_mid_peak_var_max_monthly_diff',
    'off_peak_mid_peak_var_max_monthly_diff', 'off_peak_peak_fix_max_monthly_diff',
    'peak_mid_peak_fix_max_monthly_diff', 'off_peak_mid_peak_fix_max_monthly_diff'
]


df_cleaned = df.drop(columns=columns_to_drop)


print(df_cleaned.columns)
```

```
Index(['id', 'channel_sales', 'cons_12m', 'cons_gas_12m', 'cons_last_month',
       'date_activ', 'date_end', 'date_renewal', 'forecast_cons_12m',
```

```
       'forecast_cons_year', 'forecast_meter_rent_12m',
       'forecast_price_energy_off_peak', 'forecast_price_energy_peak',
       'has_gas', 'margin_gross_pow_ele', 'margin_net_pow_ele', 'nb_prod_act',
       'net_margin', 'num_years_antig', 'origin_up', 'churn',
       'customer_tenure_days', 'time_to_renewal_days',
       'avg_monthly_consumption', 'forecast_error', 'consumption_change',
       'product_diversity'],
      dtype='object')
```

In [39]:
```
# Transforming into categorical type
df['channel_sales'] = df['channel_sales'].astype('category')
df['origin_up'] = df['origin_up'].astype('category')
```

In [40]:
```
# Let's see how many categories are within channel sales
df['channel_sales'].value_counts()
```

Out[40]:
```
channel_sales
foosdfpfkusacimwkcsosbicdxkicaua    6754
MISSING                             3725
lmkebamcaaclubfxadlmueccxoimlema    1843
usilxuppasemubllopkaafesmlibmsdf    1375
ewpakwlliwisiwduibdlfmalxowmwpci     893
sddiedcslfslkckwlfkdpoeeailfpeds      11
epumfxlbckeskwekxbiuasklxalciiuu       3
fixdbufsefwooaasfcxdxadsiekoceaa       2
Name: count, dtype: int64
```

In [42]:
```
# Let's see how many categories are within origin
df['origin_up'].value_counts()
```

Out[42]:
```
origin_up
lxidpiddsbxsbosboudacockeimpuepw    7097
kamkkxfxxuwbdslkwifmmcsiusiuosws    4294
ldkssxwpmemidmecebumciepifcamkci    3148
MISSING                               64
usapbepcfoloekilkwsdiboslwaxobdp       2
ewxeelcelemmiwuafmddpobolfuxioce       1
Name: count, dtype: int64
```

In [43]:
```
#creating dummy variables
df = pd.get_dummies(df, columns=['channel_sales'], prefix='channel')
df = pd.get_dummies(df, columns=['origin_up'], prefix='origin_up')
```

In [44]:
```
#dropping less frequent categories
df = df.drop(columns=['channel_sddiedcslfslkckwlfkdpoeeailfpeds', 'channel_epumfxlbckesk
df.head()
```

Out[44]:

| | id | cons_12m | cons_gas_12m | cons_last_month | date_activ | date_end | date_mod |
|---|---|---|---|---|---|---|---|
| **0** | 24011ae4ebbe3035111d65fa7c15bc57 | 0 | 54946 | 0 | 2013-06-15 | 2016-06-15 | 201 |
| **1** | d29c2c54acc38ff3c0614d0a653813dd | 4660 | 0 | 0 | 2009-08-21 | 2016-08-30 | 200 |
| **2** | 764c75f661154dac3a6c254cd082ea7d | 544 | 0 | 0 | 2010-04-16 | 2016-04-16 | 201 |
| **3** | bba03439a292a1e166f80264c16191cb | 1584 | 0 | 0 | 2010-03-30 | 2016-03-30 | 201 |
| **4** | 149d57cf92fc41cf94415803a877cb4b | 4425 | 0 | 526 | 2010-01-13 | 2016-03-07 | 201 |

5 rows × 67 columns

In [45]:
```
#dropping less frequent categories
df = df.drop(columns=['origin_up_MISSING', 'origin_up_usapbepcfoloekilkwsdiboslwaxobdp',
df.head()
```

Out[45]:

| | id | cons_12m | cons_gas_12m | cons_last_month | date_activ | date_end | date_mod |
|---|---|---|---|---|---|---|---|
| **0** | 24011ae4ebbe3035111d65fa7c15bc57 | 0 | 54946 | 0 | 2013-06-15 | 2016-06-15 | 201 |
| **1** | d29c2c54acc38ff3c0614d0a653813dd | 4660 | 0 | 0 | 2009-08-21 | 2016-08-30 | 200 |
| **2** | 764c75f661154dac3a6c254cd082ea7d | 544 | 0 | 0 | 2010-04-16 | 2016-04-16 | 201 |
| **3** | bba03439a292a1e166f80264c16191cb | 1584 | 0 | 0 | 2010-03-30 | 2016-03-30 | 201 |
| **4** | 149d57cf92fc41cf94415803a877cb4b | 4425 | 0 | 526 | 2010-01-13 | 2016-03-07 | 201 |

5 rows × 64 columns

In [46]:
```
# Seperating numerical columns
numerical_columns = df.select_dtypes(include=['number']).columns.tolist()


print("Numerical Columns:")
print(numerical_columns)
```

```
Numerical Columns:
['cons_12m', 'cons_gas_12m', 'cons_last_month', 'forecast_cons_12m', 'forecast_cons_yea
r', 'forecast_discount_energy', 'forecast_meter_rent_12m', 'forecast_price_energy_off_pe
ak', 'forecast_price_energy_peak', 'forecast_price_pow_off_peak', 'has_gas', 'imp_cons',
'margin_gross_pow_ele', 'margin_net_pow_ele', 'nb_prod_act', 'net_margin', 'num_years_an
tig', 'pow_max', 'var_year_price_off_peak_var', 'var_year_price_peak_var', 'var_year_pri
ce_mid_peak_var', 'var_year_price_off_peak_fix', 'var_year_price_peak_fix', 'var_year_pr
ice_mid_peak_fix', 'var_year_price_off_peak', 'var_year_price_peak', 'var_year_price_mid
_peak', 'var_6m_price_off_peak_var', 'var_6m_price_peak_var', 'var_6m_price_mid_peak_va
r', 'var_6m_price_off_peak_fix', 'var_6m_price_peak_fix', 'var_6m_price_mid_peak_fix',
'var_6m_price_off_peak', 'var_6m_price_peak', 'var_6m_price_mid_peak', 'churn', 'offpeak
_diff_dec_january_energy', 'offpeak_diff_dec_january_power', 'off_peak_peak_var_max_mont
hly_diff', 'peak_mid_peak_var_max_monthly_diff', 'off_peak_mid_peak_var_max_monthly_dif
f', 'off_peak_peak_fix_max_monthly_diff', 'peak_mid_peak_fix_max_monthly_diff', 'off_pea
k_mid_peak_fix_max_monthly_diff', 'customer_tenure_days', 'time_to_renewal_days', 'avg_m
onthly_consumption', 'forecast_error', 'consumption_change', 'product_diversity']
```

In [50]:
```
# Display the columns with high skewness and their skewness values
skewness = df[numerical_columns].skew()

high_skew_columns = skewness[abs(skewness) > 1].index.tolist()


print("Numerical Columns with High Skewness:")
print(skewness[abs(skewness) > 1])
```

```
Numerical Columns with High Skewness:
cons_12m                           5.997308
cons_gas_12m                       9.597530
cons_last_month                    6.391407
forecast_cons_12m                  7.155853
forecast_cons_year                16.587990
forecast_discount_energy           5.155098
forecast_meter_rent_12m            1.505148
forecast_price_pow_off_peak       -4.998772
```

```
has_gas                                  1.652855
imp_cons                                13.198799
margin_gross_pow_ele                     4.472632
margin_net_pow_ele                       4.473326
nb_prod_act                              8.636878
net_margin                              36.569515
num_years_antig                          1.446214
pow_max                                  5.786785
var_year_price_off_peak_var             12.052339
var_year_price_peak_var                  8.979052
var_year_price_mid_peak_var              8.720249
var_year_price_off_peak_fix             22.051551
var_year_price_peak_fix                  7.908174
var_year_price_mid_peak_fix              7.907288
var_year_price_off_peak                 22.051457
var_year_price_peak                      7.908207
var_year_price_mid_peak                  7.907295
var_6m_price_off_peak_var               16.954975
var_6m_price_peak_var                    8.948396
var_6m_price_mid_peak_var               10.072325
var_6m_price_off_peak_fix               22.886924
var_6m_price_peak_fix                   10.721053
var_6m_price_mid_peak_fix                9.862636
var_6m_price_off_peak                   22.886911
var_6m_price_peak                       10.721069
var_6m_price_mid_peak                    9.862635
churn                                    2.720715
offpeak_diff_dec_january_energy          3.377645
offpeak_diff_dec_january_power          -9.278376
peak_mid_peak_fix_max_monthly_diff       1.883115
customer_tenure_days                     1.252599
avg_monthly_consumption                  5.997308
forecast_error                          -6.002940
consumption_change                       7.301251
product_diversity                        4.169416
dtype: float64
```

In [51]:

```
# Applying transformations to reduce skewness
for col in high_skew_columns:
    if df[col].min() > 0:

        df[col] = np.log1p(df[col])
    elif df[col].min() == 0:

        df[col] = np.sqrt(df[col])
    else:

        df[col] = np.cbrt(df[col])
```

In [52]:

```
# Verifying the transformations by recalculating skewness
new_skewness = df[high_skew_columns].skew()
print("Skewness after transformation:")
print(new_skewness)
```

```
Skewness after transformation:
cons_12m                                 3.559270
cons_gas_12m                             5.162885
cons_last_month                          3.665623
forecast_cons_12m                        1.190263
forecast_cons_year                       1.552367
forecast_discount_energy                 5.086939
forecast_meter_rent_12m                  0.478535
forecast_price_pow_off_peak             -9.787009
has_gas                                  1.652855
imp_cons                                 1.484692
```

```
margin_gross_pow_ele                   0.866097
margin_net_pow_ele                     0.866350
nb_prod_act                            2.247376
net_margin                             1.796784
num_years_antig                        0.376138
pow_max                                1.804466
var_year_price_off_peak_var            6.457782
var_year_price_peak_var                4.316402
var_year_price_mid_peak_var            5.679938
var_year_price_off_peak_fix           15.039135
var_year_price_peak_fix                6.039512
var_year_price_mid_peak_fix            6.080530
var_year_price_off_peak               15.047306
var_year_price_peak                    6.039818
var_year_price_mid_peak                6.080560
var_6m_price_off_peak_var             10.144648
var_6m_price_peak_var                  6.450470
var_6m_price_mid_peak_var              8.769017
var_6m_price_off_peak_fix             19.018426
var_6m_price_peak_fix                  9.181933
var_6m_price_mid_peak_fix              9.095326
var_6m_price_off_peak                 19.027813
var_6m_price_peak                      9.182055
var_6m_price_mid_peak                  9.095334
churn                                  2.720715
offpeak_diff_dec_january_energy        3.897305
offpeak_diff_dec_january_power        -0.460047
peak_mid_peak_fix_max_monthly_diff     0.633225
customer_tenure_days                   0.126074
avg_monthly_consumption                3.559270
forecast_error                        -2.685337
consumption_change                     1.109733
product_diversity                      1.739422
dtype: float64
```

In [60]:

```python
# New list of Columns with high skewness
new_high_skew_columns = [
    'cons_12m', 'cons_gas_12m', 'cons_last_month', 'forecast_discount_energy',
    'forecast_price_pow_off_peak', 'nb_prod_act',
     'var_year_price_off_peak_var', 'var_year_price_peak_var',
    'var_year_price_mid_peak_var', 'var_year_price_off_peak_fix',
    'var_year_price_peak_fix', 'var_year_price_mid_peak_fix',
    'var_year_price_off_peak', 'var_year_price_peak',
    'var_year_price_mid_peak', 'var_6m_price_off_peak_var',
    'var_6m_price_peak_var', 'var_6m_price_mid_peak_var',
    'var_6m_price_off_peak_fix', 'var_6m_price_peak_fix',
    'var_6m_price_mid_peak_fix', 'var_6m_price_off_peak',
    'var_6m_price_peak', 'var_6m_price_mid_peak',
    'offpeak_diff_dec_january_energy',

    'avg_monthly_consumption',
    'forecast_error']
```

In [61]:

```python
# Applying Box-Cox or Yeo-Johnson transformation to each skewed column
for col in new_high_skew_columns:
    if df[col].min() > 0:

        df[col], _ = stats.boxcox(df[col])
    else:

        df[col] = stats.yeojohnson(df[col])[0]
```

In [62]:

```python
# Verify the transformations by recalculating skewness
new_skewness = df[high_skew_columns].skew()
```

```
print("Skewness after aggressive transformation:")
print(new_skewness)
```

```
Skewness after aggressive transformation:
cons_12m                              0.055097
cons_gas_12m                          1.677775
cons_last_month                      -0.059593
forecast_cons_12m                     1.190263
forecast_cons_year                    1.552367
forecast_discount_energy              5.056572
forecast_meter_rent_12m               0.478535
forecast_price_pow_off_peak           1.097363
has_gas                               1.652855
imp_cons                              1.484692
margin_gross_pow_ele                  0.866097
margin_net_pow_ele                    0.866350
nb_prod_act                           1.370963
net_margin                            1.796784
num_years_antig                       0.376138
pow_max                               1.804466
var_year_price_off_peak_var           0.385293
var_year_price_peak_var               0.895232
var_year_price_mid_peak_var           1.443285
var_year_price_off_peak_fix           0.593984
var_year_price_peak_fix               2.237046
var_year_price_mid_peak_fix           2.461060
var_year_price_off_peak               0.623190
var_year_price_peak                   2.203034
var_year_price_mid_peak               2.462765
var_6m_price_off_peak_var             0.253658
var_6m_price_peak_var                 0.894766
var_6m_price_mid_peak_var             2.292272
var_6m_price_off_peak_fix             1.001696
var_6m_price_peak_fix                 2.594947
var_6m_price_mid_peak_fix             2.805353
var_6m_price_off_peak                 1.016863
var_6m_price_peak                     2.462900
var_6m_price_mid_peak                 2.803790
churn                                 2.720715
offpeak_diff_dec_january_energy      -1.652828
offpeak_diff_dec_january_power       -0.460047
peak_mid_peak_fix_max_monthly_diff    0.633225
customer_tenure_days                  0.126074
avg_monthly_consumption              -0.030519
forecast_error                       -0.331135
consumption_change                    1.109733
product_diversity                     1.739422
dtype: float64
```

In [65]:

```
#getting correlation co-efficient for columns
correlation = df[numerical_columns].corr()
correlation
```

Out[65]:

| | cons_12m | cons_gas_12m | cons_last_month | forecast_cons_12m | forecast_c |
|---|---|---|---|---|---|
| **cons_12m** | 1.000000 | 0.160613 | 0.659207 | 0.579439 | |
| **cons_gas_12m** | 0.160613 | 1.000000 | 0.135489 | 0.118934 | |
| **cons_last_month** | 0.659207 | 0.135489 | 1.000000 | 0.377800 | |
| **forecast_cons_12m** | 0.579439 | 0.118934 | 0.377800 | 1.000000 | |
| **forecast_cons_year** | 0.422271 | 0.088391 | 0.695472 | 0.629237 | |
| **forecast_discount_energy** | -0.030005 | 0.007259 | -0.016262 | 0.064141 | |

| | | | | | |
|---|---|---|---|---|---|
| **forecast_meter_rent_12m** | 0.219388 | 0.053156 | 0.358914 | 0.297272 | |
| **forecast_price_energy_off_peak** | -0.144358 | -0.033294 | -0.230743 | -0.119540 | - |
| **forecast_price_energy_peak** | 0.283816 | 0.059723 | 0.395257 | 0.297744 | |
| **forecast_price_pow_off_peak** | -0.206694 | -0.044271 | -0.205496 | -0.102680 | - |
| **has_gas** | 0.160948 | 0.966319 | 0.139494 | 0.122026 | |
| **imp_cons** | 0.399666 | 0.088494 | 0.690902 | 0.625686 | |
| **margin_gross_pow_ele** | -0.066618 | -0.028359 | 0.008605 | -0.130401 | - |
| **margin_net_pow_ele** | -0.066566 | -0.028336 | 0.008548 | -0.130403 | - |
| **nb_prod_act** | 0.140620 | 0.869259 | 0.122243 | 0.116757 | |
| **net_margin** | 0.554768 | 0.122150 | 0.373810 | 0.929794 | |
| **num_years_antig** | 0.003825 | 0.001296 | 0.020645 | -0.024161 | |
| **pow_max** | 0.237867 | 0.064747 | 0.314291 | 0.380584 | |
| **var_year_price_off_peak_var** | 0.103449 | 0.028620 | 0.208585 | 0.194087 | |
| **var_year_price_peak_var** | 0.151417 | 0.024511 | 0.232742 | 0.183020 | |
| **var_year_price_mid_peak_var** | 0.220727 | 0.060222 | 0.316686 | 0.263169 | |
| **var_year_price_off_peak_fix** | -0.057061 | -0.024391 | -0.033358 | -0.088781 | - |
| **var_year_price_peak_fix** | 0.138527 | 0.032079 | 0.237889 | 0.130844 | |
| **var_year_price_mid_peak_fix** | 0.124972 | 0.030862 | 0.222074 | 0.110617 | |
| **var_year_price_off_peak** | -0.057201 | -0.024247 | -0.032885 | -0.088356 | - |
| **var_year_price_peak** | 0.131557 | 0.028693 | 0.228690 | 0.125165 | |
| **var_year_price_mid_peak** | 0.126884 | 0.031688 | 0.224712 | 0.113445 | |
| **var_6m_price_off_peak_var** | 0.151813 | 0.044439 | 0.249595 | 0.229738 | |
| **var_6m_price_peak_var** | 0.182991 | 0.046327 | 0.287486 | 0.224213 | |
| **var_6m_price_mid_peak_var** | 0.115324 | 0.046389 | 0.193041 | 0.156786 | |
| **var_6m_price_off_peak_fix** | -0.112461 | -0.012190 | -0.103879 | -0.119007 | - |
| **var_6m_price_peak_fix** | 0.067018 | 0.031339 | 0.149733 | 0.063802 | |
| **var_6m_price_mid_peak_fix** | 0.060352 | 0.030335 | 0.142907 | 0.051181 | |
| **var_6m_price_off_peak** | -0.110537 | -0.011646 | -0.100529 | -0.116381 | - |
| **var_6m_price_peak** | 0.065947 | 0.030438 | 0.150236 | 0.065584 | |
| **var_6m_price_mid_peak** | 0.061834 | 0.030763 | 0.144807 | 0.053917 | |
| **churn** | -0.018295 | -0.019852 | -0.013357 | 0.011573 | |
| **offpeak_diff_dec_january_energy** | -0.066653 | -0.006480 | -0.117043 | -0.080559 | - |
| **offpeak_diff_dec_january_power** | -0.078746 | -0.024825 | -0.051600 | -0.108595 | - |
| **off_peak_peak_var_max_monthly_diff** | -0.278193 | -0.059088 | -0.399173 | -0.276100 | - |
| **peak_mid_peak_var_max_monthly_diff** | 0.177338 | 0.013575 | 0.183758 | 0.136881 | |
| **off_peak_mid_peak_var_max_monthly_diff** | -0.195777 | -0.051785 | -0.325986 | -0.218687 | - |
| **off_peak_peak_fix_max_monthly_diff** | -0.215428 | -0.054398 | -0.348286 | -0.229404 | - |
| **peak_mid_peak_fix_max_monthly_diff** | 0.206879 | 0.054602 | 0.364106 | 0.279248 | |

| | | | | | |
|---|---|---|---|---|---|
| **off_peak_mid_peak_fix_max_monthly_diff** | -0.224868 | -0.053909 | -0.330674 | -0.200394 | - |
| **customer_tenure_days** | -0.024829 | -0.000078 | -0.008917 | -0.043679 | |
| **time_to_renewal_days** | 0.073379 | 0.008738 | 0.064462 | -0.034008 | |
| **avg_monthly_consumption** | 0.998890 | 0.157760 | 0.658212 | 0.592153 | |
| **forecast_error** | -0.949679 | -0.175019 | -0.656051 | -0.491186 | - |
| **consumption_change** | 0.272095 | 0.128140 | 0.588541 | 0.092840 | |
| **product_diversity** | 0.141898 | 0.911878 | 0.125708 | 0.109825 | |

51 rows × 51 columns

In [67]:
```
# Displaying features with high VIF
X = df.select_dtypes(include=[np.number]).drop(columns=['churn'])  # Exclude the target


X = add_constant(X)


vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]


high_vif = vif_data[vif_data["VIF"] > 5]
print("Features with high VIF:")
print(high_vif)
```

```
Features with high VIF:
                          Feature           VIF
0                           const   6763.942567
1                        cons_12m    774.144374
2                    cons_gas_12m     15.199347
4               forecast_cons_12m     12.742537
5              forecast_cons_year     50.844690
7         forecast_meter_rent_12m      5.211478
8     forecast_price_energy_off_peak     22.418976
9         forecast_price_energy_peak     94.214214
10        forecast_price_pow_off_peak     17.740544
11                        has_gas     22.331307
12                       imp_cons     49.950572
13             margin_gross_pow_ele   3988.869868
14               margin_net_pow_ele   3986.076811
15                    nb_prod_act     12.791238
16                     net_margin     10.433035
17                num_years_antig     14.586940
19        var_year_price_off_peak_var      6.223403
20            var_year_price_peak_var     27.583380
21        var_year_price_mid_peak_var     11.663679
22        var_year_price_off_peak_fix   2200.717906
23            var_year_price_peak_fix   1058.558186
24        var_year_price_mid_peak_fix  11779.131222
25            var_year_price_off_peak   2188.594627
26                var_year_price_peak    788.210691
27            var_year_price_mid_peak  11196.654074
28          var_6m_price_off_peak_var      5.723145
29              var_6m_price_peak_var     22.798143
30          var_6m_price_mid_peak_var      7.165353
31          var_6m_price_off_peak_fix   3254.322868
32              var_6m_price_peak_fix    594.220354
33          var_6m_price_mid_peak_fix   3338.598838
```

```
34                   var_6m_price_off_peak   3222.020114
35                       var_6m_price_peak    147.735258
36                   var_6m_price_mid_peak   3005.382767
39      off_peak_peak_var_max_monthly_diff    260.072586
40      peak_mid_peak_var_max_monthly_diff     54.792764
41  off_peak_mid_peak_var_max_monthly_diff    185.690230
42      off_peak_peak_fix_max_monthly_diff    319.160017
43      peak_mid_peak_fix_max_monthly_diff     61.126031
44  off_peak_mid_peak_fix_max_monthly_diff    209.769810
45                     customer_tenure_days    15.502732
47                  avg_monthly_consumption   700.472971
48                           forecast_error    14.868374
50                        product_diversity    20.498913
```

In [69]:

```python
# Dropping the highly multicollinear features from the original DataFrame
vif_threshold = 100


features_to_drop = high_vif[high_vif["VIF"] > vif_threshold]["Feature"].tolist()

if 'const' in features_to_drop:
    features_to_drop.remove('const')


df_reduced = df.drop(columns=features_to_drop)

print("Dropped features:", features_to_drop)
print("Reduced DataFrame shape:", df_reduced.shape)
```

```
Dropped features: ['cons_12m', 'margin_gross_pow_ele', 'margin_net_pow_ele', 'var_year_p
rice_off_peak_fix', 'var_year_price_peak_fix', 'var_year_price_mid_peak_fix', 'var_year_
price_off_peak', 'var_year_price_peak', 'var_year_price_mid_peak', 'var_6m_price_off_pea
k_fix', 'var_6m_price_peak_fix', 'var_6m_price_mid_peak_fix', 'var_6m_price_off_peak',
'var_6m_price_peak', 'var_6m_price_mid_peak', 'off_peak_peak_var_max_monthly_diff', 'off
_peak_mid_peak_var_max_monthly_diff', 'off_peak_peak_fix_max_monthly_diff', 'off_peak_mi
d_peak_fix_max_monthly_diff', 'avg_monthly_consumption']
Reduced DataFrame shape: (14606, 44)
```

In [70]:

```python
#examining the new modified dataset
df.head()
```

Out[70]:

| | id | cons_12m | cons_gas_12m | cons_last_month | date_activ | date_end | date_mod |
|---|---|---|---|---|---|---|---|
| **0** | 24011ae4ebbe3035111d65fa7c15bc57 | 0.000000 | 0.92993 | 0.00000 | 2013-06-15 | 2016-06-15 | 201 |
| **1** | d29c2c54acc38ff3c0614d0a653813dd | 4.704260 | -0.00000 | 0.00000 | 2009-08-21 | 2016-08-30 | 200 |
| **2** | 764c75f661154dac3a6c254cd082ea7d | 3.451355 | -0.00000 | 0.00000 | 2010-04-16 | 2016-04-16 | 201 |
| **3** | bba03439a292a1e166f80264c16191cb | 4.062661 | -0.00000 | 0.00000 | 2010-03-30 | 2016-03-30 | 201 |
| **4** | 149d57cf92fc41cf94415803a877cb4b | 4.672976 | -0.00000 | 3.55748 | 2010-01-13 | 2016-03-07 | 201 |

5 rows × 64 columns

In [71]:

```python
#saving new df as csv file
df = pd.DataFrame(df)


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