Importing modules

In [25]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

Importing data

In [2]:
```
ml_tr=pd.read_csv("ml_case_training_data.csv")
ml_tr_hist=pd.read_csv("ml_case_training_hist_data.csv")
ml_tr_out=pd.read_csv("ml_case_training_output.csv")
```

Examining transaction data

Exploratory data analysis

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

Out[3]:

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

5 rows × 32 columns

In [4]:
```
ml_tr_hist.head()
```

Out[4]:

	id	price_date	price_p1_var	price_p2_var	price_p3_var	price_p1_
0	038af19179925da21a25619c5a24b745	2015-01-01	0.151367	0.0	0.0	44.2669
1	038af19179925da21a25619c5a24b745	2015-02-01	0.151367	0.0	0.0	44.2669
2	038af19179925da21a25619c5a24b745	2015-03-01	0.151367	0.0	0.0	44.2669
3	038af19179925da21a25619c5a24b745	2015-04-01	0.149626	0.0	0.0	44.2669
4	038af19179925da21a25619c5a24b745	2015-05-01	0.149626	0.0	0.0	44.2669

In [5]:
```
ml_tr_out.head()
```

Out[5]:

	id	churn
0	48ada52261e7cf58715202705a0451c9	0
1	24011ae4ebbe3035111d65fa7c15bc57	1
2	d29c2c54acc38ff3c0614d0a653813dd	0
3	764c75f661154dac3a6c254cd082ea7d	0
4	bba03439a292a1e166f80264c16191cb	0

In [6]:
```
ml_tr.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
```

Many inconsistancies. Needs work.

In [7]:
```
ml_tr = ml_tr.fillna(ml_tr.mean())
ml_tr=ml_tr.drop(columns=["campaign_disc_ele"])
```

dropping dulpicates

In [8]:
```
ml_tr=ml_tr.drop_duplicates('id',keep='first')
```

Converting types

In [9]:
```
ml_tr['date_renewal']= pd.to_datetime(ml_tr['date_renewal'])
ml_tr['date_modif_prod']= pd.to_datetime(ml_tr['date_modif_prod'])
ml_tr['date_first_activ']= pd.to_datetime(ml_tr['date_first_activ'])
ml_tr['date_end']= pd.to_datetime(ml_tr['date_end'])
ml_tr['date_activ']= pd.to_datetime(ml_tr['date_activ'])
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16096 entries, 0 to 16095
Data columns (total 31 columns):
 #   Column                    Non-Null Count  Dtype         
---  ------                    --------------  -----         
 0   id                        16096 non-null  object        
 1   activity_new              6551 non-null   object        
 2   channel_sales             11878 non-null  object        
 3   cons_12m                  16096 non-null  int64         
 4   cons_gas_12m              16096 non-null  int64         
 5   cons_last_month           16096 non-null  int64         
 6   date_activ                16096 non-null  datetime64[ns]
 7   date_end                  16094 non-null  datetime64[ns]
 8   date_first_activ          3508 non-null   datetime64[ns]
 9   date_modif_prod           15939 non-null  datetime64[ns]
 10  date_renewal              16056 non-null  datetime64[ns]
 11  forecast_base_bill_ele    16096 non-null  float64       
 12  forecast_base_bill_year   16096 non-null  float64       
 13  forecast_bill_12m         16096 non-null  float64       
 14  forecast_cons             16096 non-null  float64       
 15  forecast_cons_12m         16096 non-null  float64       
 16  forecast_cons_year        16096 non-null  int64         
 17  forecast_discount_energy  16096 non-null  float64       
 18  forecast_meter_rent_12m   16096 non-null  float64       
 19  forecast_price_energy_p1  16096 non-null  float64       
 20  forecast_price_energy_p2  16096 non-null  float64       
 21  forecast_price_pow_p1     16096 non-null  float64       
 22  has_gas                   16096 non-null  object        
 23  imp_cons                  16096 non-null  float64       
 24  margin_gross_pow_ele      16096 non-null  float64       
 25  margin_net_pow_ele        16096 non-null  float64       
 26  nb_prod_act               16096 non-null  int64         
 27  net_margin                16096 non-null  float64       
 28  num_years_antig           16096 non-null  int64         
 29  origin_up                 16009 non-null  object        
 30  pow_max                   16096 non-null  float64       
dtypes: datetime64[ns](5), float64(15), int64(6), object(5)
memory usage: 3.9+ MB
```

In [11]:
```
ml_tr_out.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
```

No data missing.

In [12]:
```
ml_tr_hist.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_p1_var  191643 non-null  float64
 3   price_p2_var  191643 non-null  float64
 4   price_p3_var  191643 non-null  float64
 5   price_p1_fix  191643 non-null  float64
 6   price_p2_fix  191643 non-null  float64
 7   price_p3_fix  191643 non-null  float64
dtypes: float64(6), object(2)
memory usage: 11.8+ MB
```

Data clearning needed.

In [13]:
```
ml_tr_hist=ml_tr_hist.dropna()
ml_tr_hist.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 191643 entries, 0 to 193001
Data columns (total 8 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   id            191643 non-null  object 
 1   price_date    191643 non-null  object 
 2   price_p1_var  191643 non-null  float64
 3   price_p2_var  191643 non-null  float64
 4   price_p3_var  191643 non-null  float64
 5   price_p1_fix  191643 non-null  float64
 6   price_p2_fix  191643 non-null  float64
 7   price_p3_fix  191643 non-null  float64
dtypes: float64(6), object(2)
memory usage: 13.2+ MB
```

Feature Engineering

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

correlation mattrix

In [14]:
```
ml_tr.corr()
```

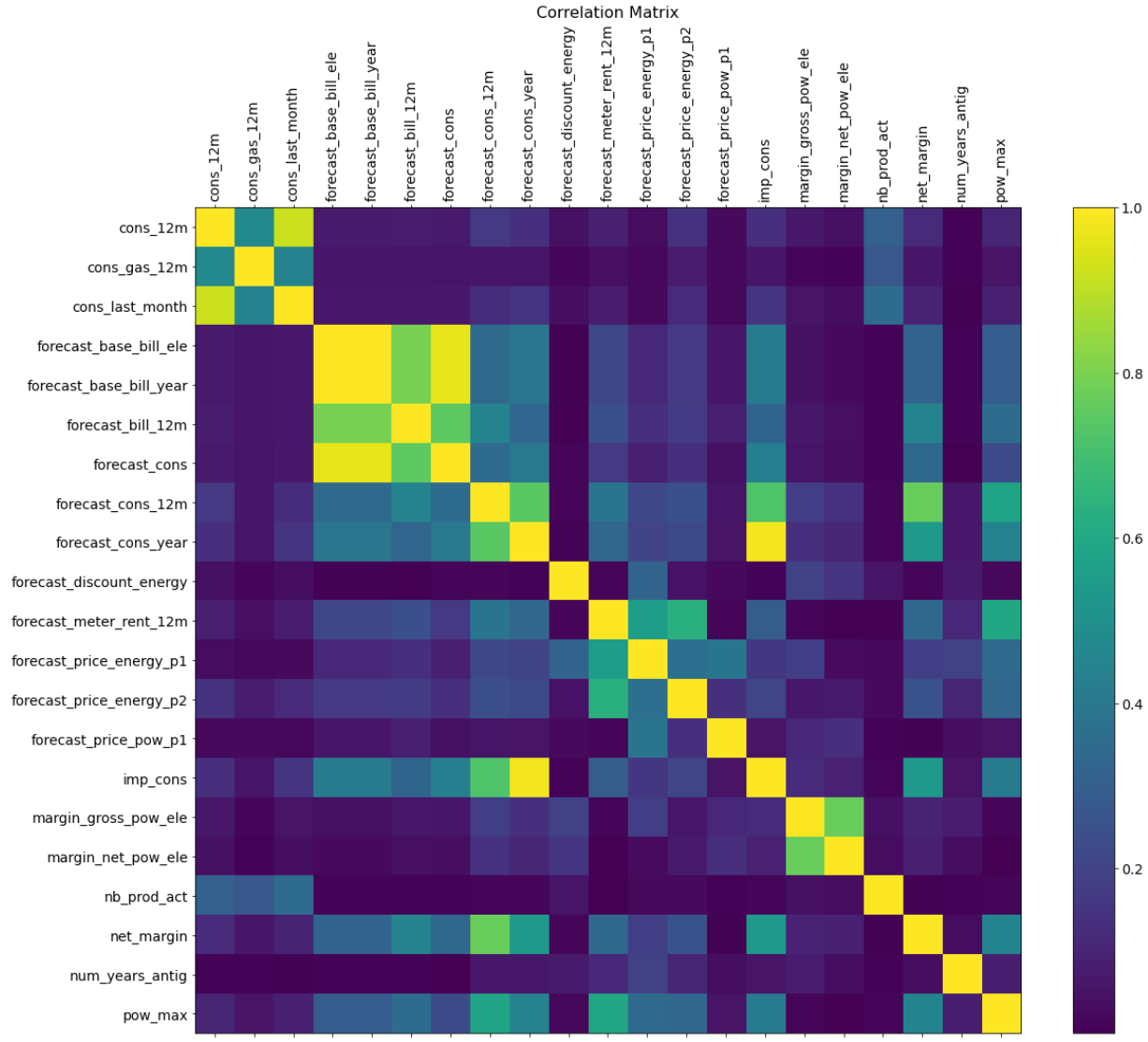
Out[14]:

	cons_12m	cons_gas_12m	cons_last_month	forecast_base_bill_ele	forecast_l
cons_12m	1.000000	0.471233	0.919545	0.071443	
cons_gas_12m	0.471233	1.000000	0.447209	0.061064	
cons_last_month	0.919545	0.447209	1.000000	0.066250	
forecast_base_bill_ele	0.071443	0.061064	0.066250	1.000000	
forecast_base_bill_year	0.071443	0.061064	0.066250	1.000000	

	cons_12m	cons_gas_12m	cons_last_month	forecast_base_bill_ele	forecast_l
forecast_bill_12m	0.080056	0.059547	0.065208	0.794776	
forecast_cons	0.071527	0.054740	0.066546	0.964402	
forecast_cons_12m	0.165168	0.059525	0.129574	0.344620	
forecast_cons_year	0.139526	0.057619	0.151476	0.393361	
forecast_discount_energy	-0.043551	-0.014407	-0.037699	0.005792	
forecast_meter_rent_12m	0.085996	0.040327	0.076066	0.214113	
forecast_price_energy_p1	-0.033425	-0.021608	-0.024195	-0.116035	
forecast_price_energy_p2	0.146229	0.075628	0.122922	0.165854	
forecast_price_pow_p1	-0.025326	-0.026212	-0.020017	0.062149	
imp_cons	0.139353	0.060609	0.153861	0.414904	
margin_gross_pow_ele	-0.065184	-0.016866	-0.054069	-0.044562	
margin_net_pow_ele	-0.045558	-0.008242	-0.037665	-0.027109	
nb_prod_act	0.308567	0.272005	0.350711	0.010411	
net_margin	0.119910	0.058928	0.096343	0.320016	
num_years_antig	0.008810	-0.008626	0.004860	0.008122	
pow_max	0.102422	0.052365	0.089565	0.291136	

21 rows × 21 columns

In [20]:
```
df=ml_tr.copy()
f = plt.figure(figsize=(19, 15))
plt.matshow(df.corr().abs(), fignum=f.number)
plt.xticks(range(df.select_dtypes(['number']).shape[1]), df.select_dtypes(['number']
plt.yticks(range(df.select_dtypes(['number']).shape[1]), df.select_dtypes(['number']
cb = plt.colorbar()
cb.ax.tick_params(labelsize=14)
plt.title('Correlation Matrix', fontsize=16);
```



Selecting imporant features as having corr moire than 0.6

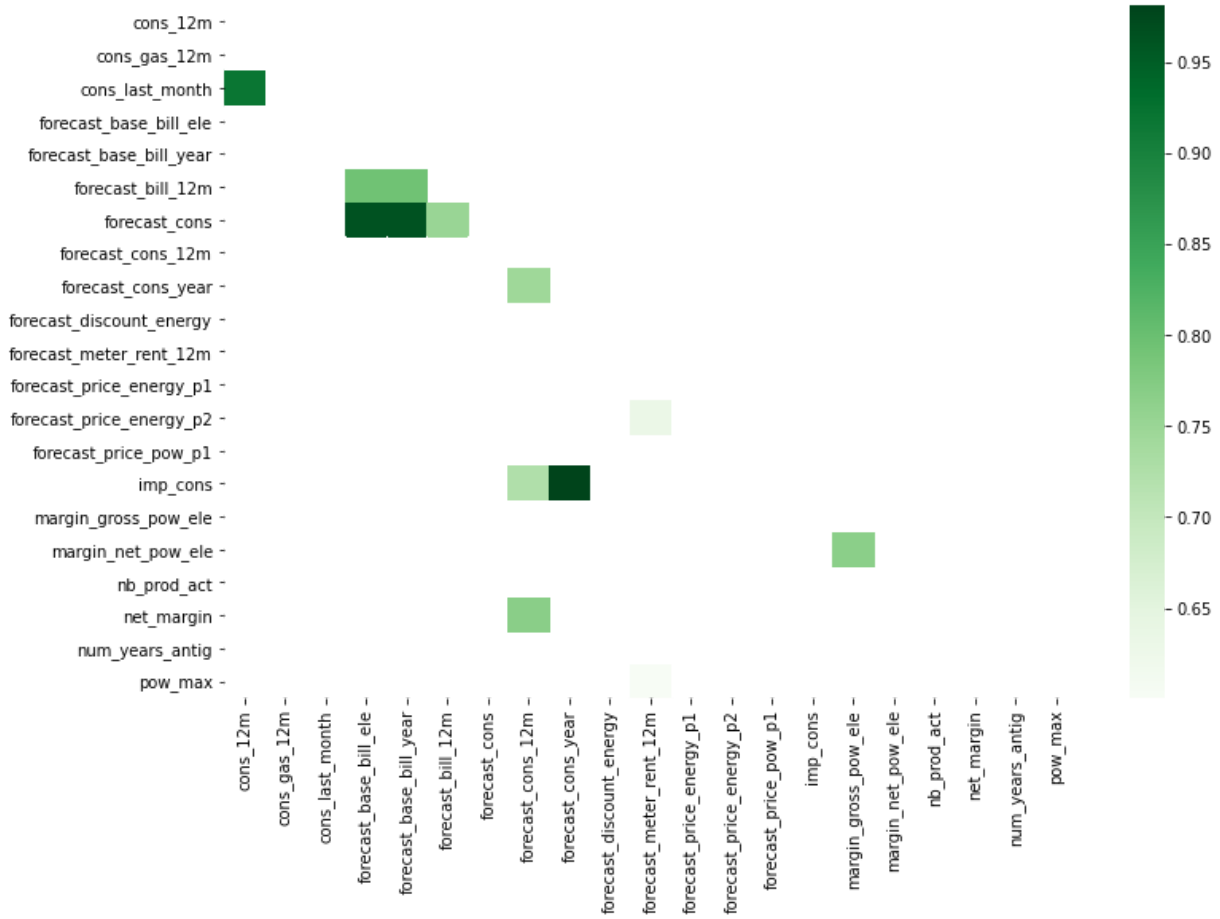
In [41]:
```
corr = df.corr().abs()

corr = corr.where(np.tril(np.ones(corr.shape)).astype(np.bool))

kot = corr[corr>=.6]
kot=kot[kot<1]


plt.figure(figsize=(12,8))
sns.heatmap(kot, cmap="Greens")
```

Out[41]: <AxesSubplot:>



In [43]:
```
data=kot.unstack().sort_values().drop_duplicates()
print(data)
```

```
forecast_meter_rent_12m  pow_max                    0.600566
                         forecast_price_energy_p2   0.632863
forecast_cons_12m        imp_cons                   0.725550
                         forecast_cons_year         0.746076
forecast_bill_12m        forecast_cons              0.751430
margin_gross_pow_ele     margin_net_pow_ele         0.766521
forecast_cons_12m        net_margin                 0.768609
forecast_base_bill_ele   forecast_bill_12m          0.794776
cons_12m                 cons_last_month            0.919545
forecast_base_bill_ele   forecast_cons              0.964402
forecast_cons_year       imp_cons                   0.981732
cons_12m                 cons_12m                        NaN
dtype: float64
```

In [51]:
```
names = []

for n in data.index:
    if n[0] not in names:
        names.append(n[0])
    if n[1] not in names:
        names.append(n[1])

print("List of usable features : ",names)
```

```
List of usable features :  ['forecast_meter_rent_12m', 'pow_max', 'forecast_price_en
ergy_p2', 'forecast_cons_12m', 'imp_cons', 'forecast_cons_year', 'forecast_bill_12
m', 'forecast_cons', 'margin_gross_pow_ele', 'margin_net_pow_ele', 'net_margin', 'fo
recast_base_bill_ele', 'cons_12m', 'cons_last_month']
```

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

In [54]:
```
ml_tr=ml_tr[names]
```

In [55]:
```
ml_tr.head()
```

Out[55]:

	forecast_meter_rent_12m	pow_max	forecast_price_energy_p2	forecast_cons_12m	imp_cons	forec
0	359.29	180.000	0.088347	26520.30	831.8	
1	1.78	43.648	0.098142	0.00	0.0	
2	16.27	13.800	0.000000	189.95	0.0	
3	38.72	13.856	0.087899	47.96	0.0	
4	19.83	13.200	0.000000	240.04	0.0	

In [56]:
```
df=ml_tr.copy()
f = plt.figure(figsize=(19, 15))
plt.matshow(df.corr().abs(), fignum=f.number)
plt.xticks(range(df.select_dtypes(['number']).shape[1]), df.select_dtypes(['number']
plt.yticks(range(df.select_dtypes(['number']).shape[1]), df.select_dtypes(['number']
cb = plt.colorbar()
cb.ax.tick_params(labelsize=14)
plt.title('Correlation Matrix', fontsize=16);
```