Report

Importing Packages

```
import pandas as pd
import numpy as np
```

Importing Datasets

```
data1 = pd.read_csv("ml_case_training_data.csv")
data2 = pd.read_csv("ml_case_training_hist_data.csv")
```

```
data3 = pd.read_csv("ml_case_training_output.csv")
```

Exploratory Data Analysis

Dataset Information

```
data1.head()
```

	id	activity_new	campaign_disc_ele
0	48ada52261e7cf58715202705a0451c9	esoiiifxdlbkcsluxmfuacbdckommixw	NaN
1	24011ae4ebbe3035111d65fa7c15bc57	NaN	NaN
2	d29c2c54acc38ff3c0614d0a653813dd	NaN	NaN
3	764c75f661154dac3a6c254cd082ea7d	NaN	NaN
4	bba03439a292a1e166f80264c16191cb	NaN	NaN

```
data2.head()
```

	id	price_date	price_p1_var	price_p2_var	price_p3_v
0	038af19179925da21a25619c5a24b745	2015-01-01	0.151367	0.0	
1	038af19179925da21a25619c5a24b745	2015-02-01	0.151367	0.0	
2	038af19179925da21a25619c5a24b745	2015-03-01	0.151367	0.0	

```
data3.tail()
```

	id	churn
16091	18463073fb097fc0ac5d3e040f356987	0
16092	d0a6f71671571ed83b2645d23af6de00	1
16093	10e6828ddd62cbcf687cb74928c4c2d2	1
16094	1cf20fd6206d7678d5bcafd28c53b4db	0
16095	563dde550fd624d7352f3de77c0cdfcd	0

```
data1.describe()
```

	campaign_disc_ele	cons_12m	cons_gas_12m	cons_last_month	forecast_base_b
count	0.0	1.609600e+04	1.609600e+04	1.609600e+04	3508
mean	NaN	1.948044e+05	3.191164e+04	1.946154e+04	335
std	NaN	6.795151e+05	1.775885e+05	8.235676e+04	649
min	NaN	-1.252760e+05	-3.037000e+03	-9.138600e+04	-364
25%	NaN	5.906250e+03	0.000000e+00	0.000000e+00	
50%	NaN	1.533250e+04	0.000000e+00	9.010000e+02	162
75%	NaN	5.022150e+04	0.000000e+00	4.127000e+03	396
max	NaN	1.609711e+07	4.188440e+06	4.538720e+06	12566

```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16096 entries, 0 to 16095
Data columns (total 32 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   id                        16096 non-null  object 
 1   activity_new              6551 non-null   object 
 2   campaign_disc_ele         0 non-null      float64
 3   channel_sales             11878 non-null  object 
 4   cons_12m                  16096 non-null  int64  
```

```
 5   cons_gas_12m              16096 non-null  int64
 6   cons_last_month           16096 non-null  int64
 7   date_activ                16096 non-null  object
 8   date_end                  16094 non-null  object
 9   date_first_activ          3508 non-null   object
 10  date_modif_prod           15939 non-null  object
 11  date_renewal              16056 non-null  object
 12  forecast_base_bill_ele    3508 non-null   float64
 13  forecast_base_bill_year   3508 non-null   float64
 14  forecast_bill_12m         3508 non-null   float64
 15  forecast_cons             3508 non-null   float64
 16  forecast_cons_12m         16096 non-null  float64
 17  forecast_cons_year        16096 non-null  int64
 18  forecast_discount_energy  15970 non-null  float64
 19  forecast_meter_rent_12m   16096 non-null  float64
 20  forecast_price_energy_p1  15970 non-null  float64
 21  forecast_price_energy_p2  15970 non-null  float64
 22  forecast_price_pow_p1     15970 non-null  float64
 23  has_gas                   16096 non-null  object
 24  imp_cons                  16096 non-null  float64
 25  margin_gross_pow_ele      16083 non-null  float64
 26  margin_net_pow_ele        16083 non-null  float64
 27  nb_prod_act               16096 non-null  int64
 28  net_margin                16081 non-null  float64
 29  num_years_antig           16096 non-null  int64
 30  origin_up                 16009 non-null  object
 31  pow_max                   16093 non-null  float64
dtypes: float64(16), int64(6), object(10)
memory usage: 3.9+ MB
```

```
data2.describe()
```

	price_p1_var	price_p2_var	price_p3_var	price_p1_fix	price_p2_fix	price_
count	181951.000000	181950.000000	181950.000000	181950.000000	181950.000000	181950
mean	0.140989	0.054327	0.030723	43.320736	10.699303	6
std	0.025118	0.050026	0.036337	5.432583	12.851347	7
min	0.000000	0.000000	0.000000	-0.177779	-0.097752	-0
25%	0.125976	0.000000	0.000000	40.728885	0.000000	0
50%	0.146033	0.085390	0.000000	44.266930	0.000000	0
75%	0.151635	0.101758	0.072558	44.444710	24.339581	16
max	0.280700	0.196782	0.114102	59.444710	36.490692	17

```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183235 entries, 0 to 183234
Data columns (total 8 columns):
 #   Column        Non-Null Count   Dtype
```

```
---  ------        --------------   -----
 0   id            183235 non-null  object
 1   price_date    183235 non-null  object
 2   price_p1_var  181951 non-null  float64
 3   price_p2_var  181950 non-null  float64
 4   price_p3_var  181950 non-null  float64
 5   price_p1_fix  181950 non-null  float64
 6   price_p2_fix  181950 non-null  float64
 7   price_p3_fix  181950 non-null  float64
dtypes: float64(6), object(2)
memory usage: 11.2+ MB
```

```
data3.describe()
```

	churn
count	16096.000000
mean	0.099093
std	0.298796
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

```
data3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16096 entries, 0 to 16095
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      16096 non-null  object
 1   churn   16096 non-null  int64
dtypes: int64(1), object(1)
memory usage: 251.6+ KB
```

Finding total missing values

```
data1.isnull().sum()
```

```
id                              0
activity_new                 9545
campaign_disc_ele           16096
channel_sales                4218
cons_12m                        0
```

```
cons_gas_12m                  0
cons_last_month               0
date_activ                    0
date_end                      2
date_first_activ          12588
date_modif_prod             157
date_renewal                 40
forecast_base_bill_ele    12588
forecast_base_bill_year   12588
forecast_bill_12m         12588
forecast_cons             12588
forecast_cons_12m             0
forecast_cons_year            0
forecast_discount_energy    126
forecast_meter_rent_12m       0
forecast_price_energy_p1    126
forecast_price_energy_p2    126
forecast_price_pow_p1       126
has_gas                       0
imp_cons                      0
margin_gross_pow_ele         13
margin_net_pow_ele           13
nb_prod_act                   0
net_margin                   15
num_years_antig               0
origin_up                    87
pow_max                       3
dtype: int64
```

```
data2.isnull().sum()
```

```
id                0
price_date        0
price_p1_var    1284
price_p2_var    1285
price_p3_var    1285
price_p1_fix    1285
price_p2_fix    1285
price_p3_fix    1285
dtype: int64
```

```
data3.isnull().sum()
```

```
id       0
churn    0
dtype: int64
```

Data Visuals for missing entries

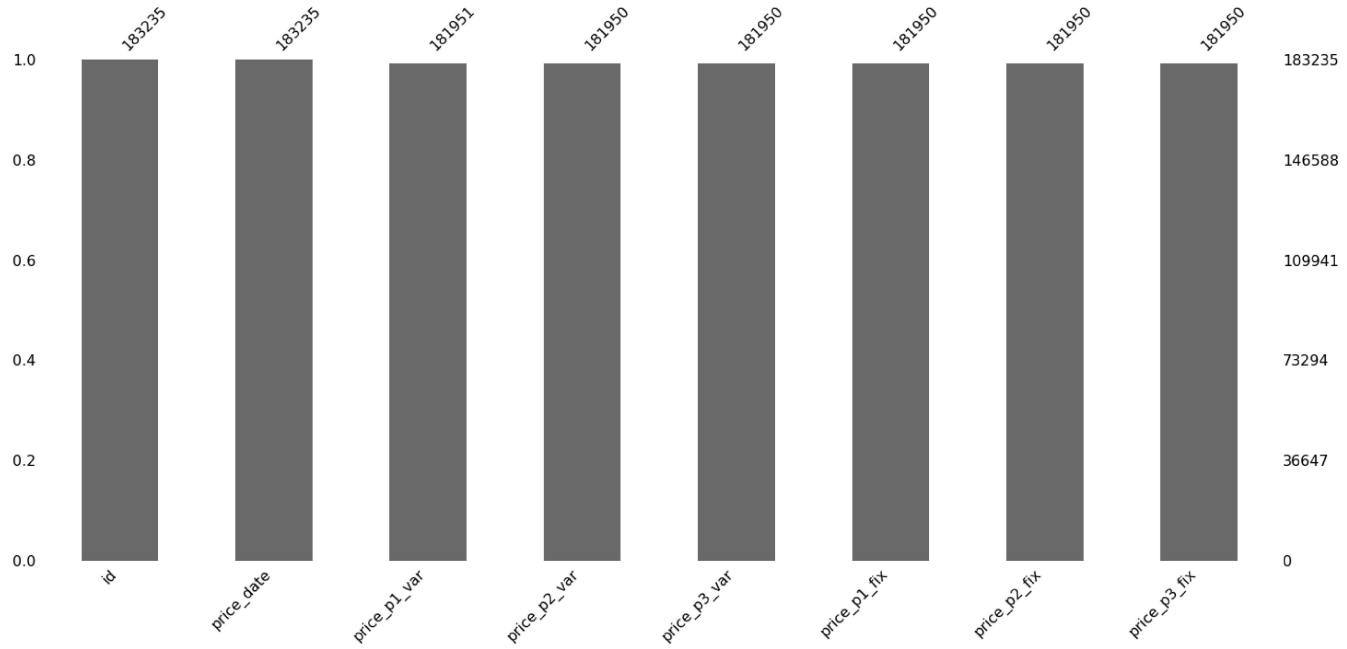
Importing necessary packages

```
import seaborn as sns
```

```
import missingno as msno
```

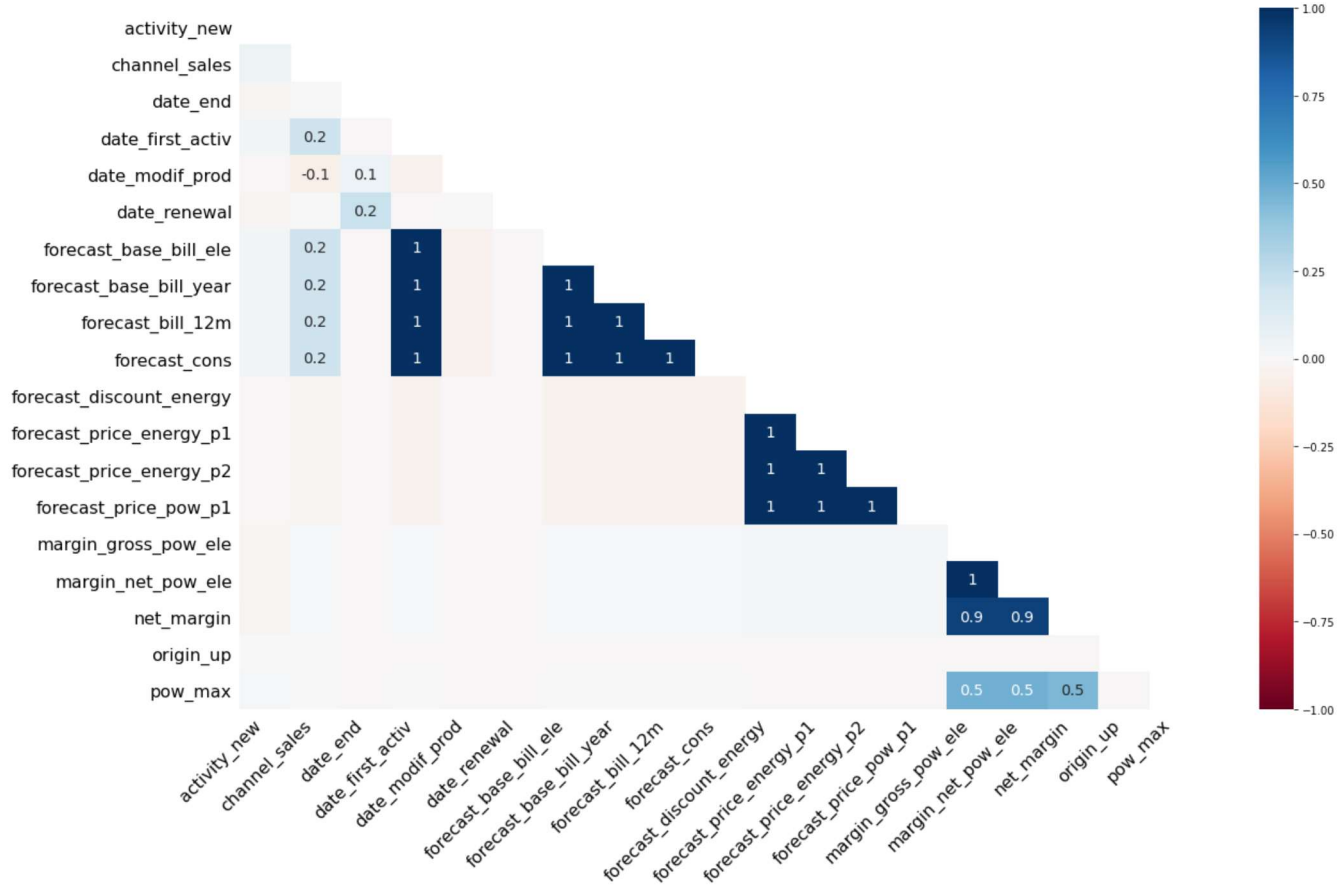
```
msno.bar(data2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4de96f69d0>
```



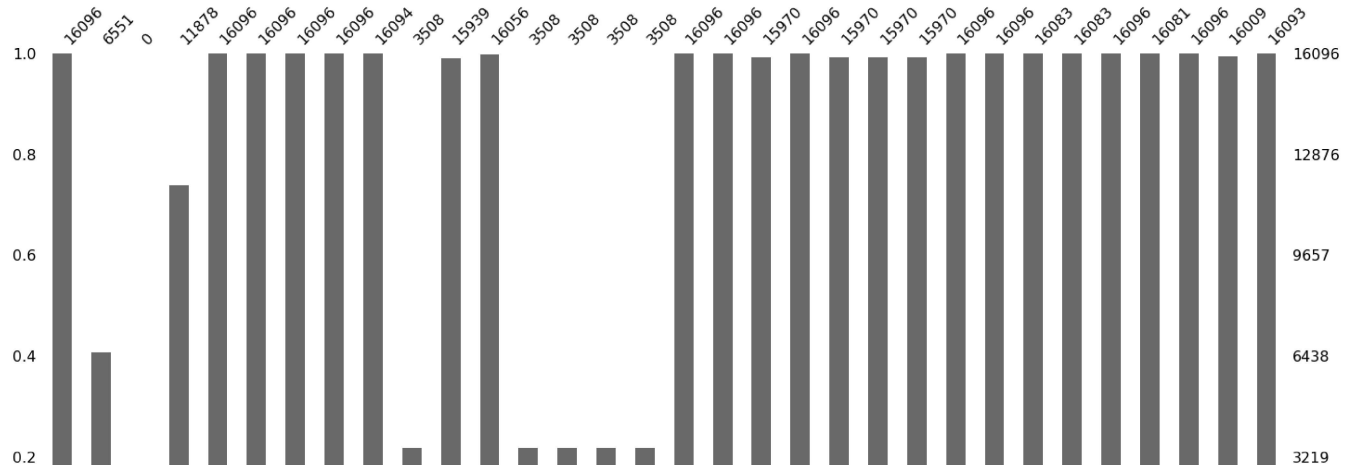

```
msno.heatmap(data1)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4dddb15050>
```

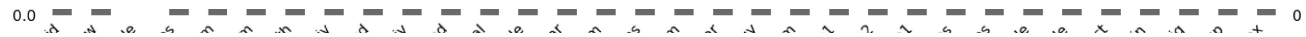



```
msno.bar(data1)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ddd6bf5d0>
```



Rectifying negative price values

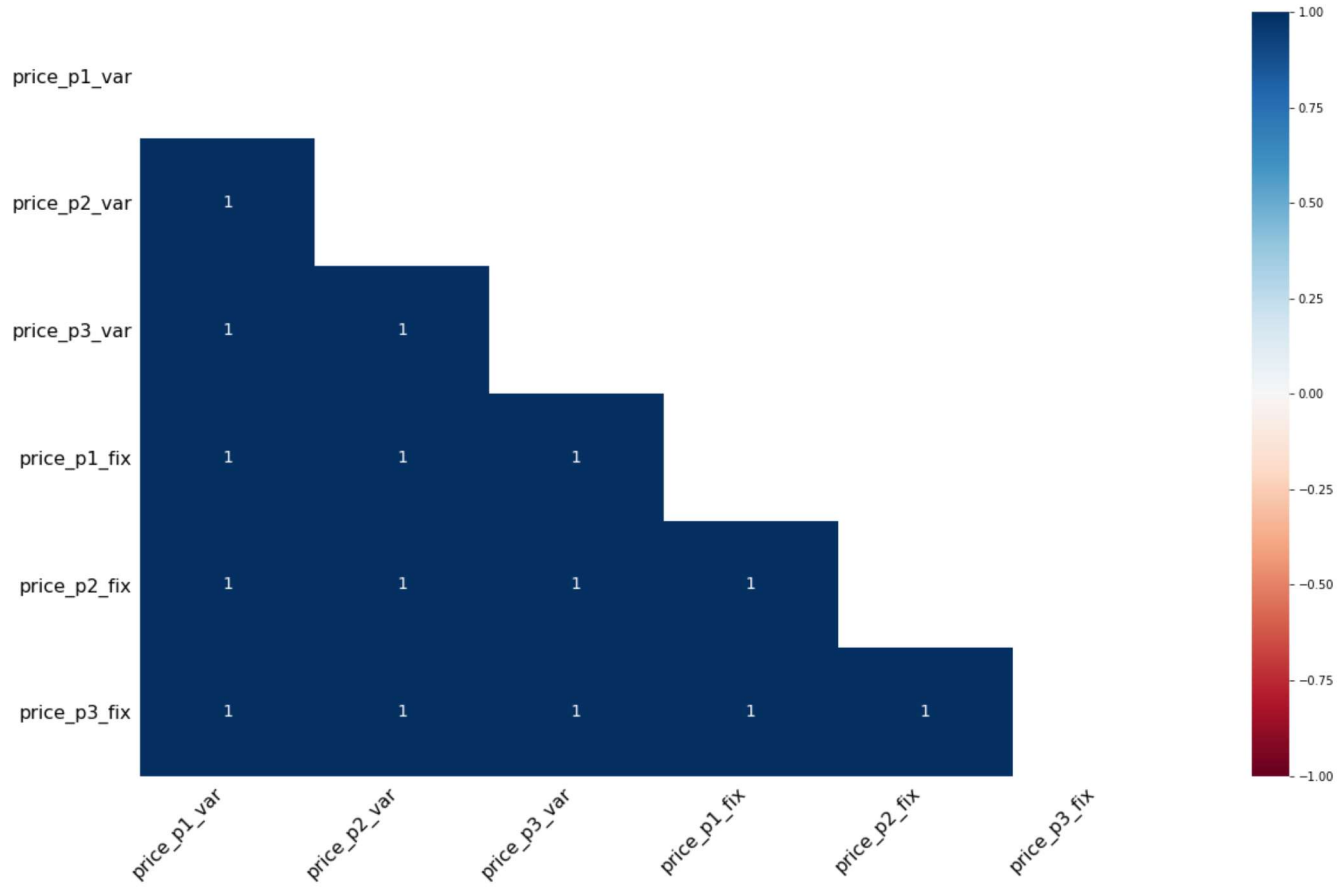



```
# Identify negative columns
negative_cols = ['price_p1_fix','price_p2_fix','price_p3_fix']

# Convert to positive the negative columns in pco_hist
data2[negative_cols] = data2[negative_cols].apply(abs)

#pco_hist.describe()


data2.describe()
```

	price_p1_var	price_p2_var	price_p3_var	price_p1_fix	price_p2_fix	price_
count	191643.000000	191643.000000	191643.000000	191643.000000	191643.000000	191643
mean	0.140991	0.054412	0.030712	43.325563	10.698210	6
std	0.025117	0.050033	0.036335	5.437816	12.856039	7
min	0.000000	0.000000	0.000000	0.000000	0.000000	0
25%	0.125976	0.000000	0.000000	40.728885	0.000000	0
50%	0.146033	0.085483	0.000000	44.266930	0.000000	0
75%	0.151635	0.101780	0.072558	44.444710	24.339581	16
max	0.280700	0.229788	0.114102	59.444710	36.490692	17

```
msno.heatmap(data2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcfc9cc84d0>
```



Data Pre Processing

Treating Missing Values

```
# Identify the index of the IDs containing missing values.
data2_NAN_index = data2[data2.isnull().any(axis=1)].index.values.tolist()

# Obtain a dataframe with the missing values
data2_missing = data2.iloc[data2_NAN_index,:]


data2_missing.head()
```

	id	price_date	price_p1_var	price_p2_var	price_p3
75	ef716222bbd97a8bdfcbb831e3575560	2015-04-01	NaN	NaN	
221	0f5231100b2febab862f8dd8eaab3f43	2015-06-01	NaN	NaN	
377	2f93639de582fadfbe3e86ce1c8d8f35	2015-06-01	NaN	NaN	
413	f83c1ab1ca1d1802bb1df4d72820243c	2015-06-01	NaN	NaN	
461	3076c6d4a060e12a049d1700d9b09cf3	2015-06-01	NaN	NaN	

```
# extract the unique dates of missing data
date_lst = data2_missing['price_date'].unique()
id_lst = data2_missing['id'].unique()

# Create a time dataframe with the unique dates
time_df = pd.DataFrame(data=date_lst, columns=['price_date'] )

# Glimpse the time dataframe
time_df.sort_values(by=['price_date'])
```

	price_date
9	2015-01-01
11	2015-02-01
8	2015-03-01
0	2015-04-01
2	2015-05-01
1	2015-06-01
10	2015-07-01
3	2015-08-01
4	2015-09-01
7	2015-10-01
6	2015-11-01
5	2015-12-01

Imputing Missing Values

```
# Make a copy of pco_hist dataset
data2_ff = data2.copy(deep=True)

# Print prior to imputing missing values
print(data2_ff.iloc[data2_NAN_index,3:9].head())
```

```
# Fill NaNs using forward fill
data2_ff.fillna(method = 'ffill', inplace=True)

print(data2_ff.iloc[data2_NAN_index,3:9].head())
```

```
     price_p2_var  price_p3_var  price_p1_fix  price_p2_fix  price_p3_fix
75            NaN           NaN           NaN           NaN           NaN
221           NaN           NaN           NaN           NaN           NaN
377           NaN           NaN           NaN           NaN           NaN
413           NaN           NaN           NaN           NaN           NaN
461           NaN           NaN           NaN           NaN           NaN
     price_p2_var  price_p3_var  price_p1_fix  price_p2_fix  price_p3_fix
75       0.000000      0.000000     44.266931      0.000000      0.000000
221      0.000000      0.000000     44.266931      0.000000      0.000000
377      0.087970      0.000000     44.266931      0.000000      0.000000
413      0.102239      0.070381     40.565969     24.339581     16.226389
461      0.000000      0.000000     44.266931      0.000000      0.000000
```

```
data2.describe()
```

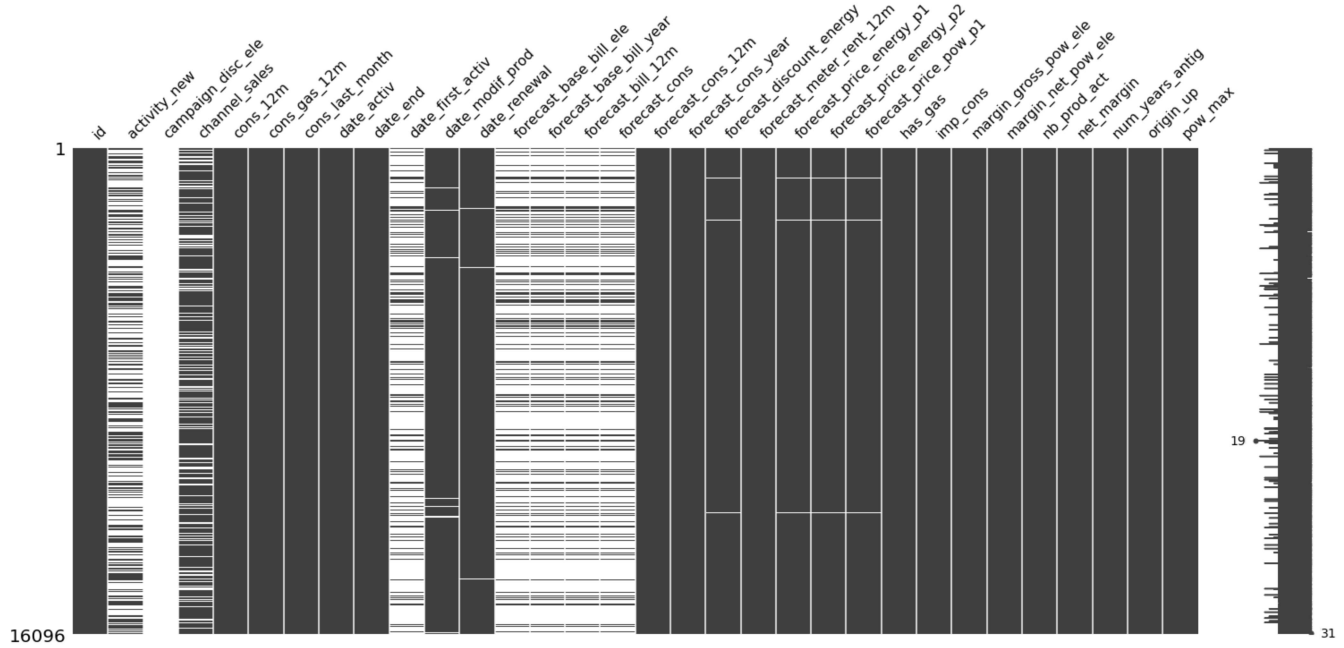
	price_p1_var	price_p2_var	price_p3_var	price_p1_fix	price_p2_fix	price_
count	191643.000000	191643.000000	191643.000000	191643.000000	191643.000000	191643
mean	0.140991	0.054412	0.030712	43.325563	10.698210	6
std	0.025117	0.050033	0.036335	5.437816	12.856039	7
min	0.000000	0.000000	0.000000	0.000000	0.000000	0
25%	0.125976	0.000000	0.000000	40.728885	0.000000	0
50%	0.146033	0.085483	0.000000	44.266930	0.000000	0
75%	0.151635	0.101780	0.072558	44.444710	24.339581	16
max	0.280700	0.229788	0.114102	59.444710	36.490692	17

```
# Merge output dataset with historical forward fill dataset
data2_ff_merged = data2_ff.merge(right=data2,on=['id'])

data2_ff_merged.head()
```

```
                                        id  price_date_x  price_p1_var_x  price_p2_var_x  pric
```

```
msno.matrix(data1)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcfbf764110>
```



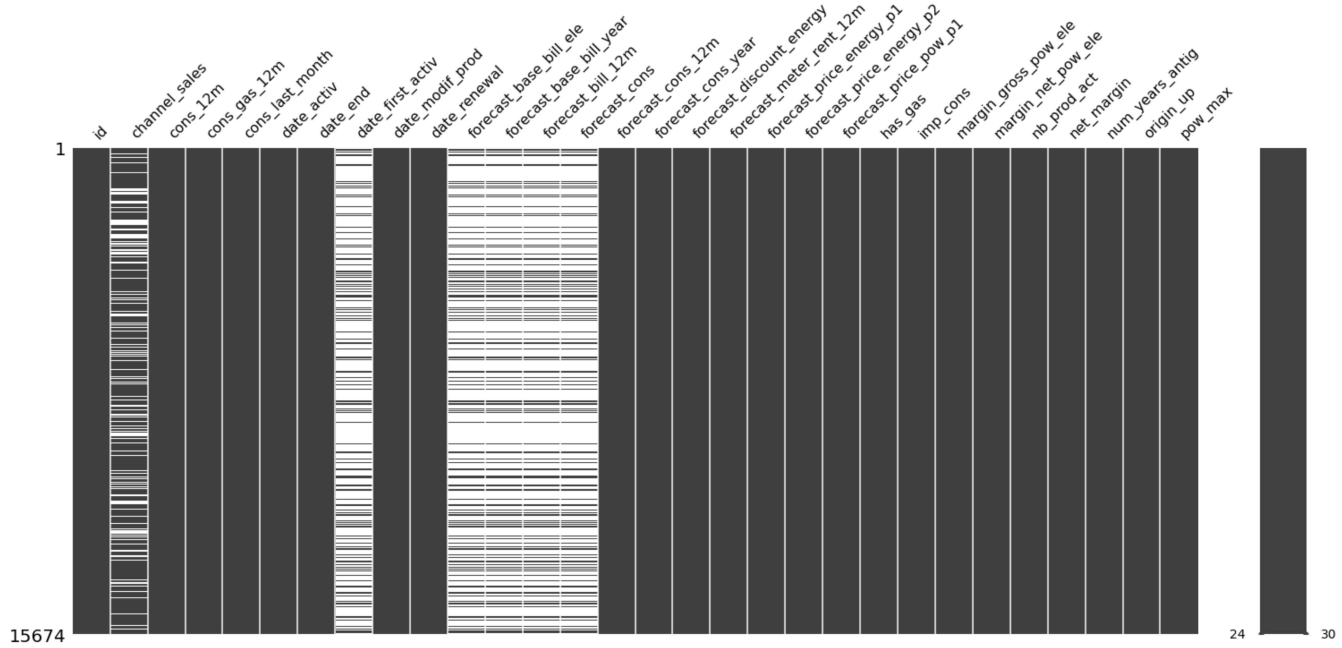
Removing Irrelevant Columns

```
# Drop the column activity_new and campaign_disc_elec
data1_drop = data1.drop(labels= ['activity_new','campaign_disc_ele'] , axis=1)

# Remove date_end date_modif_prod date_renewal origin_up pow_max margin_gross_pow_ele margin_
brush = ['date_end','date_modif_prod','date_renewal','origin_up','pow_max','margin_gross_pow_
         'margin_net_pow_ele', 'net_margin','forecast_discount_energy','forecast_price_energy
         'forecast_price_energy_p2','forecast_price_pow_p1']
data1_drop.dropna(subset=brush, how='any',inplace=True)


data1_drop.describe()
```

	cons_12m	cons_gas_12m	cons_last_month	forecast_base_bill_ele	forecast_b
count	1.567400e+04	1.567400e+04	1.567400e+04	3368.000000	
mean	1.915538e+05	3.132277e+04	1.929701e+04	335.473230	
std	6.724861e+05	1.716293e+05	8.229678e+04	647.645417	
min	-1.252760e+05	-3.037000e+03	-9.138600e+04	-364.940000	
25%	5.883500e+03	0.000000e+00	0.000000e+00	0.000000	
50%	1.520100e+04	0.000000e+00	8.960000e+02	163.615000	
75%	4.948550e+04	0.000000e+00	4.096000e+03	396.512500	
max	1.609711e+07	4.154590e+06	4.538720e+06	12566.080000	

```
msno.matrix(data1_drop)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcfbf1209d0>
```



Double-click (or enter) to edit

```
# Choose the columns without missing values
incomplete_cols = ['channel_sales','date_first_activ','forecast_base_bill_ele','forecast_base

complete_cols = [column_name for column_name in data1_drop.columns if column_name not in incc

data1_cc = data1_drop[complete_cols]

# Fix negative numeric variables
numeric = [column_name for column_name in data1_cc.columns
           if data1_cc[column_name].dtype == 'float64'
           or data1_cc[column_name].dtype == 'int64']

# Overwrite positive values on negative values
data1_cc[numeric] = data1_cc[numeric].apply(abs)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:3069: SettingWithCopyWarnin
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user
  self[k1] = value[k2]
```

```
data1_cc.describe()
```

	cons_12m	cons_gas_12m	cons_last_month	forecast_cons_12m	forecast_cons_ye
count	1.567400e+04	1.567400e+04	1.567400e+04	15674.000000	15674.0000
mean	1.916143e+05	3.132400e+04	1.941588e+04	2359.676441	1911.6983
std	6.724688e+05	1.716291e+05	8.226881e+04	3979.605687	5224.8135
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.0000
25%	5.893250e+03	0.000000e+00	0.000000e+00	514.045000	0.0000
50%	1.522000e+04	0.000000e+00	9.090000e+02	1178.970000	382.0000
75%	4.953825e+04	0.000000e+00	4.131500e+03	2677.220000	1994.7500
max	1.609711e+07	4.154590e+06	4.538720e+06	103801.930000	175375.0000

```
# Convert the has_gas column to  Yes/No
data1_cc['has_gas'] = data1_cc['has_gas'].replace({'t':'Yes','f':'No'})

# Merge the main dataset with the output dataset
data1_cc_merged = data1_cc.merge(right=data3,on=['id'])

# Convert the churn column to Churned/Stayed
data1_cc_merged['churn'] = data1_cc_merged['churn'].replace({1:'Churned',0:'Stayed'})
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user
```

```
data1_cc_merged.head()
```

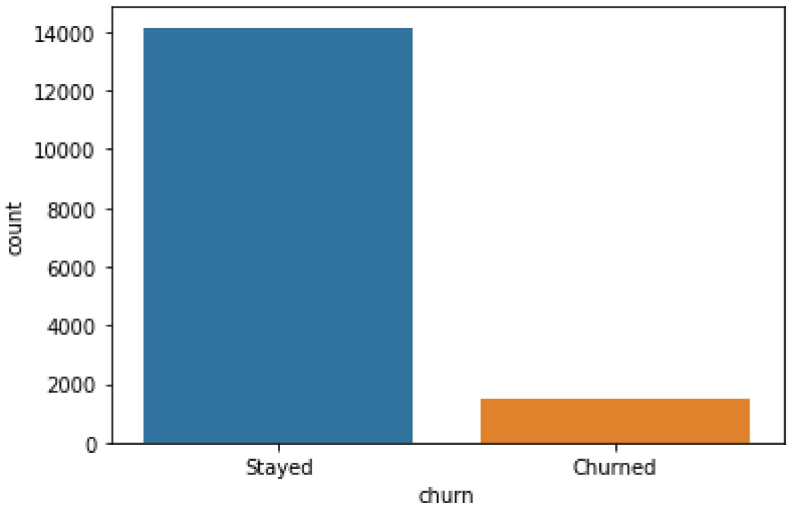
	id	cons_12m	cons_gas_12m	cons_last_month	date_acti
0	48ada52261e7cf58715202705a0451c9	309275	0	10025	2012-11-0
1	d29c2c54acc38ff3c0614d0a653813dd	4660	0	0	2009-08-2
2	764c75f661154dac3a6c254cd082ea7d	544	0	0	2010-04-1
3	bba03439a292a1e166f80264c16191cb	1584	0	0	2010-03-3
4	568bb38a1afd7c0fc49c77b3789b59a3	121335	0	12400	2010-04-0

```
data1_cc_merged.tail()
```

;as	imp_cons	margin_gross_pow_ele	margin_net_pow_ele	nb_prod_act	net_margin	num_yea
⁄es	0.00	27.88	27.88	2	381.77	
No	15.94	0.00	0.00	1	90.34	
No	18.05	39.84	39.84	1	20.38	
No	0.00	13.08	13.08	1	0.96	
No	0.00	11.84	11.84	1	96.34	

Details to check Churn Rate

```
plot = sns.countplot(x='churn', data = data1_cc_merged).set_xticklabels(labels=['Stayed', 'Ch
```




```
# What number of customers have churned in the last 3 months?
attrition_count = data1_cc_merged['churn'].value_counts()
print('Total Number of Churned Customers:\n', attrition_count)
```

```
Total Number of Churned Customers:
 Stayed     14154
Churned     1520
Name: churn, dtype: int64
```

```
#What is the proportion of customer attrition in the last 3 months?
attrition_rate = data1_cc_merged['churn'].value_counts() / data1_cc_merged.shape[0] * 100
print('Attrition rate: \n', attrition_rate)
```

```
Attrition rate:
 Stayed     90.302412
Churned     9.697588
Name: churn, dtype: float64
```

✓ 3s completed at 2:40 AM