jupyter task_2_descriptive_analysis Last Checkpoint: a few seconds ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

Exploratory Data Analysis & Data Cleaning

Understanding the business through data

Sub-Task 1:

Clean the data – you might have to address missing values, duplicates, data type conversions, transformations, and multicolinearity, as well as outliers.

Sub-Task 2:

Perform some exploratory data analysis. Look into the data types, data statistics, and identify any missing data or null values, and how often they appear in the data. Visualize specific parameters as well as variable distributions.

In [1]:

```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
files= os.listdir('.')
for i in files:
    print(i)
```

```
.ipynb_checkpoints
ml_case_training_data.csv
ml_case_training_hist_data.csv
ml_case_training_output.csv
task-2.docx
task1-task-description.pdf
task_2_descriptive_analysis.ipynb
~$task-2.docx
```

In [3]:

```
df_train_data= pd.read_csv('ml_case_training_data.csv')
df_train_hist= pd.read_csv('ml_case_training_hist_data.csv')
df_train_output= pd.read_csv('ml_case_training_output.csv')
```

In [4]:

```
# DESCRIPTIVE analysis
df_train_data.describe()
```

Out[4]:

	campaign_disc_ele	cons_12m	cons_gas_12m	cons_last_month	forecast_base_bill_ele	forecast_base_bill_year	fore
count	0.0	1.609600e+04	1.609600e+04	1.609600e+04	3508.000000	3508.000000	
mean	NaN	1.948044e+05	3.191164e+04	1.946154e+04	335.843857	335.843857	
std	NaN	6.795151e+05	1.775885e+05	8.235676e+04	649.406000	649.406000	
min	NaN	-1.252760e+05	-3.037000e+03	-9.138600e+04	-364.940000	-364.940000	
25%	NaN	5.906250e+03	0.000000e+00	0.000000e+00	0.000000	0.000000	
50%	NaN	1.533250e+04	0.000000e+00	9.010000e+02	162.955000	162.955000	
75%	NaN	5.022150e+04	0.000000e+00	4.127000e+03	396.185000	396.185000	
max	NaN	1.609711e+07	4.188440e+06	4.538720e+06	12566.080000	12566.080000	

8 rows × 22 columns

NULL VALUES

In [5]:

```
df_train_data.isnull().sum()
```

Out[5]:

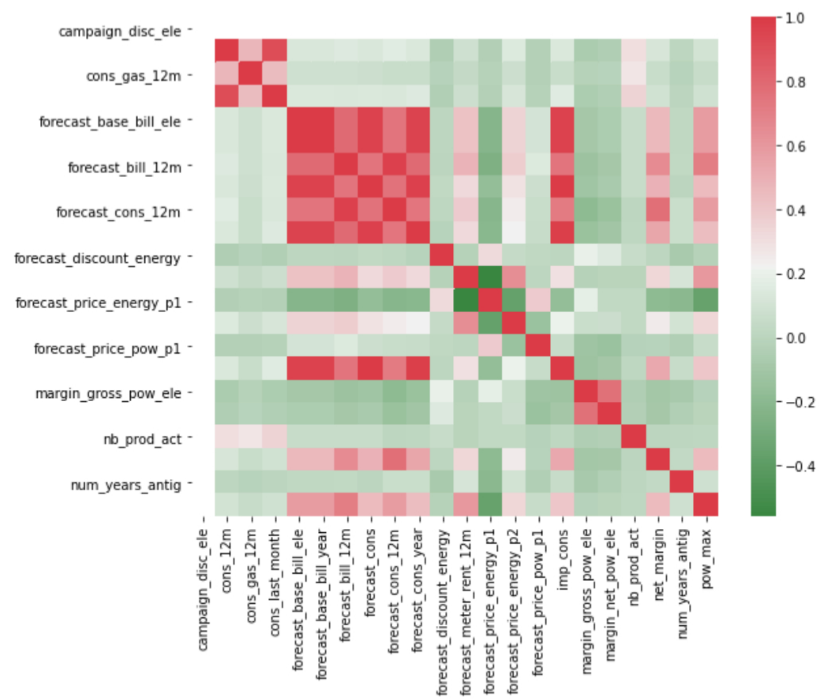
```
id                              0
activity_new                 9545
```

```
campaign_disc_ele            16096
channel_sales                 4218
cons_12m                         0
cons_gas_12m                     0
cons_last_month                  0
date_activ                       0
date_end                         2
date_first_activ             12588
date_modif_prod                157
date_renewal                    40
forecast_base_bill_ele       12588
forecast_base_bill_year      12588
forecast_bill_12m            12588
forecast_cons                12588
forecast_cons_12m                0
forecast_cons_year               0
forecast_discount_energy       126
forecast_meter_rent_12m          0
forecast_price_energy_p1       126
forecast_price_energy_p2       126
forecast_price_pow_p1          126
has_gas                          0
imp_cons                         0
margin_gross_pow_ele            13
margin_net_pow_ele              13
nb_prod_act                      0
net_margin                      15
num_years_antig                  0
origin_up                       87
pow_max                          3
dtype: int64
```

In [6]:
```
df_train_data.duplicated().sum()
```

Out[6]: 0

In [7]:
```
corr_data= df_train_data.corr()
sns.heatmap(corr_data, xticklabels= corr_data.columns, cmap=sns.diverging_palette(130, 10, as_cmap=True
plt.gcf().set_size_inches(9,7)
```



HISTORY of data

In [8]:
```
df_train_hist.describe()
```

Out[8]:

	price_p1_var	price_p2_var	price_p3_var	price_p1_fix	price_p2_fix	price_p3_fix
count	191643.000000	191643.000000	191643.000000	191643.000000	191643.000000	191643.000000
mean	0.140991	0.054412	0.030712	43.325546	10.698201	6.455436
std	0.025117	0.050033	0.036335	5.437952	12.856046	7.782279
min	0.000000	0.000000	0.000000	-0.177779	-0.097752	-0.065172
25%	0.125976	0.000000	0.000000	40.728885	0.000000	0.000000
50%	0.146033	0.085483	0.000000	44.266930	0.000000	0.000000

50%	0.146033	0.085483	0.000000	44.266930	0.000000	0.000000
75%	0.151635	0.101780	0.072558	44.444710	24.339581	16.226389
max	0.280700	0.229788	0.114102	59.444710	36.490692	17.458221

In [9]:
```
df_train_hist.head(3)
```

Out[9]:

	id	price_date	price_p1_var	price_p2_var	price_p3_var	price_p1_fix	price_p2_fix	price_p3
0	038af19179925da21a25619c5a24b745	2015-01-01	0.151367	0.0	0.0	44.266931	0.0	
1	038af19179925da21a25619c5a24b745	2015-02-01	0.151367	0.0	0.0	44.266931	0.0	
2	038af19179925da21a25619c5a24b745	2015-03-01	0.151367	0.0	0.0	44.266931	0.0	

In [ ]:

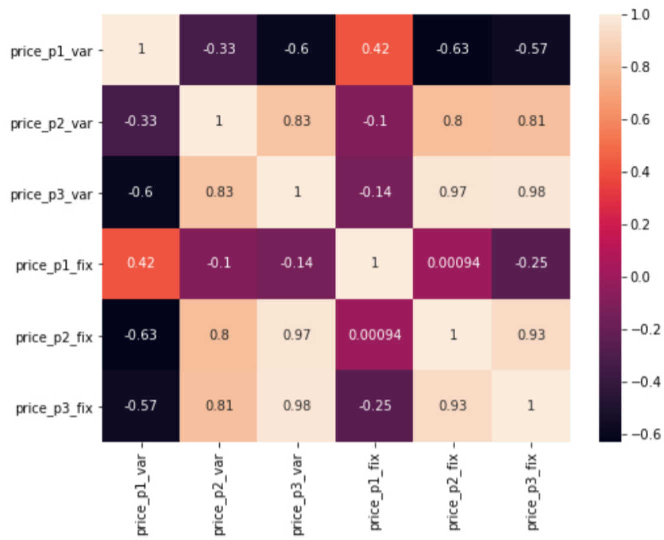
Finding NULL VALUES

In [10]:
```
df_train_hist.isnull().sum()
```

Out[10]:
```
id                 0
price_date         0
price_p1_var    1359
price_p2_var    1359
price_p3_var    1359
price_p1_fix    1359
price_p2_fix    1359
price_p3_fix    1359
dtype: int64
```

Co-orealtionship between variables

In [11]:
```
corr_hist= df_train_hist.corr()
sns.heatmap(corr_hist, xticklabels=corr_hist.columns, yticklabels= corr_hist.columns
            ,annot= True)
plt.gcf().set_size_inches(8,6)
```



In [ ]:

In [ ]:

OUTPUT

In [12]:
```
df_train_output.head(3)
```

Out[12]:

	id	churn
0	48ada52261e7cf58715202705a0451c9	0
1	24011ae4ebbe3035111d65fa7c15bc57	1
2	d29c2c54acc38ff3c0614d0a653813dd	0

In [13]:
```
df_train_output.describe()
```

Out[13]:

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

FINDING any NULL values

In [14]:
```
df_train_output.isnull().sum()
```

Out[14]:
```
id       0
churn    0
dtype: int64
```

Checking if the ID are duplicates

In [35]:
```
df_train_output['id'].duplicated().sum()
```

Out[35]:
```
0
```

APPLYING label encoder for EVERY unique ID to understand the data

In [17]:
```
from sklearn.preprocessing import LabelEncoder
```

In [18]:
```
le= LabelEncoder()
```

In [19]:
```
df_train_output['id_code']= le.fit_transform(df_train_output['id'])
```

In [20]:
```
df_train_output.head(4)
```

Out[20]:

	id	churn	id_code
0	48ada52261e7cf58715202705a0451c9	0	4666
1	24011ae4ebbe3035111d65fa7c15bc57	1	2361
2	d29c2c54acc38ff3c0614d0a653813dd	0	13250
3	764c75f661154dac3a6c254cd082ea7d	0	7430

In [33]:
```
df_train_output['id_code'].duplicated().sum()
```

Out[33]:
```
0
```

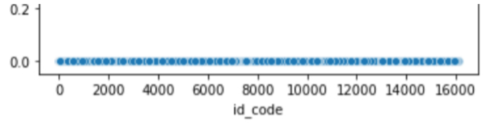
In [ ]:

Co-orelationship Between varaibles

In [23]:
```
corr_out= df_train_output.corr()
sns.scatterplot(df_train_output['id_code'], df_train_output['churn'])
```

```
C:\Users\daiko\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the followi
ng variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `da
ta`, and passing other arguments without an explicit keyword will result in an error or misinterpreta
tion.
  warnings.warn(
```

Out[23]:
```
<AxesSubplot:xlabel='id_code', ylabel='churn'>
```



Distribution based on customers who have 'churn' and who have 'NOT churn'

In [58]:
```
print('CUSTOMERS who have churned: ',df_train_output['churn'][df_train_output['churn'].values == 1].cou
```

```
CUSTOMERS who have churned:  1595
```

In [59]:
```
print('CUSTOMERS who have NOT churned: ',df_train_output['churn'][df_train_output['churn'].values == 0]
```

```
CUSTOMERS who have NOT churned:  14501
```

In [ ]: