Feature Engineering

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import datetime
from datetime import datetime,timedelta

%matplotlib inline
```

In [2]:
```
df = pd.read_csv('client_data.csv')
df.head()
```

Out[2]:

	id	channel_sales	cons_12m	cons_gas_12m	cons_last_month	date_activ
0	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	2013-06-15
1	d29c2c54acc38ff3c0614d0a653813dd	MISSING	4660	0	0	2009-08-21
2	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0	0	2010-04-16
3	bba03439a292a1e166f80264c16191cb	lmkebamcaaclubfxadlmueccxoimlema	1584	0	0	2010-03-30
4	149d57cf92fc41cf94415803a877cb4b	MISSING	4425	0	526	2010-01-13

5 rows × 26 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14606 entries, 0 to 14605
Data columns (total 26 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  churn                           14606 non-null  int64  
dtypes: float64(11), int64(7), object(8)
memory usage: 2.9+ MB
```

In [4]:
```
df.describe()
```

Out[4]:

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

In [5]:
```
df.columns
```

Out[5]:
```
Index(['id', 'channel_sales', 'cons_12m', 'cons_gas_12m', 'cons_last_month',
       'date_activ', 'date_end', 'date_modif_prod', 'date_renewal',
       'forecast_cons_12m', 'forecast_cons_year', 'forecast_discount_energy',
       'forecast_meter_rent_12m', 'forecast_price_energy_off_peak',
       '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_antig', 'origin_up', 'pow_max', 'churn'],
      dtype='object')
```

Drop Unwanted Columns

In [6]:
```
df.drop(['channel_sales', 'cons_12m', 'cons_gas_12m', 'date_modif_prod', 'date_renewal', 'forecast_cons_
         'forecast_discount_energy', 'forecast_meter_rent_12m', 'forecast_price_energy_off_peak', 'forec
         'forecast_price_pow_off_peak', 'imp_cons', 'origin_up'],inplace=True,axis=1)
```

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

Out[7]:

	id	cons_last_month	date_activ	date_end	has_gas	margin_gross_pow_ele	margin_net_pow_e
0	24011ae4ebbe3035111d65fa7c15bc57	0	2013-06-15	2016-06-15	t	25.44	25.
1	d29c2c54acc38ff3c0614d0a653813dd	0	2009-08-21	2016-08-30	f	16.38	16.
2	764c75f661154dac3a6c254cd082ea7d	0	2010-04-16	2016-04-16	f	28.60	28.
3	bba03439a292a1e166f80264c16191cb	0	2010-03-30	2016-03-30	f	30.22	30.
4	149d57cf92fc41cf94415803a877cb4b	526	2010-01-13	2016-03-07	f	44.91	44.

In [8]:
```
df.to_csv('customerchurn.csv',index=False)
```

In [9]:
```
cust_churn = pd.read_csv('customerchurn.csv',parse_dates=['date_activ','date_end'],dayfirst=True)
```

In [10]:
```
cust_churn.head()
```

Out[10]:

	id	cons_last_month	date_activ	date_end	has_gas	margin_gross_pow_ele	margin_net_pow_e
0	24011ae4ebbe3035111d65fa7c15bc57	0	2013-06-15	2016-06-15	t	25.44	25.
1	d29c2c54acc38ff3c0614d0a653813dd	0	2009-08-21	2016-08-30	f	16.38	16.
2	764c75f661154dac3a6c254cd082ea7d	0	2010-04-16	2016-04-16	f	28.60	28.
3	bba03439a292a1e166f80264c16191cb	0	2010-03-30	2016-03-30	f	30.22	30.
4	149d57cf92fc41cf94415803a877cb4b	526	2010-01-13	2016-03-07	f	44.91	44.

In [11]:
```
cust_churn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14606 entries, 0 to 14605
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    14606 non-null  object
 1   cons_last_month       14606 non-null  int64
 2   date_activ            14606 non-null  datetime64[ns]
 3   date_end              14606 non-null  datetime64[ns]
 4   has_gas               14606 non-null  object
 5   margin_gross_pow_ele  14606 non-null  float64
 6   margin_net_pow_ele    14606 non-null  float64
 7   nb_prod_act           14606 non-null  int64
 8   net_margin            14606 non-null  float64
 9   num_years_antig       14606 non-null  int64
 10  pow_max               14606 non-null  float64
 11  churn                 14606 non-null  int64
dtypes: datetime64[ns](2), float64(4), int64(4), object(2)
memory usage: 1.3+ MB
```

In [12]:
```
cust_churn['duration'] = (cust_churn.date_end - cust_churn.date_activ).dt.days
```

In [13]:
```
cust_churn.head()
```

Out[13]:

	id	cons_last_month	date_activ	date_end	has_gas	margin_gross_pow_ele	margin_net_pow_e
0	24011ae4ebbe3035111d65fa7c15bc57	0	2013-06-15	2016-06-15	t	25.44	25.
1	d29c2c54acc38ff3c0614d0a653813dd	0	2009-08-21	2016-08-30	f	16.38	16.
2	764c75f661154dac3a6c254cd082ea7d	0	2010-04-16	2016-04-16	f	28.60	28.
3	bba03439a292a1e166f80264c16191cb	0	2010-03-30	2016-03-30	f	30.22	30.
4	149d57cf92fc41cf94415803a877cb4b	526	2010-01-13	2016-03-07	f	44.91	44.

In [14]:
```
cust_price = pd.read_csv('price_data.csv')
cust_price.head()
```

Out[14]:

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

In [15]:
```
final_df = pd.merge(left=cust_churn, right=cust_price, on='id', how='inner')
```

In [16]:
```
final_df.head()
```

Out[16]:

	id	cons_last_month	date_activ	date_end	has_gas	margin_gross_pow_ele	margin_net_pow_e
0	24011ae4ebbe3035111d65fa7c15bc57	0	2013-06-15	2016-06-15	t	25.44	25.4
1	24011ae4ebbe3035111d65fa7c15bc57	0	2013-06-15	2016-06-15	t	25.44	25.4
2	24011ae4ebbe3035111d65fa7c15bc57	0	2013-06-15	2016-06-15	t	25.44	25.4
3	24011ae4ebbe3035111d65fa7c15bc57	0	2013-06-15	2016-06-15	t	25.44	25.4
4	24011ae4ebbe3035111d65fa7c15bc57	0	2013-06-15	2016-06-15	t	25.44	25.4

In [17]:
```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 175149 entries, 0 to 175148
Data columns (total 20 columns):
 #   Column                Non-Null Count   Dtype         
---  ------                --------------   -----         
 0   id                    175149 non-null  object        
 1   cons_last_month       175149 non-null  int64         
 2   date_activ            175149 non-null  datetime64[ns]
 3   date_end              175149 non-null  datetime64[ns]
 4   has_gas               175149 non-null  object        
 5   margin_gross_pow_ele  175149 non-null  float64       
 6   margin_net_pow_ele    175149 non-null  float64       
 7   nb_prod_act           175149 non-null  int64         
 8   net_margin            175149 non-null  float64       
 9   num_years_antig       175149 non-null  int64         
 10  pow_max               175149 non-null  float64       
 11  churn                 175149 non-null  int64         
 12  duration              175149 non-null  int64         
 13  price_date            175149 non-null  object        
 14  price_off_peak_var    175149 non-null  float64       
 15  price_peak_var        175149 non-null  float64       
 16  price_mid_peak_var    175149 non-null  float64       
 17  price_off_peak_fix    175149 non-null  float64       
 18  price_peak_fix        175149 non-null  float64       
 19  price_mid_peak_fix    175149 non-null  float64       
dtypes: datetime64[ns](2), float64(10), int64(5), object(3)
memory usage: 28.1+ MB
```

In [18]:
```
final_df.isna().sum()
```

Out[18]:
```
id                      0
cons_last_month         0
date_activ              0
date_end                0
has_gas                 0
margin_gross_pow_ele    0
margin_net_pow_ele      0
nb_prod_act             0
net_margin              0
num_years_antig         0
pow_max                 0
churn                   0
duration                0
price_date              0
price_off_peak_var      0
price_peak_var          0
price_mid_peak_var      0
price_off_peak_fix      0
price_peak_fix          0
price_mid_peak_fix      0
dtype: int64
```

In [19]:
```
final_df.drop(['id','date_activ','date_end','price_date'],axis=1,inplace=True)
```

In [20]:
```
final_df.head()
```

Out[20]:

	cons_last_month	has_gas	margin_gross_pow_ele	margin_net_pow_ele	nb_prod_act	net_margin	num_years_antig	pow_max
0	0	t	25.44	25.44	2	678.99	3	43.648
1	0	t	25.44	25.44	2	678.99	3	43.648
2	0	t	25.44	25.44	2	678.99	3	43.648
3	0	t	25.44	25.44	2	678.99	3	43.648
4	0	t	25.44	25.44	2	678.99	3	43.648

In [21]:
```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 175149 entries, 0 to 175148
Data columns (total 16 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   cons_last_month       175149 non-null  int64  
 1   has_gas               175149 non-null  object 
 2   margin_gross_pow_ele  175149 non-null  float64
 3   margin_net_pow_ele    175149 non-null  float64
 4   nb_prod_act           175149 non-null  int64  
 5   net_margin            175149 non-null  float64
 6   num_years_antig       175149 non-null  int64  
 7   pow_max               175149 non-null  float64
 8   churn                 175149 non-null  int64  
 9   duration              175149 non-null  int64  
 10  price_off_peak_var    175149 non-null  float64
 11  price_peak_var        175149 non-null  float64
 12  price_mid_peak_var    175149 non-null  float64
 13  price_off_peak_fix    175149 non-null  float64
 14  price_peak_fix        175149 non-null  float64
 15  price_mid_peak_fix    175149 non-null  float64
dtypes: float64(10), int64(5), object(1)
memory usage: 22.7+ MB
```

#Treat Duplicate Values

In [22]:
```
final_df.duplicated(keep='first').sum()
```

Out[22]: 118784

In [23]:
```
final_df.drop_duplicates(ignore_index=True,inplace=True)
```

In [24]:
```
final_df.duplicated(keep='first').sum()
```

Out[24]: 0

One Hot-Encoding

In [25]:
```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56365 entries, 0 to 56364
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   cons_last_month       56365 non-null  int64  
 1   has_gas               56365 non-null  object 
 2   margin_gross_pow_ele  56365 non-null  float64
 3   margin_net_pow_ele    56365 non-null  float64
 4   nb_prod_act           56365 non-null  int64  
 5   net_margin            56365 non-null  float64
 6   num_years_antig       56365 non-null  int64  
 7   pow_max               56365 non-null  float64
 8   churn                 56365 non-null  int64  
 9   duration              56365 non-null  int64  
 10  price_off_peak_var    56365 non-null  float64
 11  price_peak_var        56365 non-null  float64
 12  price_mid_peak_var    56365 non-null  float64
 13  price_off_peak_fix    56365 non-null  float64
 14  price_peak_fix        56365 non-null  float64
 15  price_mid_peak_fix    56365 non-null  float64
dtypes: float64(10), int64(5), object(1)
memory usage: 6.9+ MB
```

In [26]:
```
final_df['has_gas'] = pd.get_dummies(data=final_df['has_gas'],drop_first=True)
```

In [27]:
```
final_df.head()
```

Out[27]:

	cons_last_month	has_gas	margin_gross_pow_ele	margin_net_pow_ele	nb_prod_act	net_margin	num_years_antig	pow_max
0	0	1	25.44	25.44	2	678.99	3	43.648
1	0	1	25.44	25.44	2	678.99	3	43.648
2	0	1	25.44	25.44	2	678.99	3	43.648
3	0	1	25.44	25.44	2	678.99	3	43.648
4	0	1	25.44	25.44	2	678.99	3	43.648

In [28]:
```
final_df.to_csv('final2.csv',index=False)
```

Modelling and Evaluation

In [29]:
```
df = pd.read_csv('final2.csv')
```

In [30]:
```
df.head()
```

Out[30]:

	cons_last_month	has_gas	margin_gross_pow_ele	margin_net_pow_ele	nb_prod_act	net_margin	num_years_antig	pow_max
0	0	1	25.44	25.44	2	678.99	3	43.648
1	0	1	25.44	25.44	2	678.99	3	43.648
2	0	1	25.44	25.44	2	678.99	3	43.648
3	0	1	25.44	25.44	2	678.99	3	43.648
4	0	1	25.44	25.44	2	678.99	3	43.648

In [31]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56365 entries, 0 to 56364
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   cons_last_month       56365 non-null  int64  
 1   has_gas               56365 non-null  int64  
 2   margin_gross_pow_ele  56365 non-null  float64
 3   margin_net_pow_ele    56365 non-null  float64
 4   nb_prod_act           56365 non-null  int64  
 5   net_margin            56365 non-null  float64
 6   num_years_antig       56365 non-null  int64  
 7   pow_max               56365 non-null  float64
 8   churn                 56365 non-null  int64  
 9   duration              56365 non-null  int64  
 10  price_off_peak_var    56365 non-null  float64
 11  price_peak_var        56365 non-null  float64
 12  price_mid_peak_var    56365 non-null  float64
 13  price_off_peak_fix    56365 non-null  float64
 14  price_peak_fix        56365 non-null  float64
 15  price_mid_peak_fix    56365 non-null  float64
dtypes: float64(10), int64(6)
memory usage: 6.9 MB
```

In [32]:
```
df.describe()
```

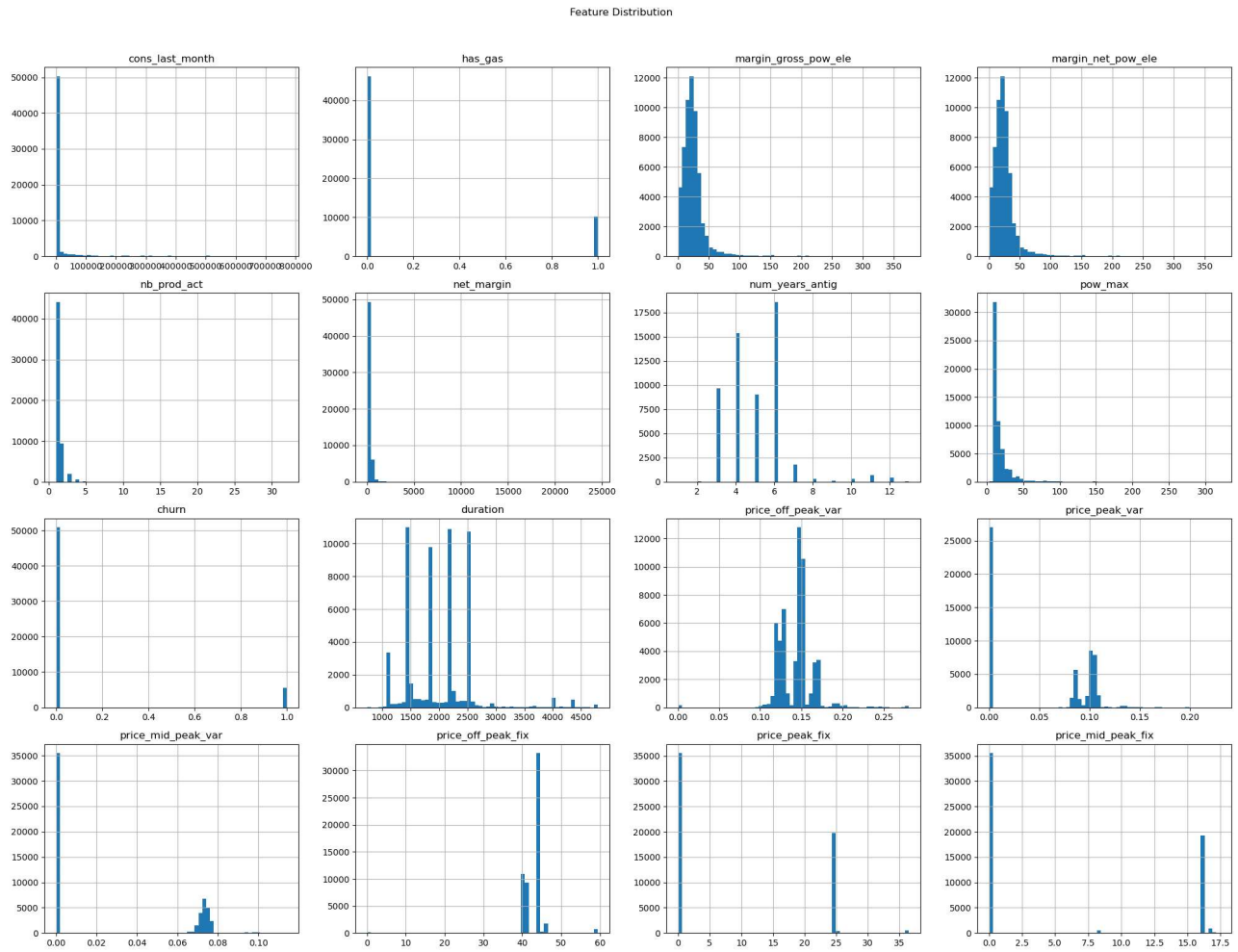
Out[32]:

	cons_last_month	has_gas	margin_gross_pow_ele	margin_net_pow_ele	nb_prod_act	net_margin	num_years_antig
count	56365.000000	56365.000000	56365.000000	56365.000000	56365.000000	56365.000000	56365.000000
mean	14853.179030	0.179970	24.353153	24.349780	1.289825	187.138070	4.960809
std	60874.127265	0.384166	20.078420	20.077219	0.692266	300.027285	1.580625
min	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000
25%	0.000000	0.000000	13.810000	13.800000	1.000000	50.520000	4.000000
50%	751.000000	0.000000	21.520000	21.520000	1.000000	111.580000	5.000000
75%	3219.000000	0.000000	29.760000	29.760000	1.000000	238.450000	6.000000
max	771203.000000	1.000000	374.640000	374.640000	32.000000	24570.650000	13.000000

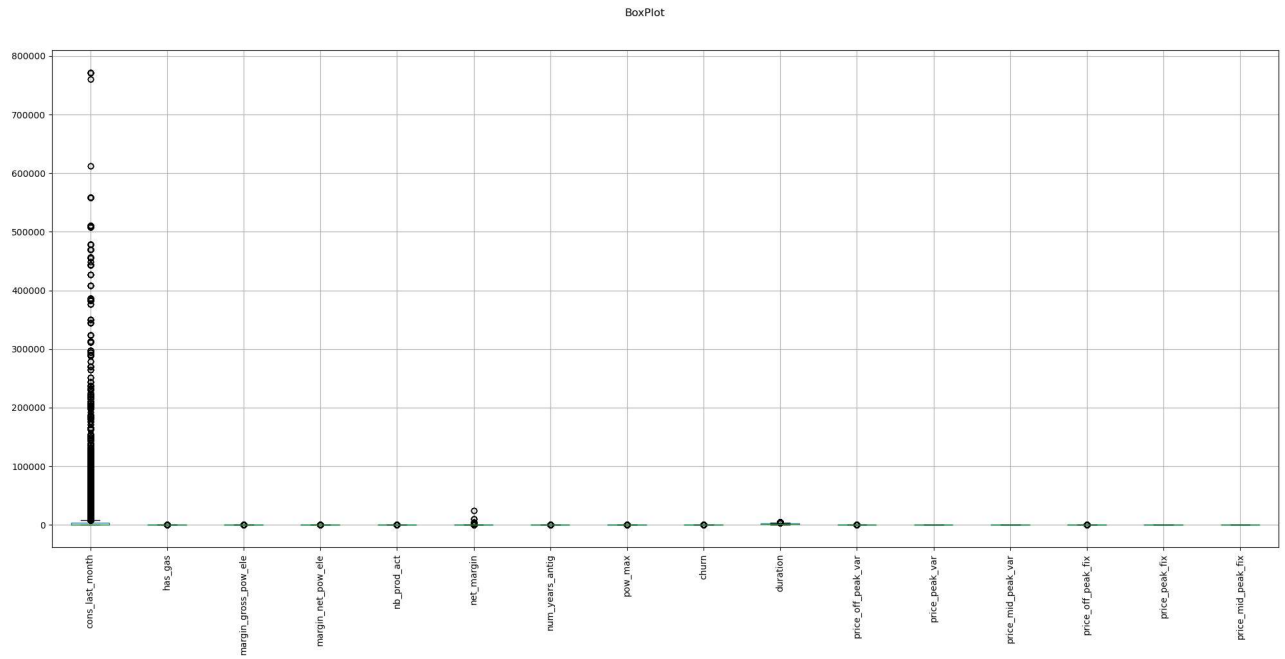
Data Visualization

Univariant Data Analysis

In [33]:
```
df.hist(figsize=(20,15),bins=60)
plt.suptitle('Feature Distribution',x=0.5,y=1.02,ha='center',fontsize='large')
plt.tight_layout()
```



In [34]:
```
df.boxplot(figsize=(20,10))
plt.suptitle('BoxPlot',x=0.5,y=1.02,ha='center',fontsize='large')
plt.xticks(rotation=90)
plt.tight_layout()
```



Correlation

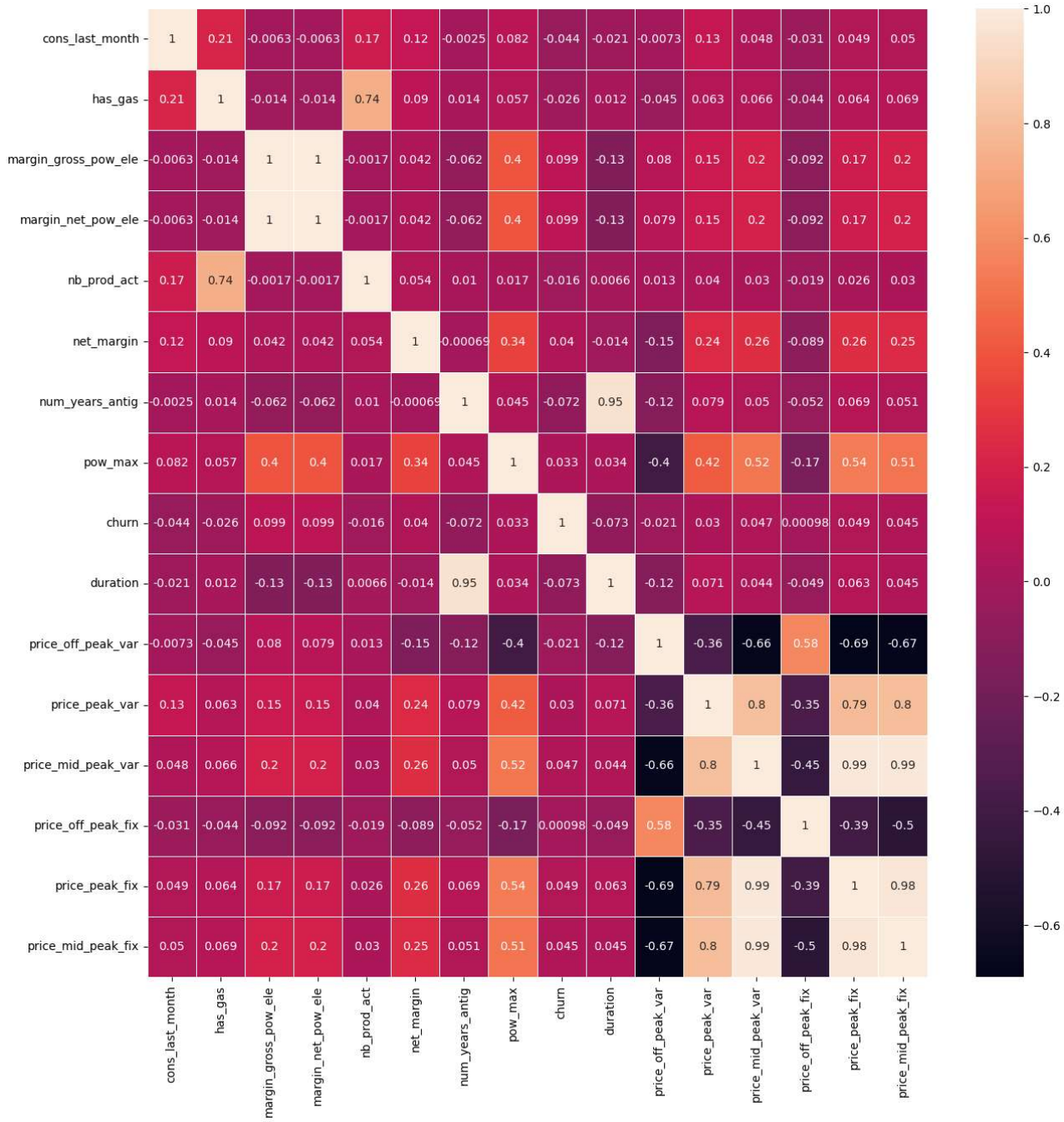
In [35]:
```
df.corr()
```

Out[35]:

	cons_last_month	has_gas	margin_gross_pow_ele	margin_net_pow_ele	nb_prod_act	net_margin	num_y
cons_last_month	1.000000	0.214337	-0.006334	-0.006308	0.166333	0.124351	
has_gas	0.214337	1.000000	-0.013829	-0.013828	0.741109	0.089646	
margin_gross_pow_ele	-0.006334	-0.013829	1.000000	0.999887	-0.001690	0.041990	
margin_net_pow_ele	-0.006308	-0.013828	0.999887	1.000000	-0.001662	0.041752	
nb_prod_act	0.166333	0.741109	-0.001690	-0.001662	1.000000	0.053698	
net_margin	0.124351	0.089646	0.041990	0.041752	0.053698	1.000000	
num_years_antig	-0.002540	0.013515	-0.061706	-0.061852	0.010008	-0.000695	
pow_max	0.082437	0.056716	0.395865	0.395935	0.016563	0.338906	
churn	-0.043791	-0.025860	0.098558	0.098619	-0.015963	0.039519	
duration	-0.020681	0.011907	-0.129721	-0.129831	0.006573	-0.013898	
price_off_peak_var	-0.007295	-0.045188	0.079769	0.079473	0.013351	-0.148348	
price_peak_var	0.130009	0.062632	0.153547	0.153667	0.039732	0.236182	
price_mid_peak_var	0.047510	0.065957	0.200499	0.200577	0.030278	0.255942	
price_off_peak_fix	-0.030821	-0.043896	-0.091542	-0.091623	-0.018877	-0.089100	
price_peak_fix	0.049493	0.063962	0.166615	0.166693	0.025992	0.258401	
price_mid_peak_fix	0.050388	0.068919	0.196688	0.196765	0.030398	0.246584	

In [36]:
```
plt.figure(figsize=(15,15))
sns.heatmap(data=df.corr(),annot=True,linecolor='white',linewidths=0.5)
```

Out[36]: <AxesSubplot:>



Train Test Split

In [37]:
```
df['churn_final'] = df.churn
```

In [38]:
```
df.drop('churn',axis=1,inplace=True)
```

In [39]:
```
df.head()
```

Out[39]:

	cons_last_month	has_gas	margin_gross_pow_ele	margin_net_pow_ele	nb_prod_act	net_margin	num_years_antig	pow_max
0	0	1	25.44	25.44	2	678.99	3	43.648
1	0	1	25.44	25.44	2	678.99	3	43.648
2	0	1	25.44	25.44	2	678.99	3	43.648
3	0	1	25.44	25.44	2	678.99	3	43.648
4	0	1	25.44	25.44	2	678.99	3	43.648

In [40]:
```
df.shape
```

Out[40]:
```
(56365, 16)
```

In [41]:
```
X = df.iloc[:,0:15]
y = df.iloc[:,15]
```

Treat Imbalance Data

In [42]:
```
y.value_counts()
```

Out[42]:
```
0    50833
1     5532
Name: churn_final, dtype: int64
```

In [43]:
```
from imblearn.under_sampling import RandomUnderSampler
```

In [44]:
```
ros = RandomUnderSampler(sampling_strategy='majority',random_state=0)
```

In [45]:
```
new_X,new_y = ros.fit_resample(X,y)
```

In [46]:
```
new_y.value_counts()
```

Out[46]:
```
0    5532
1    5532
Name: churn_final, dtype: int64
```

In [47]:
```
new_X.head()
```

Out[47]:

	cons_last_month	has_gas	margin_gross_pow_ele	margin_net_pow_ele	nb_prod_act	net_margin	num_years_antig	pow_m
39551	15772	0	27.44	27.44	1	73.94	4	17.
54261	0	0	29.34	29.34	1	37.97	6	14.
46551	78	0	34.68	34.68	1	15.29	4	13.
32893	790	0	31.64	31.64	1	80.08	4	13.
42230	0	0	36.24	36.24	1	114.20	4	14.

In [48]:
```
from sklearn.model_selection import train_test_split
```

In [107]:
```
X_train, X_test, y_train, y_test = train_test_split(new_X, new_y, test_size=0.2, random_state=0)
```

In [108]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[108]:
```
((8851, 15), (2213, 15), (8851,), (2213,))
```

Feature Scaling

In [109]:
```
from pycaret.classification import *
```

In [110]:
```
exp = setup(data = df, target = 'churn_final', session_id=0, normalize=True, train_size=0.8, fix_imbalan
```

	Description	Value
0	Session id	0
1	Target	churn_final
2	Target type	Binary
3	Original data shape	(56365, 16)
4	Transformed data shape	(56365, 16)
5	Transformed train set shape	(45092, 16)
6	Transformed test set shape	(11273, 16)
7	Numeric features	15
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Normalize	True
13	Normalize method	zscore
14	Fold Generator	StratifiedKFold
15	Fold Number	10
16	CPU Jobs	-1
17	Use GPU	False
18	Log Experiment	False
19	Experiment Name	clf-default-name
20	USI	ee7e

In [111]:
```
compare_models(exclude=['catboost','lightgbm','lda','qda','mlp','nb','ridge'],fold=5)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
dt	Decision Tree Classifier	0.9617	0.8932	0.8079	0.8029	0.8053	0.7841	0.7841	0.2060
rf	Random Forest Classifier	0.9527	0.9740	0.5289	0.9790	0.6867	0.6635	0.7005	0.6700
et	Extra Trees Classifier	0.9505	0.9579	0.5520	0.9072	0.6859	0.6607	0.6851	0.7180
gbc	Gradient Boosting Classifier	0.9034	0.7156	0.0169	0.9152	0.0333	0.0298	0.1169	0.3300
knn	K Neighbors Classifier	0.9027	0.8248	0.2488	0.5090	0.3341	0.2882	0.3099	2.3900
dummy	Dummy Classifier	0.9018	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2120
ada	Ada Boost Classifier	0.9017	0.6733	0.0018	0.4600	0.0036	0.0026	0.0224	0.3900
svm	SVM - Linear Kernel	0.9016	0.0000	0.0023	0.0645	0.0044	0.0031	0.0089	0.2040
lr	Logistic Regression	0.9007	0.6381	0.0034	0.1861	0.0066	0.0031	0.0124	0.2460

Out[111]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       random_state=0, splitter='best')
```

In [112]:
```
dt = create_model('dt',fold=5)
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9652	0.9068	0.8341	0.8157	0.8248	0.8055	0.8055
1	0.9665	0.9059	0.8305	0.8286	0.8296	0.8110	0.8110
2	0.9600	0.8902	0.8034	0.7918	0.7975	0.7753	0.7753
3	0.9580	0.8856	0.7955	0.7805	0.7879	0.7646	0.7646
4	0.9587	0.8774	0.7763	0.7979	0.7869	0.7641	0.7642
Mean	0.9617	0.8932	0.8079	0.8029	0.8053	0.7841	0.7841
Std	0.0035	0.0115	0.0218	0.0172	0.0183	0.0202	0.0202

In [113]:
```
print(dt)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       random_state=0, splitter='best')
```

In [114]:
```
tuned_dt = tune_model(dt,optimize='F1',fold=5)
```

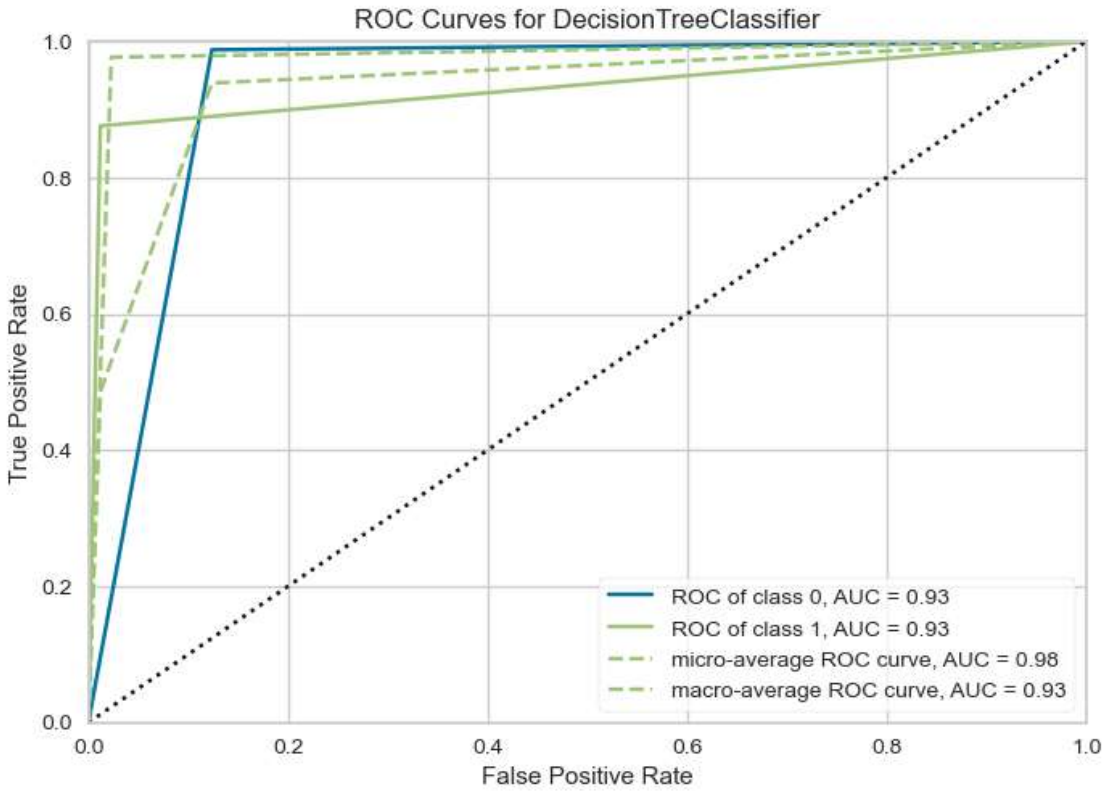
Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9129	0.7580	0.1659	0.7577	0.2722	0.2456	0.3285
1	0.9122	0.7566	0.1503	0.7688	0.2514	0.2266	0.3153
2	0.9120	0.7806	0.1740	0.7097	0.2795	0.2505	0.3228
3	0.9126	0.7481	0.1763	0.7256	0.2836	0.2551	0.3296
4	0.9112	0.7514	0.1345	0.7727	0.2291	0.2060	0.2989
Mean	0.9122	0.7589	0.1602	0.7469	0.2632	0.2367	0.3190
Std	0.0006	0.0114	0.0158	0.0249	0.0203	0.0182	0.0113

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Original model was better than the tuned model, hence it will be returned. NOTE: The display metrics ar
e for the tuned model (not the original one).
```

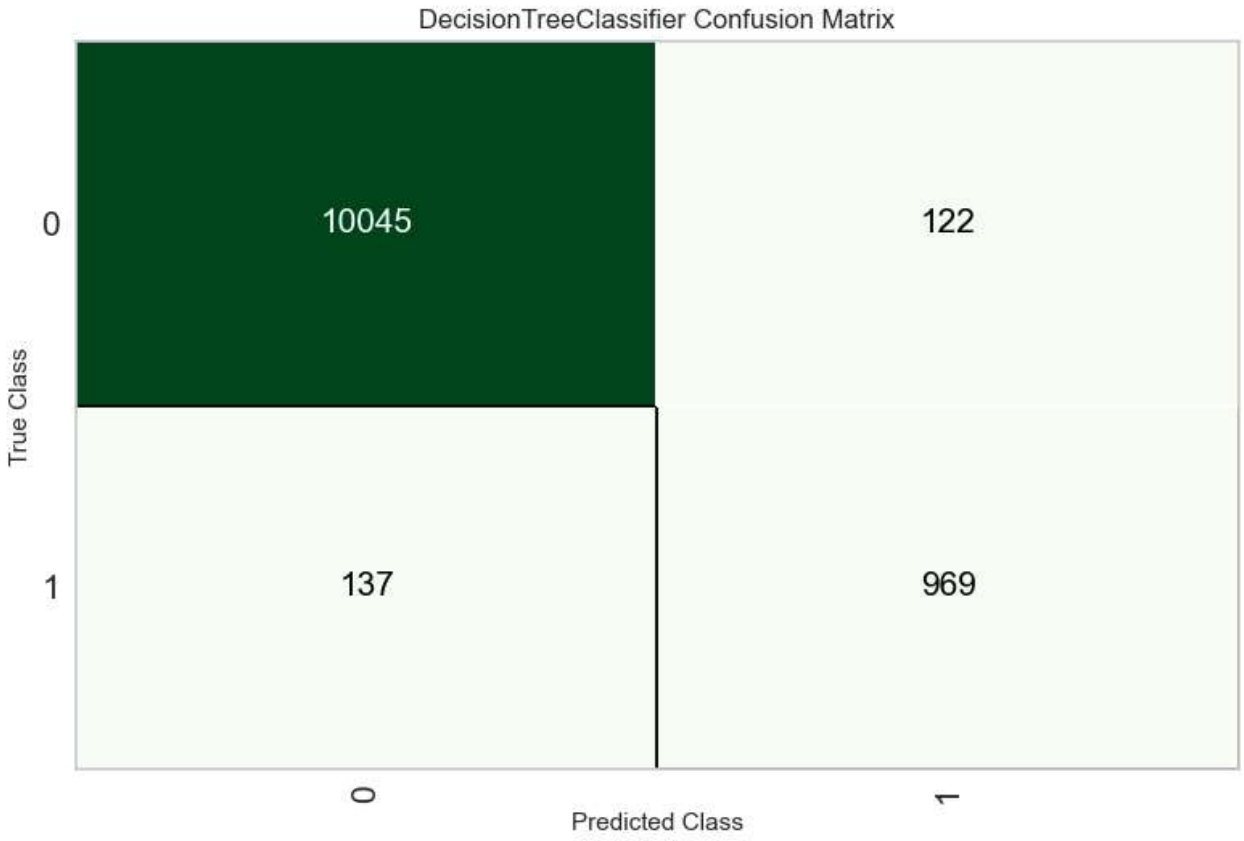
In [115]:
```
print(tuned_dt)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       random_state=0, splitter='best')
```

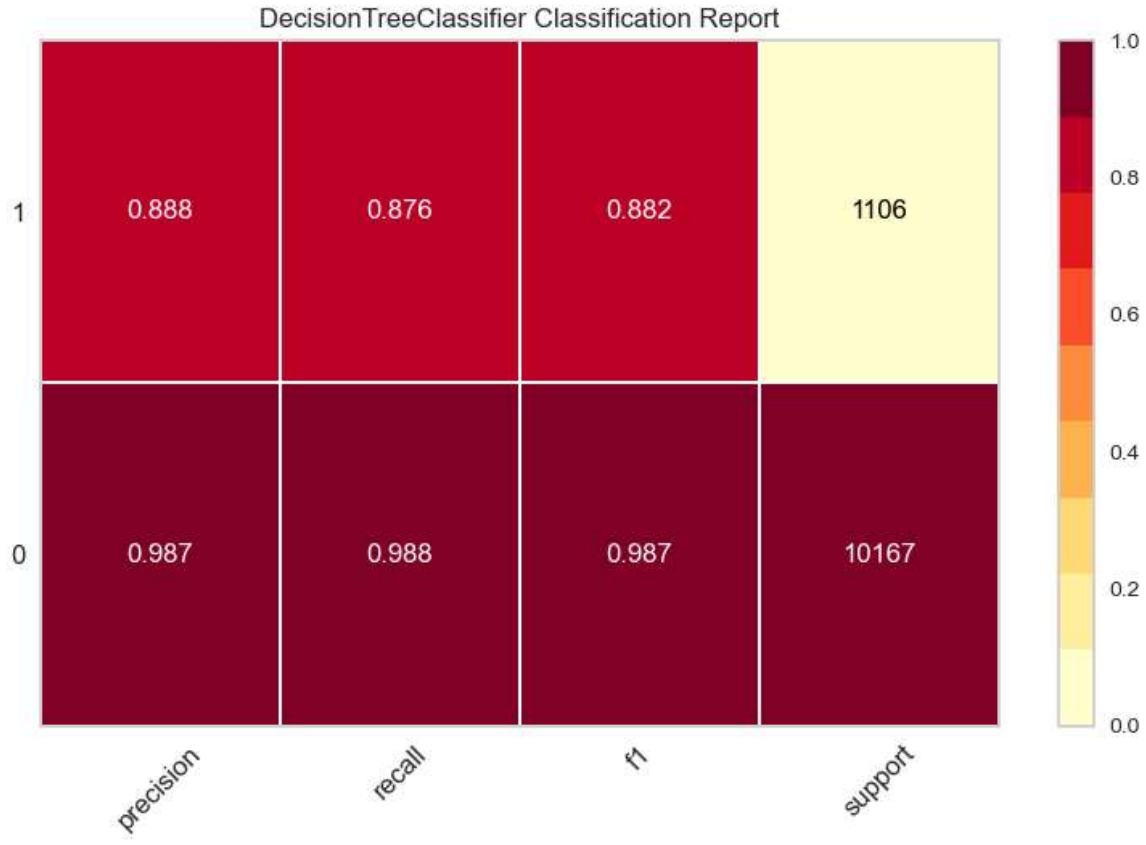
In [116]:
```
plot_model(tuned_dt)
```



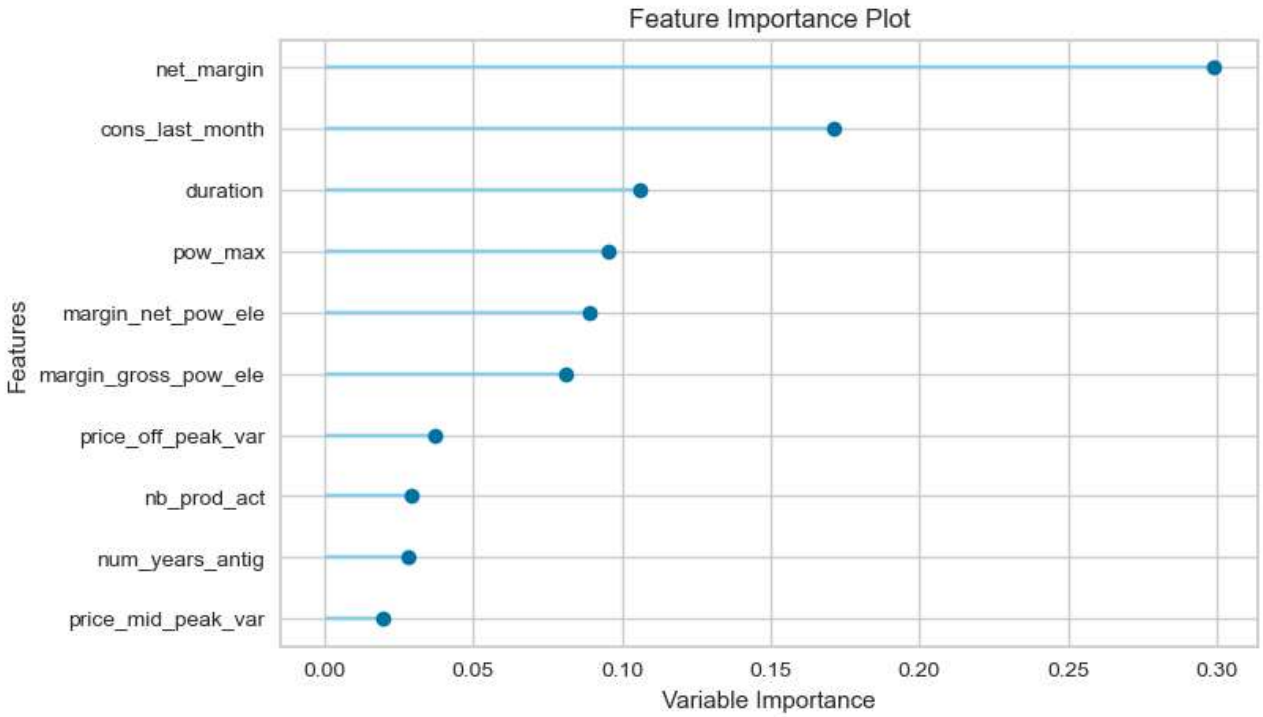
In [117]:
```
plot_model(tuned_dt,plot='confusion_matrix')
```



In [118]:
```
plot_model(tuned_dt,plot='class_report')
```



In [119]:
```
plot_model(tuned_dt,plot='feature')
```



Classification and Regression Model

In [93]:
```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in e:\python\lib\site-packages (1.7.6)
Requirement already satisfied: numpy in e:\python\lib\site-packages (from xgboost) (1.21.6)
Requirement already satisfied: scipy in e:\python\lib\site-packages (from xgboost) (1.10.1)
```

In [94]:
```
import xgboost as xg
```

In [95]:
```
from xgboost import XGBClassifier
from xgboost import XGBRegressor
```

In [96]:
```
from sklearn.model_selection import GridSearchCV,RandomizedSearchCV
```

In [97]:
```
model = XGBClassifier(random_state=0,n_estimator=100, objective='binary:logistic' )
```

In [98]:
```
parameters = {'max_depth': np.arange(3,10,1),
              'eta': np.arange(0.05,0.3,0.05),
              'n_estimators':np.arange(100,1000,100),
              'min_child_weight': np.arange(1,4,1),
              'gamma':np.arange(0,10,2),
              'subsample':np.arange(0.5,0.9,0.1),
              'colsample_bytree':np.arange(0.5,0.9,0.1),
              'reg_alpha':np.arange(0,1,0.1),
              'reg_lambda':np.arange(0,1,0.1)
             }
```

In [99]:
```
randm = RandomizedSearchCV(estimator=model, param_distributions=parameters, cv=5, n_iter=10, n_jobs=-1,
```

In [100]:
```
randm.fit(new_X,new_y)
```

```
[18:14:59] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0fdc6d574b9c0d1
68-1\xgboost\xgboost-ci-windows\src\learner.cc:767:
Parameters: { "n_estimator" } are not used.
```

Out[100]:

RandomizedSearchCV
estimator: XGBClassifier
XGBClassifier

In [101]:
```
randm.best_estimator_
```

Out[101]:

XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.6, early_stopping_rounds=None,
              enable_categorical=False, eta=0.05, eval_metric=None,
              feature_types=None, gamma=8, gpu_id=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=None, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=5,
              max_leaves=None, min_child_weight=1, missing=nan,
              monotone_constraints=None, n_estimator=100, n_estimators=300,
```

In [102]:
```
randm.best_score_
```

Out[102]:
```
0.5904042826955269
```

In [103]:
```
randm.best_params_
```

Out[103]:
```
{'subsample': 0.5,
 'reg_lambda': 0.6000000000000001,
 'reg_alpha': 0.7000000000000001,
 'n_estimators': 300,
 'min_child_weight': 1,
 'max_depth': 5,
 'gamma': 8,
 'eta': 0.05,
 'colsample_bytree': 0.6}
```

Final Model

In [104]:
```
xgbtmodel = XGBClassifier(subsample=0.5,reg_lambda=0.6,reg_alpha=0.7,n_estimatores=300,min_child_weight=
                          colsample_bytree=0.6,random_state=0,objective='binary:logistic')
```

In [120]:
```
xgbtmodel.fit(X_train,y_train,eval_set=[(X_test,y_test)],eval_metric='logloss',early_stopping_rounds=10)
```

```
[95]    validation_0-logloss:0.62160
[96]    validation_0-logloss:0.62124
[97]    validation_0-logloss:0.62126
[98]    validation_0-logloss:0.62127
[99]    validation_0-logloss:0.62127
```

Out[120]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.6, early_stopping_rounds=None,
              enable_categorical=False, eta=0.05, eval_metric=None,
              feature_types=None, gamma=8, gpu_id=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=None, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=5,
              max_leaves=None, min_child_weight=1, missing=nan,
              monotone_constraints=None, n_estimatores=300, n_estimators=100,
              n_jobs=None, num_parallel_tree=None, ...)
```

In [121]:
```
y_pred = xgbtmodel.predict(X_test)
```

In [122]:
```
y_pred
```

Out[122]:
```
array([0, 0, 1, ..., 1, 1, 1])
```

Model Evaluation

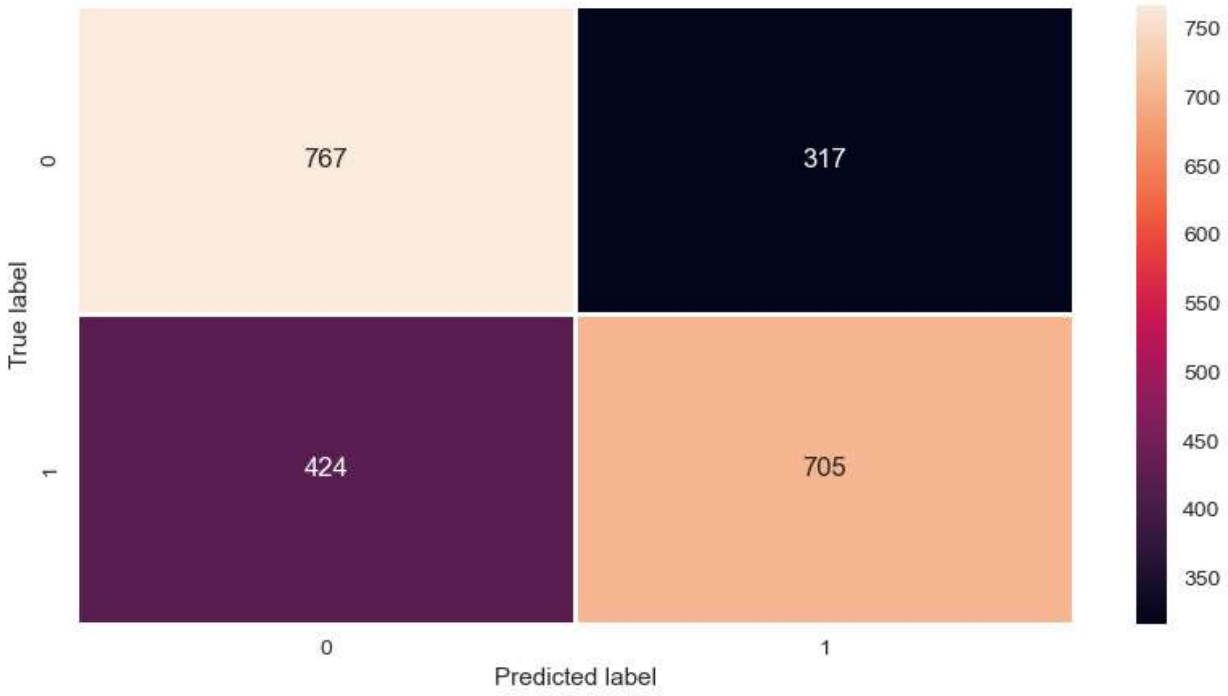
In [136]:
```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_curve,roc_auc_scor
```

In [124]:
```
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[124]:
```
array([[767, 317],
       [424, 705]], dtype=int64)
```

In [125]:
```
fig , ax = plt.subplots(figsize=(10,5))
sns.heatmap(cm, annot=True,fmt='.4g',linewidths=2)
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

Out[125]:
```
Text(0.5, 27.722222222222214, 'Predicted label')
```



In [126]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.64      0.71      0.67      1084
           1       0.69      0.62      0.66      1129

    accuracy                           0.67      2213
   macro avg       0.67      0.67      0.66      2213
weighted avg       0.67      0.67      0.66      2213
```

In [128]:
```
print(accuracy_score(y_test,y_pred)*100)
```

```
66.51604157252599
```

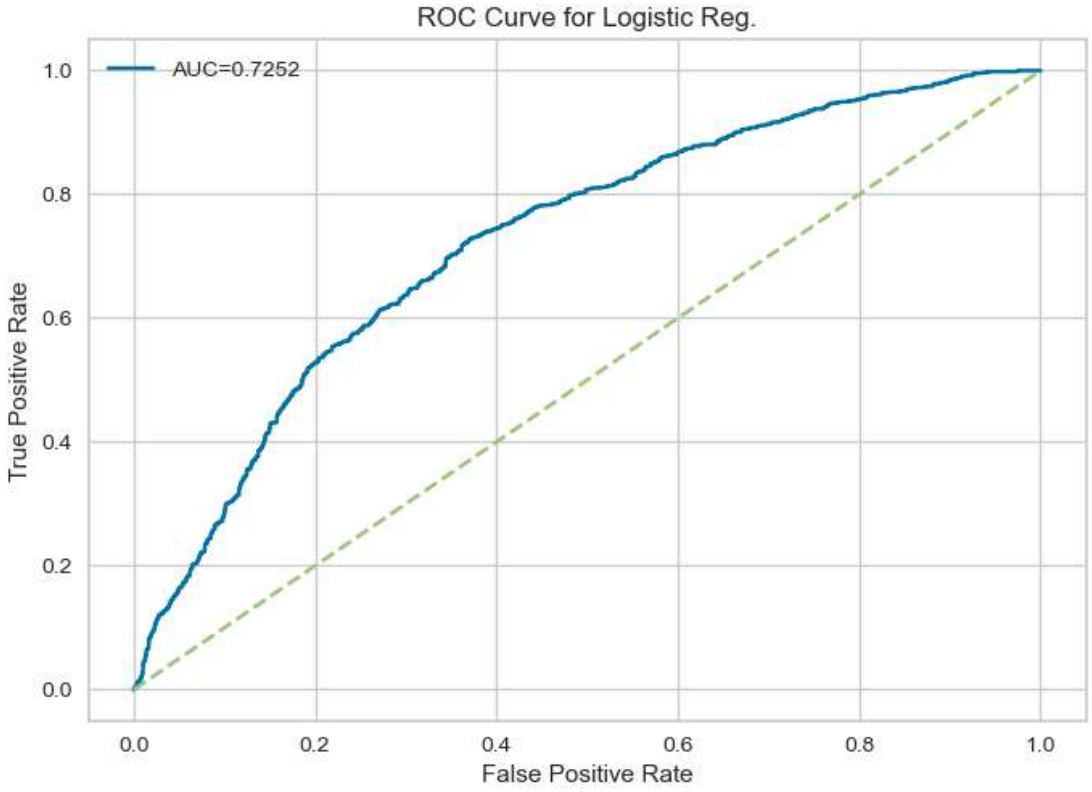
In [137]:
```
xgb_pred_prob = xgbtmodel.predict_proba(X_test)[:][:,1]

xgb_actual_predict = pd.concat([pd.DataFrame(np.array(y_test),columns=['y actual']),
                                pd.DataFrame(xgb_pred_prob,columns=['y pred prob'])],axis=1)
xgb_actual_predict.index = y_test.index

fpr, tpr, tr = roc_curve(xgb_actual_predict['y actual'],xgb_actual_predict['y pred prob'])
auc = roc_auc_score(xgb_actual_predict['y actual'],xgb_actual_predict['y pred prob'])

plt.plot(fpr,tpr, label='AUC=%.4f'%auc)
plt.plot(fpr,fpr,linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Logistic Reg.')
plt.legend()
```

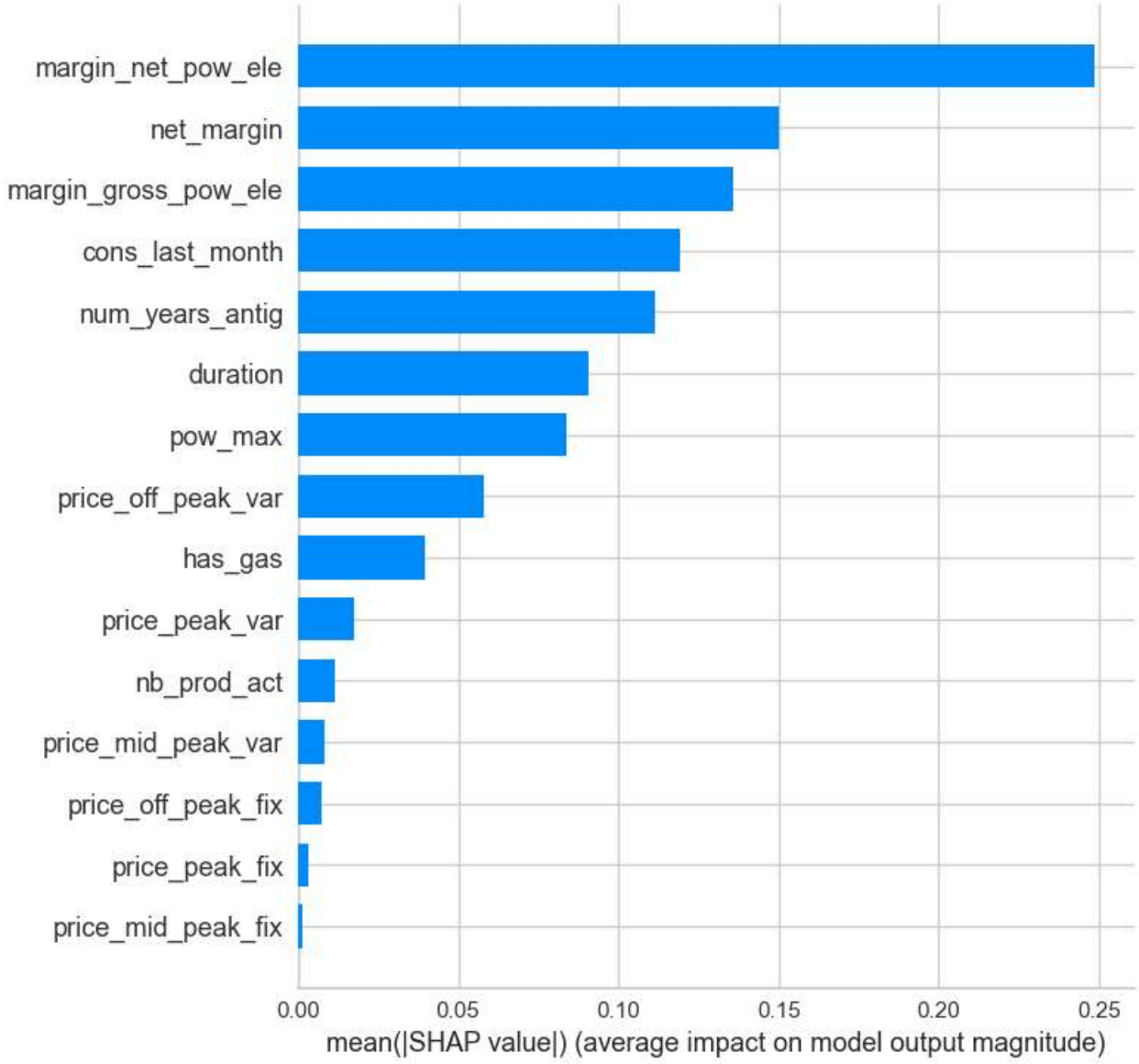
Out[137]:
```
<matplotlib.legend.Legend at 0x257cf9519d0>
```



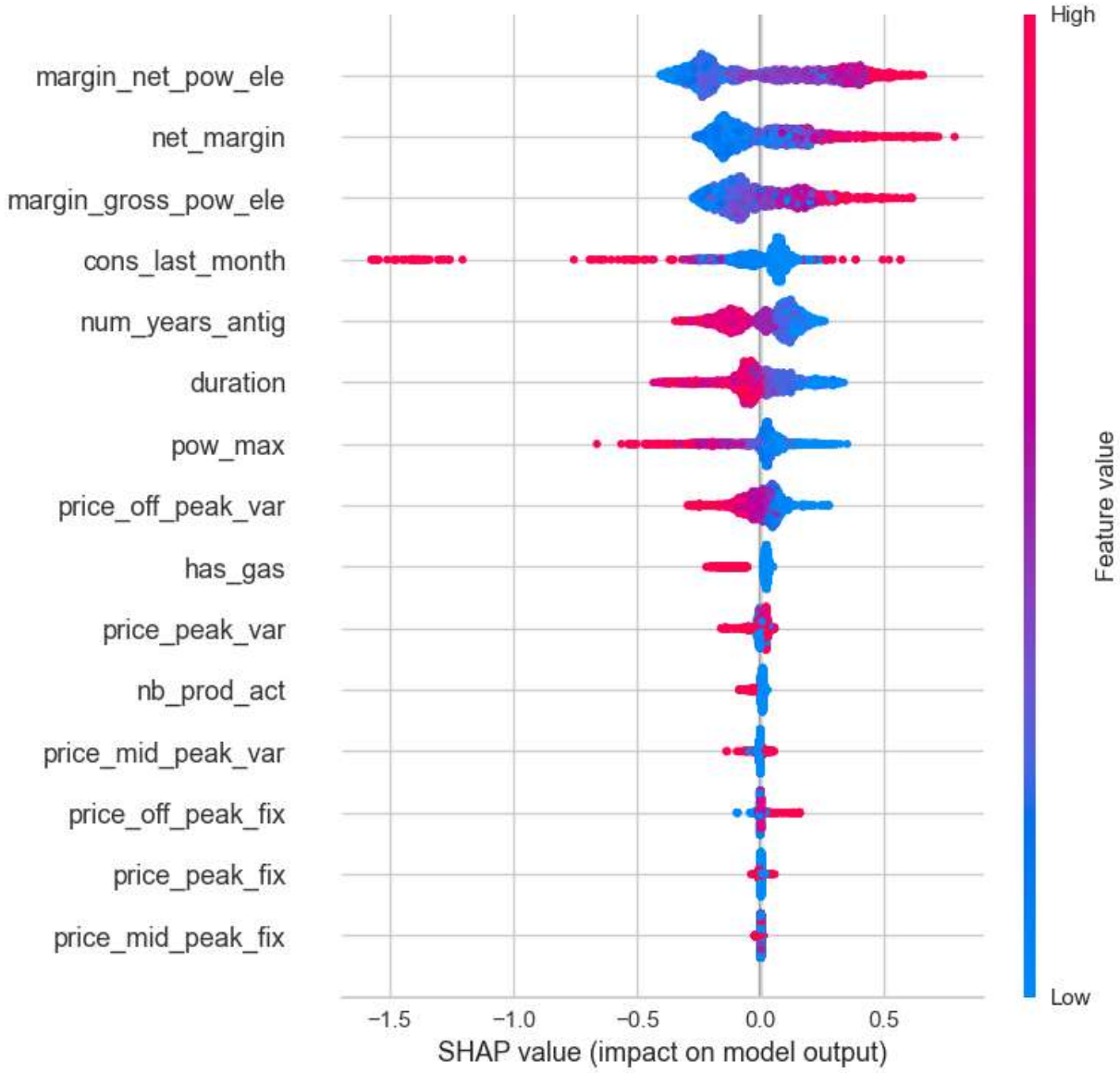
Compute Importance from SHAP Values

In [146]:
```
import shap
```

In [148]:
```
explainer = shap.TreeExplainer(xgbtmodel)
shap_values = explainer.shap_values(X_test)
shap.summary_plot(shap_values, X_test, plot_type="bar")
```



In [149]:
```
shap.summary_plot(shap_values, X_test)
```



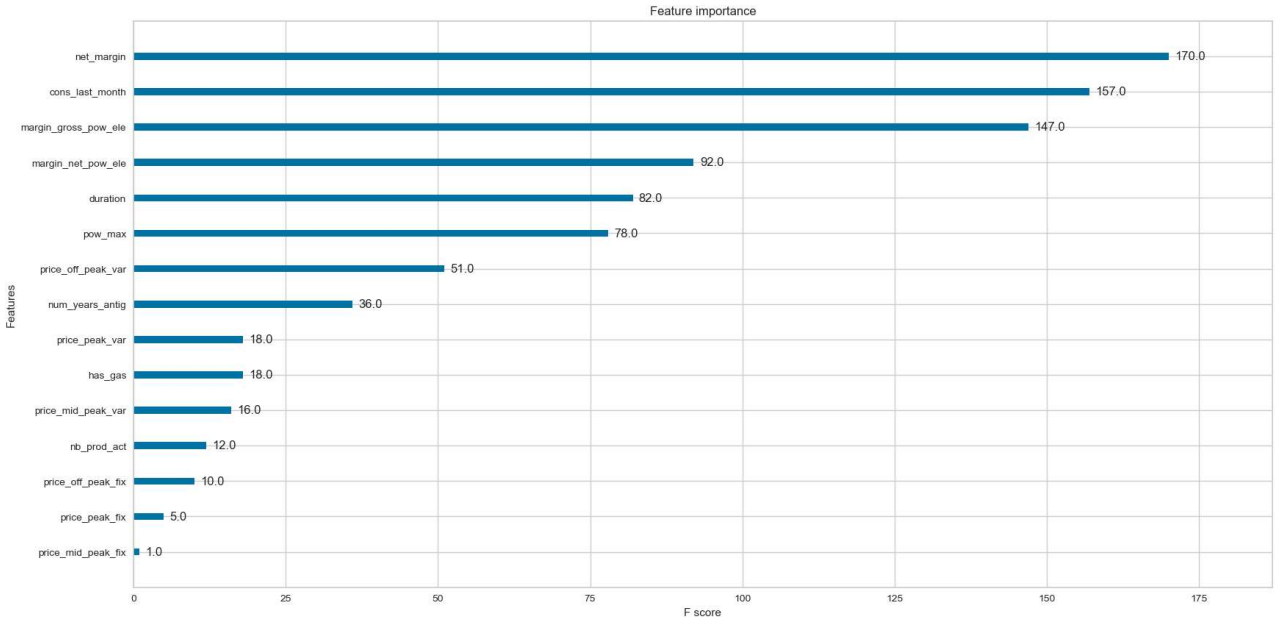
In [152]:
```
new_X.columns
```

Out[152]:
```
Index(['cons_last_month', 'has_gas', 'margin_gross_pow_ele',
       'margin_net_pow_ele', 'nb_prod_act', 'net_margin', 'num_years_antig',
       'pow_max', 'duration', 'price_off_peak_var', 'price_peak_var',
       'price_mid_peak_var', 'price_off_peak_fix', 'price_peak_fix',
       'price_mid_peak_fix'],
      dtype='object')
```

In [154]:
```
xgbtmodel.get_booster().feature_names = ['cons_last_month', 'has_gas', 'margin_gross_pow_ele', 'margin_n
                                         'nb_prod_act','net_margin', 'num_years_antig','pow_max', 'duratio
                                         'price_off_peak_var', 'price_peak_var','price_mid_peak_var',
                                         'price_off_peak_fix', 'price_peak_fix','price_mid_peak_fix']
```

In [156]:
```
fig, ax = plt.subplots(figsize=(20,10))
xg.plot_importance(xgbtmodel.get_booster(),ax=ax)
```

Out[156]: <AxesSubplot:title={'center':'Feature importance'}, xlabel='F score', ylabel='Features'>



Cross-Validation

In [158]:
```
from sklearn.model_selection import cross_val_score
```

In [159]:
```
cv = cross_val_score(xgbtmodel,new_X,new_y,cv=5,verbose=1,scoring='f1')
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

[18:42:37] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0fdc6d574b9c0d1
68-1\xgboost\xgboost-ci-windows\src\learner.cc:767:
Parameters: { "n_estimatores" } are not used.

[18:42:38] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0fdc6d574b9c0d1
68-1\xgboost\xgboost-ci-windows\src\learner.cc:767:
Parameters: { "n_estimatores" } are not used.

[18:42:39] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0fdc6d574b9c0d1
68-1\xgboost\xgboost-ci-windows\src\learner.cc:767:
Parameters: { "n_estimatores" } are not used.

[18:42:40] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0fdc6d574b9c0d1
68-1\xgboost\xgboost-ci-windows\src\learner.cc:767:
Parameters: { "n_estimatores" } are not used.

[18:42:41] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0fdc6d574b9c0d1
68-1\xgboost\xgboost-ci-windows\src\learner.cc:767:
Parameters: { "n_estimatores" } are not used.


[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    4.6s finished
```

In [160]:
```
cv.mean()
```

Out[160]: 0.5908709411640868

In [ ]: