Exploratory analysis

The propose of this notebook is to dig deeper into the dataset. There are more than 300 million transactions provided that were sampled into a 3 million dataset to be evaluated before getting to the full data.

The preprocessing script to create this sample is in the root folder of this project and is divided into two parts:

- **data-cleaner:** responsible for removing unecessary columns and converting the csv files to `utf-8` encoding.
- **data-sampler:** responsible for splitting the data into samaller sets and collecting 15 out of each 100 records.

These plots and insights will be used to provide a better undestanding of the overall data and its complexity in order to create processes and valuable techniques to better fit the problem.

Loading data

The data were exported in a naive way. This leads to difficulties into loading the types correctly by `pandas`. In orther to fix this, we have to load the dataset, and manually parse numeric columns to the right type. This is a simpler approach than trying to fix each of the 3 million rows of the csv file, but encreases the load type by almost a half.

In [3]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
sns.set()
%matplotlib inline

data = pd.read_csv('./sample.csv', delimiter=';',index_col='NU_AUTORIZACAO', low
_memory=False)
data.sample(n=10)
```

Out[3]:

NU_AUTORIZACAO	CO_UF_FARMACIA	DT_VENDA	HS_VENDA	DT_EMISSAO_RECEITA	CO_M
998468361967405	MG	2015-08-03	1524	2015-05-22	
998468745942288	PR	2018-03-10	959	2017-10-09	
998468771041946	PE	2018-05-10	1523	2018-02-28	
998468881037937	SP	2019-02-05	1305	2019-01-05	
998468722281511	PR	2018-01-10	1750	2017-06-09	
998468290186652	MG	2015-01-20	1808	2014-11-06	
998468418758015	SP	2015-12-31	943	2015-10-27	
998468432057780	GO	2016-02-05	1925	2016-02-05	
998468862649971	SP	2018-12-17	1645	2018-11-13	
998468252302758	SP	2014-10-02	1259	2014-09-11	

10 rows × 25 columns

Processing data

Some columns provided as numbers are not filled correctly, so we have to replace -1 with zeros and also set all NaN occurences to zero.

We have also created a scaler to normalize data in order to make all columns equaly important to the algoriths. The minMaxScaler was selected to provide a range in between 0-1 that is equivalent to the original values based on their variation.

In [6]:

```
data = data.replace(to_replace=-1,value=0)
data = data.fillna(0)
data.sample(n=10)
```

Out[6]:

NU_AUTORIZACAO	CO_UF_FARMACIA	DT_VENDA	HS_VENDA	DT_EMISSAO_RECEITA	CO_M
998468791187651	RS	2018-06-28	1505	2018-02-19	
998468361960844	PE	2015-08-03	1515	2015-05-11	
998468828129102	SP	2018-09-25	950	2018-06-14	
998468864573132	MT	2018-12-22	859	2018-11-03	
998468778640538	RS	2018-05-29	1014	2018-04-19	
998468189124709	SP	2014-03-27	1118	2014-02-05	
998468334499609	MG	2015-05-22	1508	2015-05-22	
998468208449870	RJ	2014-05-26	1347	2014-05-26	
998468660592867	SP	2017-08-14	906	2017-03-23	
998468818327443	BA	2018-09-01	1054	2018-04-24	

10 rows × 25 columns

In [8]:

```
from sklearn.preprocessing import MinMaxScaler

min_max_scaler = MinMaxScaler()
numeric_data = data.select_dtypes(exclude=['object'])
np_scaled = min_max_scaler.fit_transform(numeric_data.values)
data_norm = pd.DataFrame(np_scaled, columns = numeric_data.columns)
```

Exploring and ploting data to create first analysis

The dataset is now ready to be evaluated and plotted in order to collect first impressions about the inner information that it might give us. The seaborn pairplot was used in combination with a hue rotation based on the price paid for each medicine.

In [17]:

```
plot_data = data_norm[['HS_VENDA', 'CO_CPF', 'CO_CNPJ', 'CO_CNPJ_MATRIZ', 'QT_PR
ESCRITA', 'QT_AUTORIZADA', 'QT_SOLICITADA', 'VL_PRECO_SUBSIDIADO', 'VL_PRECO_VEN
DA', 'CO_CO_PRINCIPIO_ATIVO', 'CO_CO_PATOLOGIA']]
```

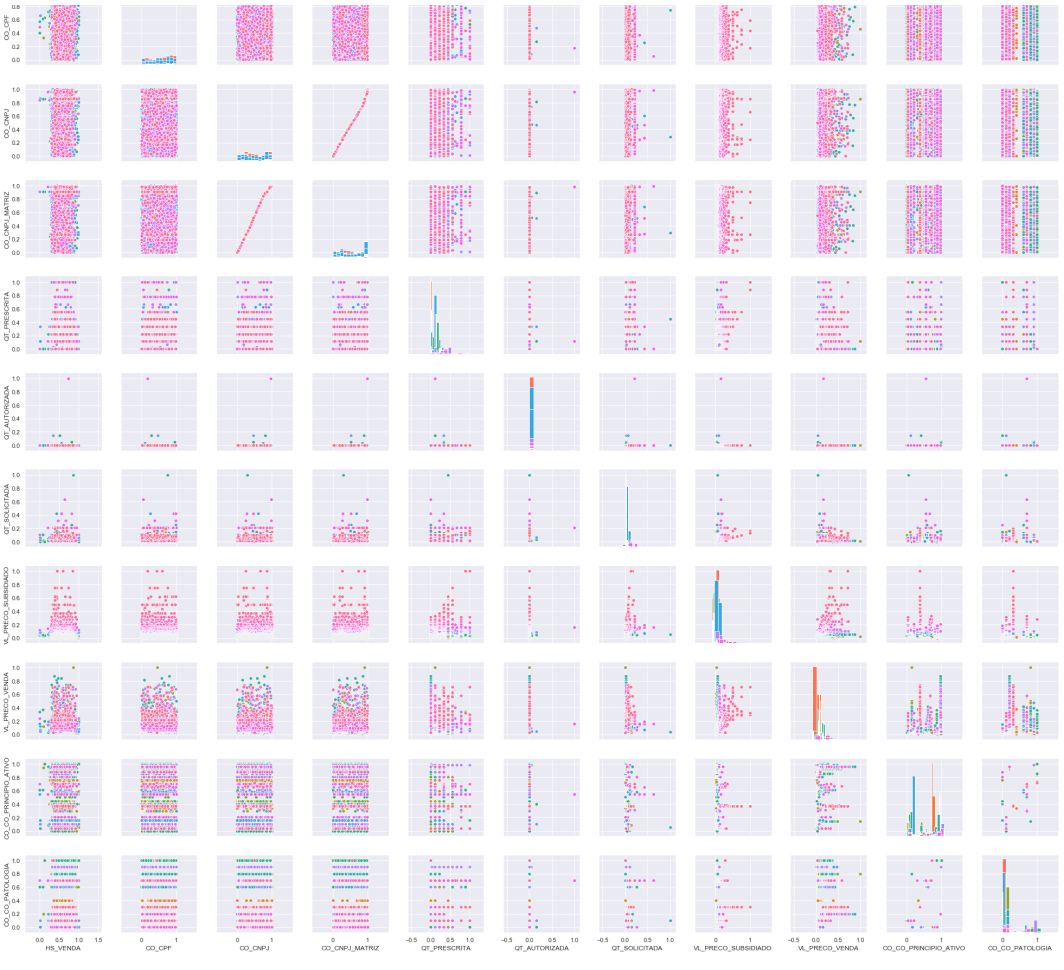
In [16]:

```
sns.pairplot(plot_data, hue='VL_PRECO_SUBSIDIADO', diag_kind='hist')
```

Out[16]:

```
<seaborn.axisgrid.PairGrid at 0x1953e4810>
```


VL_PRECO_SUBSIDIADO
HS_VENDA


NU_CPF
NU_CNPJ
NU_CNPJ_MATRIZ
QT_PRESCRITA
QT_AUTORIZADA
QT_SOLICITADA
VL_PRECO_SUBSIDIADO
VL_PRECO_VENDA
CO_CO_PRINCIPIO_ATIVO
CO_CO_PATOLOGIA
HS_VENDA
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.5
0
1
-0.5
0.5

Naive clustering - KMeans

The kmeans clustering used here is to provide some insights about the groups that are coexisting withing the data under the hood. We have created it and compared to the dataset that includes some occurences of non-conformity based on manual collecting of samples.

In [63]:

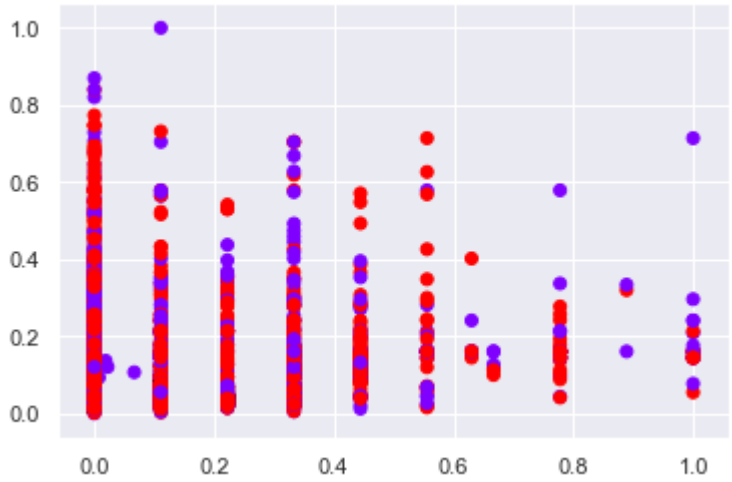
```
from sklearn.cluster import KMeans
import numpy as np

kmeans = KMeans(n_clusters=2, random_state=0).fit(np_scaled)
_labels = kmeans.predict(np_scaled)
```

In [19]:

```
plt.scatter(np_scaled[:,9],np_scaled[:,14], c=_labels, cmap='rainbow')
```

Out[19]:

```
<matplotlib.collections.PathCollection at 0x1f3212390>
```

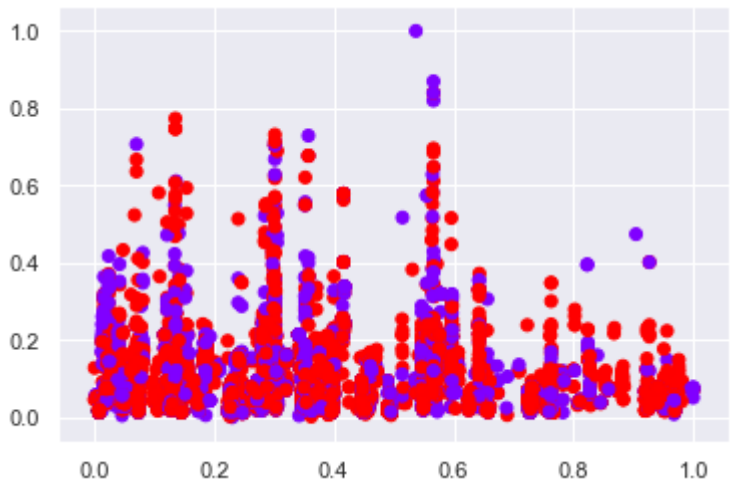


In [20]:

```
plt.scatter(np_scaled[:,15],np_scaled[:,14], c=_labels, cmap='rainbow')
```

Out[20]:

```
<matplotlib.collections.PathCollection at 0x14728b650>
```

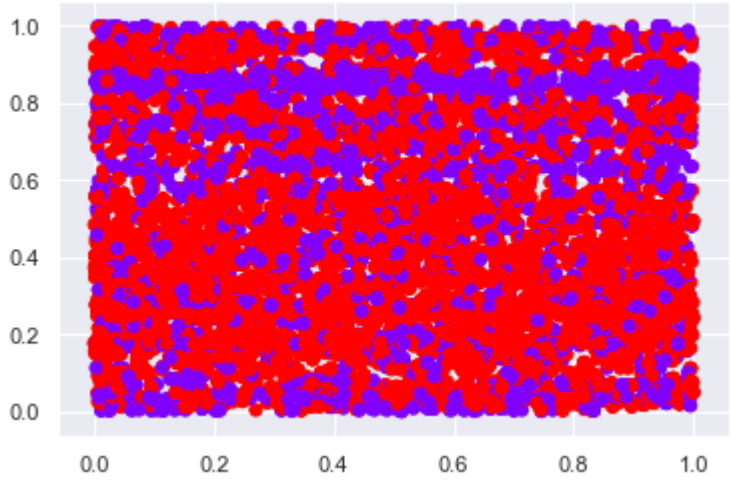


In [21]:

```
plt.scatter(np_scaled[:,19],np_scaled[:,5], c=_labels, cmap='rainbow')
```

Out[21]:

```
<matplotlib.collections.PathCollection at 0x151bbd3d0>
```

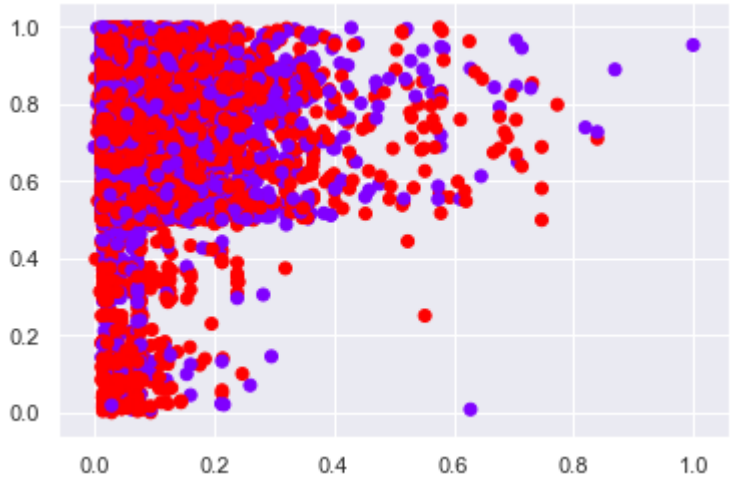


In [22]:

```
plt.scatter(np_scaled[:,14],np_scaled[:,3], c=_labels, cmap='rainbow')
```

Out[22]:

```
<matplotlib.collections.PathCollection at 0x146cc9810>
```



In [23]:

```
from sklearn.metrics import silhouette_score

silhouette_avg = silhouette_score(np_scaled, _labels)

silhouette_avg
```

Out[23]:

```
0.20391282918990958
```

Analysing plots

The plots provided indicate that the clusters are not so well defined and have hard to find boundaries, but the seccond plot gave us a clue about the formation of them. One has higher subsidized values and higher requested amounts, in an horizontal line, the other is more distributed, but with lower values and requested amounts.

Loading Labeled data

The labeled dataset loaded will be filtered inside the sample that we collected, this way we can compare the cluster labels with the occurence of inconsistencies and see if there is a cluster that better represents the ones that are consistent.

In [9]:

```
labeled_data = pd.read_csv('data/labels_occ.csv', delimiter=';', low_memory=False)
data_norm
# data_norm_ = data_norm.reset_index()
```

Out[9]:

	HS_VENDA	CO_MUNICIPIO_IBGE_FARMACIA	CO_CPF	CO_PRESCRITOR	CO_CRF
0	0.387452	0.263730	0.840303	0.824840	0.647996
1	0.595591	0.763362	0.605561	0.677788	0.100443
2	0.595591	0.763362	0.605561	0.677788	0.100443
3	0.595591	0.763362	0.605561	0.677788	0.100443
4	0.442560	0.762196	0.438329	0.567832	0.122351
...	...	...	...	...	...
1799994	0.737177	0.581723	0.467142	0.240655	0.639750
1799995	0.737177	0.581723	0.467142	0.240655	0.639750
1799996	0.469691	0.477654	0.111986	0.818827	0.447404
1799997	0.607885	0.500344	0.715533	0.513522	0.508139
1799998	0.441713	0.574509	0.450601	0.503153	0.610770

1799999 rows × 20 columns

In [11]:

```
# data_norm.loc[data_norm['NU_AUTORIZACAO'].isin(labeled_data['NU_AUTORIZACAO'].
values)]
data_i = data.reset_index()
merged = pd.merge(labeled_data, data_i, how='inner')
min_max_scaler = MinMaxScaler()
numeric_data = data.select_dtypes(exclude=['object'])
np_scaled = min_max_scaler.fit_transform(numeric_data.values)
data_norm = pd.DataFrame(np_scaled, columns = numeric_data.columns)
merged
```

Out[11]:

	NU_AUTORIZACAO	COD_OCORRENCIAS	NU_OCORRENCIAS	CO_UF_FARMACIA	DT_VEN
0	998468555314231	1, 8	2	GO	2016-12
1	998468555314231	1, 8	2	GO	2016-12
2	998468326940063	13, 8(CPF)	2	MG	2015-05
3	998468326940063	13, 8(CPF)	2	MG	2015-05
4	998468326940063	13, 8(CPF)	2	MG	2015-05
...	...	...	...	...	
92	998468445910620	13, 8(CPF)	2	RJ	2016-03
93	998468519953277	13, 8(CPF)	2	RJ	2016-09
94	998468519953277	13, 8(CPF)	2	RJ	2016-09
95	998468519953277	13, 8(CPF)	2	RJ	2016-09
96	998468519953277	13, 8(CPF)	2	RJ	2016-09

97 rows × 28 columns

In [71]:

```
min_max_scaler = MinMaxScaler()
numeric_merged = merged.select_dtypes(exclude=['object'])
np_merged_scaled = min_max_scaler.fit_transform(numeric_merged.values)
merged_norm = pd.DataFrame(np_merged_scaled, columns = numeric_merged.columns)
merged_norm.drop(['NU_OCORRENCIAS'], axis=1, inplace=True)
merged_norm.set_index('NU_AUTORIZACAO', inplace=True)
merged_labels = kmeans.predict(merged_norm)

unique, counts = np.unique(merged_labels, return_counts=True)
merged_res = dict(zip(unique, counts))
merged_res
```

Out[71]:

```
{0: 56, 1: 38}
```

In [15]:

```
unique, counts = np.unique(_labels, return_counts=True)
data_res = dict(zip(unique, counts))
data_res
```

Out[15]:

```
{0: 684991, 1: 1115008}
```

In [16]:

```
group_zero_P = 684991/(684991+1115008)
occ_zero_P = 56/(56+38)


print("probability of transactions in group 0: ", group_zero_P)
print("probability of occurrences in group 1: ", occ_zero_P)
```

```
probability of transactions in group 0:  0.38055076697264834
probability of occurrences in group 1:  0.5957446808510638
```

Probabilities and Bayes Theorem

If the probability of a transaction to have inconsistencies(I) is X. Given that the probability of a transaction to be in group 0 (A) is 0.38 and in group 1 (B) is 0.62, and considering that the probability of an incinsistent transaction of being in group 0 (A|I) is 0.60 and 0.4 in group 1 (B|I). We have that:

$$P(I) = X$$

$$P(A) = 0.38$$
$$P(B) = 0.62$$

$$P(A|I) = 0.6$$
$$P(B|I) = 0.4$$

$$P(I|A) = P(A|I)*P(I)/P(A)$$

Then, the probability of finding an inconsistency in group A is:

$$P(I|A) = 0.6*X/0.38$$

$$P(I|A) = 1,578*X$$

And in group B is:

$$P(I|B) = 0.4*X/0.62$$

$$P(I|B) = 0,64*X$$

That give us an smaller set of transactions to analyse and possibly a smaller dataset for further analysis. In order to confir or avoid this insight, we now have other paths to explore.

Nest steps

Now we are working on the following steps:

```
- Rising data about inactive pharmacies to compare their transactions with
the provided clustered groups.
- Compare data of provided clusters with GMM clusters.
- Prepare a fraud dataset to compare with the provided clusters (DenaSUS).
- Generate a classification system based on both analysis to identify inco
nsistent and fraudulent suspect transactions.
```