BCG_Model

June 15, 2021

[5]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import datetime
```

1 Main Direction

- Sub-task 1: Think through what key drivers of churn could be for our client
- Sub-task 2: Build the features in order to get ready to model

From previous EDA,

Firstly, I found that the churn percentage is about 10%. And we saw that churn is not specifically realted to any SME category in particular. As for sales channel, some channel has 0% churn rate. However it may due to the fact that there are only few customers through certain channel.
Secondly, I reckoned the consumption-related factors(cons_12m, cons_fas_12m ….) may contribute to this churn percentage. We can observe a highly right-skewed distribution in most of the consumption variables. From the box-plots we can discover there plenty outliers, which further indicate the skewness of consumptions.
The "forecast of consumption" can contribute substantially as well, however the large missing value(>75%) makes some of these data unable to provide useful information, if we fill up with estimate value, it adds confusion to the model
Further, for the attributes such as "num_years_antig", we can cleary see that less loyal customers(low years of antig) is more likely to churn. And customers with many active products and services are less likely to churn
Lastly, I observed that at certain period of time, the churn rate is significntly higher, which may results from some economic cycle or some impact on the market. So I believe this may contribute useful informaiton to our model

Potential key drivers:
1. Consumptions.
2. num_years_antig, nb_prod_act …etc
3. Dates

2 Feature Engineering

[6]:
```
train_data = pd.read_csv('ml_case_training_data.csv')
churn_data = pd.read_csv('ml_case_training_output.csv')
history_data = pd.read_csv('ml_case_training_hist_data.csv')
```

[7]:
```
train_data.head()
```

[7]:

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

	cons_gas_12m	cons_last_month	date_activ	date_end	date_first_activ
0	0	10025	2012-11-07	2016-11-06	NaN
1	54946	0	2013-06-15	2016-06-15	NaN
2	0	0	2009-08-21	2016-08-30	NaN
3	0	0	2010-04-16	2016-04-16	NaN
4	0	0	2010-03-30	2016-03-30	NaN

	…	forecast_price_pow_p1	has_gas	imp_cons	margin_gross_pow_ele
0	…	58.995952	f	831.8	-41.76
1	…	40.606701	t	0.0	25.44
2	…	44.311378	f	0.0	16.38
3	…	44.311378	f	0.0	28.60
4	…	44.311378	f	0.0	30.22

	margin_net_pow_ele	nb_prod_act	net_margin	num_years_antig
0	-41.76	1	1732.36	3
1	25.44	2	678.99	3
2	16.38	1	18.89	6
3	28.60	1	6.60	6
4	30.22	1	25.46	6

	origin_up	pow_max
0	ldkssxwpmemidmecebumciepifcamkci	180.000
1	lxidpiddsbxsbosboudacockeimpuepw	43.648
2	kamkkxfxxuwbdslkwifmmcsiusiuosws	13.800

```
3  kamkkxfxxuwbdslkwifmmcsiusiuosws   13.856
4  kamkkxfxxuwbdslkwifmmcsiusiuosws   13.200

[5 rows x 32 columns]
```

[8]:
```
train_data.describe()
```

[8]:

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

	forecast_base_bill_ele	forecast_base_bill_year	forecast_bill_12m
count	3508.000000	3508.000000	3508.000000
mean	335.843857	335.843857	3837.441866
std	649.406000	649.406000	5425.744327
min	-364.940000	-364.940000	-2503.480000
25%	0.000000	0.000000	1158.175000
50%	162.955000	162.955000	2187.230000
75%	396.185000	396.185000	4246.555000
max	12566.080000	12566.080000	81122.630000

	forecast_cons	forecast_cons_12m	forecast_cons_year	…
count	3508.000000	16096.000000	16096.000000	…
mean	206.845165	2370.555949	1907.347229	…
std	455.634288	4035.085664	5257.364759	…
min	0.000000	-16689.260000	-85627.000000	…
25%	0.000000	513.230000	0.000000	…
50%	42.215000	1179.160000	378.000000	…
75%	228.117500	2692.077500	1994.250000	…
max	9682.890000	103801.930000	175375.000000	…

	forecast_price_energy_p1	forecast_price_energy_p2
count	15970.000000	15970.000000
mean	0.135901	0.052951
std	0.026252	0.048617
min	0.000000	0.000000
25%	0.115237	0.000000
50%	0.142881	0.086163
75%	0.146348	0.098837
max	0.273963	0.195975

```
       forecast_price_pow_p1      imp_cons  margin_gross_pow_ele  \
count           15970.000000  16096.000000          16083.000000
mean               43.533496    196.123447             22.462276
std                 5.212252    494.366979             23.700883
min                -0.122184  -9038.210000           -525.540000
25%                40.606701      0.000000             11.960000
50%                44.311378     44.465000             21.090000
75%                44.311378    218.090000             29.640000
max                59.444710  15042.790000            374.640000

       margin_net_pow_ele   nb_prod_act    net_margin  num_years_antig  \
count        16083.000000  16096.000000  16081.000000     16096.000000
mean            21.460318      1.347788    217.987028         5.030629
std             27.917349      1.459808    366.742030         1.676101
min           -615.660000      1.000000  -4148.990000         1.000000
25%             11.950000      1.000000     51.970000         4.000000
50%             20.970000      1.000000    119.680000         5.000000
75%             29.640000      1.000000    275.810000         6.000000
max            374.640000     32.000000  24570.650000        16.000000

            pow_max
count  16093.000000
mean      20.604131
std       21.772421
min        1.000000
25%       12.500000
50%       13.856000
75%       19.800000
max      500.000000

[8 rows x 22 columns]
```

[9]:
```
train = train_data.merge(churn_data, on = 'id')
```

2.1 Finding null and discard useless columns

[10]:
```
null_percentage = (train_data.isnull().sum() / train_data.isnull().count()).
↪sort_values(ascending = False)
missing_data = pd.DataFrame({'Total Null': train_data.isnull().sum(),
                             'Null Percentage': null_percentage
                                }).sort_values(by='Null Percentage', ascending␣
↪= False)
```

[11]:
```
missing_data
```

[11]:

	Total Null	Null Percentage
campaign_disc_ele	16096	1.000000
date_first_activ	12588	0.782058
forecast_cons	12588	0.782058
forecast_bill_12m	12588	0.782058
forecast_base_bill_year	12588	0.782058
forecast_base_bill_ele	12588	0.782058
activity_new	9545	0.593004
channel_sales	4218	0.262053
date_modif_prod	157	0.009754
forecast_price_energy_p1	126	0.007828
forecast_price_pow_p1	126	0.007828
forecast_price_energy_p2	126	0.007828
forecast_discount_energy	126	0.007828
origin_up	87	0.005405
date_renewal	40	0.002485
net_margin	15	0.000932
margin_gross_pow_ele	13	0.000808
margin_net_pow_ele	13	0.000808
pow_max	3	0.000186
date_end	2	0.000124
forecast_meter_rent_12m	0	0.000000
forecast_cons_12m	0	0.000000
has_gas	0	0.000000
id	0	0.000000
imp_cons	0	0.000000
date_activ	0	0.000000
cons_last_month	0	0.000000
nb_prod_act	0	0.000000
cons_gas_12m	0	0.000000
num_years_antig	0	0.000000
cons_12m	0	0.000000
forecast_cons_year	0	0.000000

[12]:
```
discard_col = missing_data[missing_data['Null Percentage'] > 0.5].index
```

[13]:
```
discard_col
```

[13]:
```
Index(['campaign_disc_ele', 'date_first_activ', 'forecast_cons',
       'forecast_bill_12m', 'forecast_base_bill_year',
       'forecast_base_bill_ele', 'activity_new'],
      dtype='object')
```

[14]:
```
train = train.drop(columns = discard_col, axis = 0)
```

[15]:
```
train.shape
```

[15]:
```
(16096, 26)
```

[16]:
```
num_col = train._get_numeric_data().columns.tolist()
cat_col = set(train.columns) - set(num_col)
```

[17]:
```
num_col
```

[17]:
```
['cons_12m',
 'cons_gas_12m',
 'cons_last_month',
 'forecast_cons_12m',
 'forecast_cons_year',
 'forecast_discount_energy',
 'forecast_meter_rent_12m',
 'forecast_price_energy_p1',
 'forecast_price_energy_p2',
 'forecast_price_pow_p1',
 'imp_cons',
 'margin_gross_pow_ele',
 'margin_net_pow_ele',
 'nb_prod_act',
 'net_margin',
 'num_years_antig',
 'pow_max',
 'churn']
```

[18]:
```
cat_col
```

[18]:
```
{'channel_sales',
 'date_activ',
 'date_end',
 'date_modif_prod',
 'date_renewal',
 'has_gas',
 'id',
 'origin_up'}
```

[19]:
```
train.isnull().sum()
```

[19]:
```
id                          0
channel_sales            4218
cons_12m                    0
cons_gas_12m                0
cons_last_month             0
date_activ                  0
date_end                    2
date_modif_prod           157
```

```
date_renewal                  40
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
churn                          0
dtype: int64
```

2.2 Deal with consumption

[20]:
```
consumption_col = ['cons_12m','cons_gas_12m','cons_last_month']
consumption = train[consumption_col]
```

[21]:
```
consumption.describe()
```

[21]:

	cons_12m	cons_gas_12m	cons_last_month
count	1.609600e+04	1.609600e+04	1.609600e+04
mean	1.948044e+05	3.191164e+04	1.946154e+04
std	6.795151e+05	1.775885e+05	8.235676e+04
min	-1.252760e+05	-3.037000e+03	-9.138600e+04
25%	5.906250e+03	0.000000e+00	0.000000e+00
50%	1.533250e+04	0.000000e+00	9.010000e+02
75%	5.022150e+04	0.000000e+00	4.127000e+03
max	1.609711e+07	4.188440e+06	4.538720e+06

Firstly, the minimum of consumption is negatvie, which seems not plausible, may conclude this is due to corrupted data.

[22]:
```
for col in consumption_col:
    consumption.loc[consumption[col] <0, col] = 0
    # Check whther negative value left
    # print(consumption[consumption[col] < 0])
```

```
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1637:
SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:692:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  iloc._setitem_with_indexer(indexer, value, self.name)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1637:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:692:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  iloc._setitem_with_indexer(indexer, value, self.name)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1637:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:692:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  iloc._setitem_with_indexer(indexer, value, self.name)
```

[23]:
```
consumption.describe()
```

[23]:

	cons_12m	cons_gas_12m	cons_last_month
count	1.609600e+04	1.609600e+04	1.609600e+04
mean	1.948339e+05	3.191227e+04	1.953172e+04
std	6.795045e+05	1.775883e+05	8.231606e+04

```
min    0.000000e+00  0.000000e+00     0.000000e+00
25%    5.906250e+03  0.000000e+00     0.000000e+00
50%    1.533250e+04  0.000000e+00     9.010000e+02
75%    5.022150e+04  0.000000e+00     4.127000e+03
max    1.609711e+07  4.188440e+06     4.538720e+06
```

[24]:
```
def capping_outlier(df,col):
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - (1.5 * iqr)
    upper_bound = q3 + (1.5 * iqr)
    df.loc[df[col] < lower_bound, col ] = lower_bound
    df.loc[df[col] > upper_bound, col ] = upper_bound
#     detect_outlier(df,col)
    return df[col]


def detect_outlier(df, col):
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - (1.5 * iqr)
    upper_bound = q3 + (1.5 * iqr)
    l_outlier = df[col].apply(lambda x: x < lower_bound).sum()
    u_outlier = df[col].apply(lambda x: x > upper_bound).sum()
    total = df[col].count()

    print("Q1:{} Q3:{}".format(q1,q3))
    print("lower outlier :{}, Upper outliers: {}".format(l_outlier/
 ↪total,u_outlier/total))
```

[25]:
```
for col in consumption_col:
    print(col)
    detect_outlier(consumption, col)
```

```
cons_12m
Q1:5906.25 Q3:50221.5
lower outlier :0.0, Upper outliers: 0.15780318091451292
cons_gas_12m
Q1:0.0 Q3:0.0
lower outlier :0.0, Upper outliers: 0.18141153081510936
cons_last_month
Q1:0.0 Q3:4127.0
lower outlier :0.0, Upper outliers: 0.15339214711729623
```

2.2.1 Depend on the model we choose, such model may be non-sensitive to outliers.

Further, majority of 'cons_gas_12m' is zero, almost all of its elemnts are outliers, if we capping these data may not able to truly reflect. **Not capping cons_gas_12m**

[26]:
```
capping_col = set(consumption_col)-set(['cons_gas_12m'])

for col in capping_col:
    print(col)
    consumption[col] = capping_outlier(consumption, col)
```

```
cons_12m
cons_last_month
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
<ipython-input-26-4b62715a02ed>:5: 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_guide/indexing.html#returning-a-view-versus-a-copy
  consumption[col] = capping_outlier(consumption, col)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
<ipython-input-26-4b62715a02ed>:5: 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_guide/indexing.html#returning-a-view-versus-a-copy
  consumption[col] = capping_outlier(consumption, col)
```

[27]:
```
consumption.describe()
```

[27]:

	cons_12m	cons_gas_12m	cons_last_month
count	16096.000000	1.609600e+04	16096.000000
mean	35514.824273	3.191227e+04	2802.216793
std	40977.732430	1.775883e+05	3741.789990
min	0.000000	0.000000e+00	0.000000
25%	5906.250000	0.000000e+00	0.000000
50%	15332.500000	0.000000e+00	901.000000
75%	50221.500000	0.000000e+00	4127.000000
max	116694.375000	4.188440e+06	10317.500000

[28]:
```
for col in consumption_col:
    print(col)
    detect_outlier(consumption, col)
```

```
cons_12m
Q1:5906.25 Q3:50221.5
lower outlier :0.0, Upper outliers: 0.0
cons_gas_12m
Q1:0.0 Q3:0.0
lower outlier :0.0, Upper outliers: 0.18141153081510936
cons_last_month
Q1:0.0 Q3:4127.0
lower outlier :0.0, Upper outliers: 0.0
```

[29]:
```
consumption.isnull().sum()
```

[29]:
```
cons_12m          0
cons_gas_12m      0
cons_last_month   0
dtype: int64
```

2.3 Deal with forecast

[30]:
```
forecast_col = ['forecast_cons_12m','forecast_cons_year',␣
↪'forecast_discount_energy','forecast_meter_rent_12m','forecast_price_energy_p1','forecast_p
forecast = train[forecast_col]
```

[31]:
```
forecast.describe()
```

[31]:

	forecast_cons_12m	forecast_cons_year	forecast_discount_energy	\
count	16096.000000	16096.000000	15970.000000	
mean	2370.555949	1907.347229	0.991547	
std	4035.085664	5257.364759	5.160969	
min	-16689.260000	-85627.000000	0.000000	
25%	513.230000	0.000000	0.000000	
50%	1179.160000	378.000000	0.000000	
75%	2692.077500	1994.250000	0.000000	
max	103801.930000	175375.000000	50.000000	

	forecast_meter_rent_12m	forecast_price_energy_p1	\
count	16096.000000	15970.000000	
mean	70.309945	0.135901	
std	79.023251	0.026252	
min	-242.960000	0.000000	
25%	16.230000	0.115237	
50%	19.440000	0.142881	
75%	131.470000	0.146348	
max	2411.690000	0.273963	

	forecast_price_energy_p2	forecast_price_pow_p1
count	15970.000000	15970.000000
mean	0.052951	43.533496
std	0.048617	5.212252
min	0.000000	-0.122184
25%	0.000000	40.606701
50%	0.086163	44.311378
75%	0.098837	44.311378
max	0.195975	59.444710

[32]:
```
for col in forecast_col:
    forecast.loc[forecast[col]< 0, col] = 0
#     Check whther negative value left
#     print(forecast[forecast[col] < 0])
```

```
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
```

[33]:
```
forecast.describe()
```

[33]:
```
       forecast_cons_12m  forecast_cons_year  forecast_discount_energy  \
count       16096.000000        16096.000000              15970.000000
mean         2375.199540         1918.633387                  0.991547
std          4027.190618         5204.270838                  5.160969
min             0.000000            0.000000                  0.000000
25%           513.230000            0.000000                  0.000000
50%          1179.160000          378.000000                  0.000000
75%          2692.077500         1994.250000                  0.000000
max        103801.930000       175375.000000                 50.000000

       forecast_meter_rent_12m  forecast_price_energy_p1  \
count             16096.000000              15970.000000
mean                 70.333054                  0.135901
std                  78.974201                  0.026252
min                   0.000000                  0.000000
25%                  16.230000                  0.115237
50%                  19.440000                  0.142881
75%                 131.470000                  0.146348
max                2411.690000                  0.273963

       forecast_price_energy_p2  forecast_price_pow_p1
count              15970.000000           15970.000000
mean                   0.052951              43.533503
std                    0.048617               5.212188
min                    0.000000               0.000000
25%                    0.000000              40.606701
50%                    0.086163              44.311378
75%                    0.098837              44.311378
max                    0.195975              59.444710
```

[34]:
```
forecast.isnull().sum()
```

[34]:
```
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
dtype: int64
```

[35]:
```python
for col in forecast_col:
    # fill null with median
    forecast[col].fillna(forecast[col].median(), inplace = True)
```

```
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/series.py:4460:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().fillna(
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/series.py:4460:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().fillna(
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/series.py:4460:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().fillna(
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/series.py:4460:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().fillna(
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/series.py:4460:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().fillna(
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/series.py:4460:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().fillna(
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/series.py:4460:
```

```
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().fillna(
```

[36]:
```
for col in forecast_col:
    print(col)
    detect_outlier(forecast, col)
```

```
forecast_cons_12m
Q1:513.23 Q3:2692.0775000000003
lower outlier :0.0, Upper outliers: 0.08505218687872763
forecast_cons_year
Q1:0.0 Q3:1994.25
lower outlier :0.0, Upper outliers: 0.09903081510934393
forecast_discount_energy
Q1:0.0 Q3:0.0
lower outlier :0.0, Upper outliers: 0.03597166998011928
forecast_meter_rent_12m
Q1:16.23 Q3:131.47
lower outlier :0.0, Upper outliers: 0.02379473161033797
forecast_price_energy_p1
Q1:0.1152369999999999 Q3:0.146348
lower outlier :0.00621272365805169, Upper outliers: 0.0228006958250497
forecast_price_energy_p2
Q1:0.0 Q3:0.098837
lower outlier :0.0, Upper outliers: 0.0
forecast_price_pow_p1
Q1:40.606701 Q3:44.31137796
lower outlier :0.006336978131212724, Upper outliers: 0.04653330019880716
```

[37]:
```
capping_col = set(forecast_col)-set(['forecast_discount_energy'])
print(capping_col)
for col in capping_col:
    print(col)
    forecast[col] = capping_outlier(forecast, col)
```

```
{'forecast_price_energy_p2', 'forecast_cons_year', 'forecast_price_pow_p1',
'forecast_price_energy_p1', 'forecast_meter_rent_12m', 'forecast_cons_12m'}
forecast_price_energy_p2
forecast_cons_year
forecast_price_pow_p1
forecast_price_energy_p1
forecast_meter_rent_12m
forecast_cons_12m
```

```
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
<ipython-input-37-792e83dc9994>:5: 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_guide/indexing.html#returning-a-view-versus-a-copy
  forecast[col] = capping_outlier(forecast, col)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
<ipython-input-37-792e83dc9994>:5: 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_guide/indexing.html#returning-a-view-versus-a-copy
  forecast[col] = capping_outlier(forecast, col)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
<ipython-input-37-792e83dc9994>:5: 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_guide/indexing.html#returning-a-view-versus-a-copy
  forecast[col] = capping_outlier(forecast, col)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
<ipython-input-37-792e83dc9994>:5: 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_guide/indexing.html#returning-a-view-versus-a-copy
  forecast[col] = capping_outlier(forecast, col)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
<ipython-input-37-792e83dc9994>:5: 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_guide/indexing.html#returning-a-view-versus-a-copy
  forecast[col] = capping_outlier(forecast, col)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
/home/brian/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1720:
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_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
<ipython-input-37-792e83dc9994>:5: 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_guide/indexing.html#returning-a-view-versus-a-copy
  forecast[col] = capping_outlier(forecast, col)
```

[38]:
```
for col in forecast_col:
    print(col)
    detect_outlier(forecast, col)
```

```
forecast_cons_12m
```

```
Q1:513.23 Q3:2692.0775000000003
lower outlier :0.0, Upper outliers: 0.0
forecast_cons_year
Q1:0.0 Q3:1994.25
lower outlier :0.0, Upper outliers: 0.0
forecast_discount_energy
Q1:0.0 Q3:0.0
lower outlier :0.0, Upper outliers: 0.03597166998011928
forecast_meter_rent_12m
Q1:16.23 Q3:131.47
lower outlier :0.0, Upper outliers: 0.0
forecast_price_energy_p1
Q1:0.1152369999999999 Q3:0.146348
lower outlier :0.0, Upper outliers: 0.0
forecast_price_energy_p2
Q1:0.0 Q3:0.098837
lower outlier :0.0, Upper outliers: 0.0
forecast_price_pow_p1
Q1:40.606701 Q3:44.31137796
lower outlier :0.0, Upper outliers: 0.0
```

[39]:
```
forecast.isnull().sum()
```

[39]:
```
forecast_cons_12m           0
forecast_cons_year          0
forecast_discount_energy    0
forecast_meter_rent_12m     0
forecast_price_energy_p1    0
forecast_price_energy_p2    0
forecast_price_pow_p1       0
dtype: int64
```

2.4 Deal with categorical data

[40]:
```
for col in cat_col:
    print(col)
    # Fill the categorical column with Mode
    train[col]
    mode = train[col].mode()[0]
    train[col].fillna(mode, inplace=True)
```

```
date_renewal
has_gas
date_modif_prod
date_activ
id
date_end
```

```
origin_up
channel_sales
```

[41]:
```
train.isnull().sum()
```

[41]:
```
id                              0
channel_sales                   0
cons_12m                        0
cons_gas_12m                    0
cons_last_month                 0
date_activ                      0
date_end                        0
date_modif_prod                 0
date_renewal                    0
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                       0
pow_max                         3
churn                           0
dtype: int64
```

[42]:
```
train[consumption_col] = consumption[consumption_col]
train[forecast_col] = forecast[forecast_col]
```

[43]:
```
train.isnull().sum()
```

[43]:
```
id                            0
channel_sales                 0
cons_12m                      0
cons_gas_12m                  0
cons_last_month               0
date_activ                    0
date_end                      0
date_modif_prod               0
date_renewal                  0
```

```
forecast_cons_12m            0
forecast_cons_year           0
forecast_discount_energy     0
forecast_meter_rent_12m      0
forecast_price_energy_p1     0
forecast_price_energy_p2     0
forecast_price_pow_p1        0
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                    0
pow_max                      3
churn                        0
dtype: int64
```

[44]:
```
for col in num_col:
    train[col].fillna(train[col].median(), inplace = True)
```

[45]:
```
train.isnull().sum()
```

[45]:
```
id                          0
channel_sales               0
cons_12m                    0
cons_gas_12m                0
cons_last_month             0
date_activ                  0
date_end                    0
date_modif_prod             0
date_renewal                0
forecast_cons_12m           0
forecast_cons_year          0
forecast_discount_energy    0
forecast_meter_rent_12m     0
forecast_price_energy_p1    0
forecast_price_energy_p2    0
forecast_price_pow_p1       0
has_gas                     0
imp_cons                    0
margin_gross_pow_ele        0
margin_net_pow_ele          0
nb_prod_act                 0
net_margin                  0
num_years_antig             0
```

```
origin_up                       0
pow_max                         0
churn                           0
dtype: int64
```

3 By far, we've already dealt with ourlier and fill the null values. Next, use these dataset to build the model

[53]:
```
from sklearn.preprocessing import LabelEncoder
labelcoder = LabelEncoder()
```

[54]:
```
for col in cat_col:
    train[col] = labelcoder.fit_transform(train[col])
```

[56]:
```
train_data = train.drop(columns='churn',axis = 0)
churn = train['churn']
```

[57]:
```
from sklearn.model_selection import train_test_split
# X_train,X_test,Y_train,Y_test = train_test_split(one_hot_ticket,Survived,␣
 ↪test_size=.30)
X_train,X_test,Y_train,Y_test = train_test_split(train_data,churn, test_size=.
 ↪30)
```

[58]:
```
# machine learning
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
```

[59]:
```
# Logistic Regression
logreg = LogisticRegression()
logreg.fit(X_train, Y_train)
Y_pred = logreg.predict(X_test)
acc_log = round(logreg.score(X_test, Y_test) * 100, 2)

# Stochastic Gradient Descent
sgd = SGDClassifier()
sgd.fit(X_train, Y_train)
Y_pred = sgd.predict(X_test)
acc_sgd = round(sgd.score(X_test, Y_test) * 100, 2)
acc_sgd
```

```
# Support Vector Machines
svc = SVC()
svc.fit(X_train, Y_train)
Y_pred = svc.predict(X_test)
acc_svc = round(svc.score(X_test, Y_test) * 100, 2)
acc_svc

# KNN
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, Y_train)
Y_pred = knn.predict(X_test)
acc_knn = round(knn.score(X_test, Y_test) * 100, 2)
acc_knn

# Gaussian Naive Bayes

gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)
Y_pred = gaussian.predict(X_test)
acc_gaussian = round(gaussian.score(X_test, Y_test) * 100, 2)
acc_gaussian

# Perceptron

perceptron = Perceptron()
perceptron.fit(X_train, Y_train)
Y_pred = perceptron.predict(X_test)
acc_perceptron = round(perceptron.score(X_test, Y_test) * 100, 2)
acc_perceptron
# Linear SVC

linear_svc = LinearSVC()
linear_svc.fit(X_train, Y_train)
Y_pred = linear_svc.predict(X_test)
acc_linear_svc = round(linear_svc.score(X_test, Y_test) * 100, 2)
acc_linear_svc

# Decision Tree

decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, Y_train)
Y_pred = decision_tree.predict(X_test)
acc_decision_tree = round(decision_tree.score(X_test, Y_test) * 100, 2)
acc_decision_tree

# Random Forest
```

```
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, Y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_train, Y_train)
acc_random_forest = round(random_forest.score(X_test, Y_test) * 100, 2)
acc_random_forest
```

```
/home/brian/miniconda3/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:763: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/brian/miniconda3/lib/python3.8/site-packages/sklearn/svm/_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

[59]: 90.21

[60]:
```
# drop string
models = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest', 'Naive Bayes', 'Perceptron',
              'Stochastic Gradient Decent', 'Linear SVC',
              'Decision Tree'],
    'Accuracy_Score': [acc_svc, acc_knn, acc_log,
              acc_random_forest, acc_gaussian, acc_perceptron,
              acc_sgd, acc_linear_svc, acc_decision_tree]})
models.sort_values(by='Accuracy_Score', ascending=False)
```

[60]:

	Model	Accuracy_Score
3	Random Forest	90.21
2	Logistic Regression	89.94
0	Support Vector Machines	89.92
7	Linear SVC	89.92
6	Stochastic Gradient Decent	89.56
1	KNN	87.74
4	Naive Bayes	86.91
8	Decision Tree	83.12
5	Perceptron	26.01

[61]:
```
from sklearn.decomposition import PCA
from sklearn.decomposition import TruncatedSVD
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

[62]:
```
myPCA = PCA(10)
mySVD = TruncatedSVD(10)
myLDA = LinearDiscriminantAnalysis(10)
```

[64]:
```
myPCA.fit(X_train)
```

[64]:
```
PCA(n_components=10)
```

[65]:
```
RX_train = myPCA.transform(X_train)
RX_test = myPCA.transform(X_test)
```

[67]:
```
# Random Forest
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(RX_train, Y_train)
Y_pred = random_forest.predict(RX_test)
random_forest.score(RX_train, Y_train)
acc_random_forest = round(random_forest.score(RX_test, Y_test) * 100, 2)
acc_random_forest
```

[67]:
```
90.14
```

[69]:
```
# Logistic Regression
logreg = LogisticRegression()
logreg.fit(X_train, Y_train)
Y_pred = logreg.predict(X_test)
acc_log = round(logreg.score(X_test, Y_test) * 100, 2)
acc_log
```

```
/home/brian/miniconda3/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:763: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[69]:
```
89.94
```

[ ]:
```
```