BirminghamProjectML

November 19, 2021

1 Purchase card transactions for the Birmingham City Council Project

By Andrés A. Aristizábal P.

We follow the Data Analysis process which includes 6 main steps, but we will include just the first 5 ones (since we are not going to deploy the product):

1. Business' Questions
2. Data Collection
3. Exploratory Data Analysis
4. Modeling
5. Evaluation

1.1 1. Business Questions

For this project we are using a historical dataset which includes a collection of purchase card transactions for the Birmingham City Council.

There are several important tasks which can be performed in order to get the most of the available data.

1.1.1 1.1 We list the possible tasks that could be performed and are interesting enough to be analyzed and taked into account for this project:

1. (Clustering) Discovering profiles.
2. (Anomalies detection) Unusual activities.
3. (Forecasting) Predicting future transactional behaviors (Next purchase, Expenditures forecasting, etc.)
4. (Creativity) Stating a problem that may be solved using the available data.

1.1.2 1.2 By taking into account these tasks we establish a series of important questions which could be answered by following the Data Analysis Process stated previously.

1. Clustering: a) which are the all around purchase card transactions' profiles?, b) which are the credit card customers' profiles?, c) which are the merchants' profiles?

2. Anomalies detection: a) which transactions are unusual and may be fraudulent?
3. Forecasting: what would the next purchase be for a particular client?, b) how much would a client spend in the next period (days, month, years)?, c) how much would be the income for a particular merchant in the next period (days, months, years)?
4. Creativity: Regression: Taking into account a time frame, what would be, for a customer, the total amount of money spent? Classification: Taking into account a time frame: a) what would be, for a card transaction, the most probable type of consumption (Vehicle Fuel, Hospitality, Food, Accomodation, etc.)?, b) what would be, for a customer, the most probable type of consumption (Vehicle Fuel, Hospitality, Food, Accomodation, etc.)?, c) what would be, for a card transaction, the most probable company in which the money will be spent?, d) what would be, for a customer, the most probable company in which he/she would spend his/her money?, e) what would be the most probable card transaction associated to a directorate?, f) what would be the most probable card transaction Value Added Tax (VAT)?

Although these are the main questions that any data scientist may try to answer by exploring, cleaning and analyzing the available data, for the purpose of this test, we will focus on three specific questions:

1) Clustering:
   a) which are the credit card customers' profiles for the year 2014?
   b) which are the all around purchase card transactions' profiles for the year 2014?
2) Forecasting: how much would a client spend in the next period (days, month, years)?

Even though we will be mainly answering the latest three questions using several models, we need to collect the data and do an exploratory analysis. During this EDA we will observe, gain insights, clean, fix and perform other activities related, not only to the 2 main tasks in hand, but looking further, towards preparing the data so many other questions, related to other tasks, could be answered.

Since the scope of this test is not that broad we will focus on the three previously presented questions.

1.2 2. Data Collection

1.2.1 2.1 Importing the necessary libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import math
import pickle
import pmdarima as pm
import warnings
warnings.filterwarnings('ignore')

from sklearn.preprocessing import StandardScaler
```

```
from kmodes.kmodes import KModes
from difflib import SequenceMatcher
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from collections import Counter
from sklearn.metrics import mean_squared_error, accuracy_score,␣
↪silhouette_samples, silhouette_score, calinski_harabasz_score,␣
↪cohen_kappa_score, mean_absolute_error
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from statsmodels.tsa.statespace.sarimax import SARIMAX
from fbprophet import Prophet
from prophet.plot import plot_plotly, plot_components_plotly
from xgboost import XGBRegressor
from sklearn.model_selection import TimeSeriesSplit, GridSearchCV
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D,␣
↪Softmax, BatchNormalization, Dropout
from tensorflow.keras.wrappers.scikit_learn import KerasRegressor
```

1.2.2 2.2 Merging the data by years

We define a function that will take the months, the years, all the xls files and the path to where the files are saved in order to obtain a list of the dataframes that will include the information per year.

[2]:
```
def merging_xls_in_df(months, years, files, path):
    dfs = list()
    for y in years:
        df = pd.DataFrame()
        for m in months:
            for f in files:
                if f.endswith(".xls") and m in f and y in f:
                    df = df.append(pd.read_excel(path+'/'+f, parse_dates =␣
↪True))
        dfs.append(df)
    return dfs
```

We declare the lists of months, years and files as well as the path in which the files will be found in order to be opened and included in each particular dataframe per year.

[3]:
```
months =␣
↪['jan','feb','mar','apr','may','jun','jul','aug','sept','oct','nov','dec']
years = ['2014','2015','2016','2017','2018']
destdir = '/Users/andresaristi/Documents/BirminghamCardTransactions/dataset'
files = [ f for f in os.listdir(destdir) if os.path.isfile(os.path.
↪join(destdir,f)) ]
```

We obtain the list of dataframes per year and later we take out 5 dataframes (2014, 2015, 2016, 2017, and 2018).

We will take these 5 dataframes and we will apply the Exploratory Data Analysis to each of them in order to try to understand the clients or the merchants by means of a clustering algorithm, for example, in a time frame of a year. Also we may want to use just one year to classify clients or companies (for example) as we already explained in our business questions' section.

We will also concatenate the 5 dataframes in order to obtain more data so we may be able to perform some forecasting. It does not mean that we cannot do forecasting with information of just one year, but we may find more interesting patterns with data in a longer time frame.

We highlight the fact that we do not have the information related to december of 2017, since that csv file involves information not related to the business.

We add 2015 to the purchase card transactions xls files of february and march in order for our merging function to work according to plan.

[4]:
```
dfs = merging_xls_in_df(months, years, files, destdir)
df2014 = dfs[0]
df2015 = dfs[1]
df2016 = dfs[2]
df2017 = dfs[3]
df2018 = dfs[4]
df = pd.concat([df2014,df2015,df2016,df2017,df2018],axis=0)
df.head()
```

[4]:

	TRANS DATE	TRANS VAT DESC	ORIGINAL GROSS AMT		MERCHANT NAME
0	2014-04-29	VR	52.32		shell kings 587
1	2014-04-04	VR	65.82		shell kings 587
2	2014-04-07	VR	41.35		tesco pfs 2484
3	2014-04-25	VR	47.19	bush	service station
4	2014-04-24	VR	98.08	bush	service station

	CARD NUMBER	BILLING CUR CODE	TRANS CAC CODE 1	TRANS CAC DESC 1
0	**********5770	GBP	K020	Vehicle Fuel
1	**********5770	GBP	K020	Vehicle Fuel
2	**********8738	GBP	K020	Vehicle Fuel
3	**********2997	GBP	K020	Vehicle Fuel
4	**********2997	GBP	K020	Vehicle Fuel

	TRANS CAC CODE 2	TRANS CAC DESC 2	TRANS CAC CODE 3
0	RV12N	African-Caribbean DC	A00
1	RV12N	African-Caribbean DC	A00
2	RV11Y	Marsh Lane Dce, 79, B23	A00
3	RV1K1	Elders Group - Erdington	A00
4	RV1K1	Elders Group - Erdington	A00

	Directorate	Unnamed: 10	Directorate	Directorates	ORIGINAL CUR

```
0  Adult & Communities         NaN          NaN          NaN          NaN
1  Adult & Communities         NaN          NaN          NaN          NaN
2  Adult & Communities         NaN          NaN          NaN          NaN
3  Adult & Communities         NaN          NaN          NaN          NaN
4  Adult & Communities         NaN          NaN          NaN          NaN

   BILLING GROSS AMT  TRANS TAX AMT BILLING CUR CODE.1
0                NaN            NaN                NaN
1                NaN            NaN                NaN
2                NaN            NaN                NaN
3                NaN            NaN                NaN
4                NaN            NaN                NaN
```

We save the list of dataframes in a .pkl file.

[5]:
```
dfs_pkl_filename = '/Users/andresaristi/Documents/BirminghamCardTransactions/
↪pickle_files/dfs.pkl'
dfs_pkl = open(dfs_pkl_filename, 'wb')
pickle.dump(dfs, dfs_pkl)
dfs_pkl.close()
```

We show that the file for december 2017 does not have anything in common with the other xls files, so it cannot be taked into account for the analysis.

[6]:
```
dfdecember2017 = pd.read_csv(destdir+'/'+'open_data_planning_weekly_list.csv')
dfdecember2017.head()
```

[6]:
```
       REFERENCE     Category Received_Date  \
0  2018/00188/PA  Householder    11/01/2018
1  2017/10761/PA       Advert    18/12/2017
2  2018/00011/PA  Householder    03/01/2018
3  2018/00117/PA  Householder    08/01/2018
4  2018/00191/PA  Householder    11/01/2018

                                          LOCATION  \
0  120 Oakfield Road, Selly Oak, Birmingham, B29 7ED
1  Arya International Mini Market, 568 Bristol Ro…
2              16 Wyvern Grove, Birmingham, B29 6RN
3    9 Naunton Close, Selly Oak, Birmingham, B29 4DX
4  145 Lichfield Road, Four Oaks, Sutton Coldfiel…

                                              DEV    Accepted  \
0  Replacement doors and windows, external insula…  11/01/2018
1   Retention of internally illuminated ATM surround  11/01/2018
2           Erection of single storey rear extension  10/01/2018
3             Erection of two storey front extension  12/01/2018
4    Erection of two storey side and rear extensions  11/01/2018
```

```
                                                APPLICANT  \
0  Mr and Mrs M Webb,120 Oakfield Road, Selly Oak…
1  Cardtronics UK Ltd T/A Cashzone,PO Box 476, Ha…
2   Mr J Foxall,16 Wyvern Grove, Birmingham, B29 6RN
3  Mr Jose Sodre,9 Naunton Close, Selly Oak, Birm…
4  Mr Andrew Haslehurst,145 Lichfield Road, Four …

                                                 AGENT              WARD  geom
0  Architecture For You, 236 Franklin Road, Bourn…         Selly Oak   NaN
1  Des Ager Design & Planning Consultant, 1 Turne…         Selly Oak   NaN
2  Anglian Home Improvements, National Administra…         Selly Oak   NaN
3                                                 ,             Weoley   NaN
4  K G Bramwell MCIAT, 65 Brookhus Farm Road, Wal…  Sutton Four Oaks   NaN
```

1.3 3. Exploratory Data Analysis

As stated before, we are going to do an EDA for the 2014 dataframe. This in order to do different kind of analysis.

1.3.1 3.1 Dataframe year 2014

We return the first 5 rows (observations or data points) sorted by transaction date in order to see the different features involved in this dataframe. We can observe 12 features. Later on we will se that we can establish our transaction date as our index instead of our predefined numeric index.

[7]:
```
df2014.sort_values(by='TRANS DATE').head()
```

[7]:
```
      TRANS DATE TRANS VAT DESC  ORIGINAL GROSS AMT             MERCHANT NAME  \
28   2013-11-12             VR             -594.00    www.pcc.nhs.uk/eventma
2316 2014-03-06             VR               98.33  bp snax 24 weoley castle
1679 2014-03-27             VZ              -10.54        asda home delivery
2291 2014-04-01             VR               94.93     bp snax 24 cofton 859
682  2014-04-02             VZ              200.00          surveymonkey.com

           CARD NUMBER BILLING CUR CODE TRANS CAC CODE 1   TRANS CAC DESC 1  \
28    ************7557              GBP             L100  Equip Operational
2316  ************5353              GBP             K020       Vehicle Fuel
1679  ************9410              GBP             L220     Purchases Food
2291  ************5044              GBP             K020       Vehicle Fuel
682   ************4084              GBP             L9Y0    Computing Other

     TRANS CAC CODE 2              TRANS CAC DESC 2 TRANS CAC CODE 3  \
28              RVA0K                 Admin Support              A00
2316            RHAAC      E'baston CBHO 1 C'takers              A00
1679            REAJP  Perry Common Junior & Infant              A00
2291            RHF03           Northfield Hsg Mgmt              A00
```

```
682             RVP0J             CP Cwide Team Bus               A00

              Directorate
28    Adult & Communities
2316       Local Services
1679        CYP&F SCHOOLS
2291       Local Services
682                CYP&F
```

We confirm the amount of 12 features and find out that we are dealing with a dataset of 34467 data points.

[8]:
```
df2014.shape
```

[8]:
```
(34467, 12)
```

We found out that there are certain missing values for some of our features:

- TRANS VAT DESC: 677
- TRANS CAC CODE 1: 1
- TRANS CAC DESC 1: 1
- TRANS CAC CODE 2: 9
- TRANS CAC DESC 2: 9
- TRANS CAC CODE 3: 12

[9]:
```
df2014.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 34467 entries, 0 to 3292
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   TRANS DATE          34467 non-null  datetime64[ns]
 1   TRANS VAT DESC      33790 non-null  object
 2   ORIGINAL GROSS AMT  34467 non-null  float64
 3   MERCHANT NAME       34467 non-null  object
 4   CARD NUMBER         34467 non-null  object
 5   BILLING CUR CODE    34467 non-null  object
 6   TRANS CAC CODE 1    34466 non-null  object
 7   TRANS CAC DESC 1    34466 non-null  object
 8   TRANS CAC CODE 2    34458 non-null  object
 9   TRANS CAC DESC 2    34458 non-null  object
 10  TRANS CAC CODE 3    34455 non-null  object
 11  Directorate         34467 non-null  object
dtypes: datetime64[ns](1), float64(1), object(10)
memory usage: 3.4+ MB
```

We are going to try to find some patterns related to those missing values.

[10]:
```
df2014[df2014['TRANS VAT DESC'].isna()].head(20)
```

[10]:

	TRANS DATE	TRANS VAT DESC	ORIGINAL GROSS AMT	MERCHANT NAME
981	2014-04-11	NaN	52.81	w m morrisons
1024	2014-04-28	NaN	75.03	js online grocery
1030	2014-04-07	NaN	282.26	viking
1039	2014-04-24	NaN	40.11	js online grocery
1040	2014-04-17	NaN	82.74	js online grocery
1041	2014-04-16	NaN	45.75	js online grocery
1043	2014-04-10	NaN	50.48	js online grocery
1046	2014-04-03	NaN	80.81	js online grocery
1067	2014-05-01	NaN	10.91	tesco stores 6227
1069	2014-04-28	NaN	19.61	tesco stores 6227
1070	2014-04-08	NaN	10.12	tesco stores 6227
1075	2014-04-10	NaN	3.58	tesco stores 5207
1102	2014-04-10	NaN	43.18	w m morrisons
1186	2014-04-04	NaN	41.78	sainsbury's s/mkt
1233	2014-04-17	NaN	37.55	mcdonalds
1242	2014-04-08	NaN	124.47	waitrose 150
1256	2014-04-07	NaN	137.64	asda home delivery
1277	2014-04-15	NaN	746.75	makro deliveries
1343	2014-04-04	NaN	162.10	tesco stores 5854
1352	2014-04-28	NaN	153.49	post office shop

	CARD NUMBER	BILLING CUR CODE	TRANS CAC CODE 1	TRANS CAC DESC 1
981	**********7911	GBP	M900	Hospitality
1024	**********0073	GBP	L220	Purchases Food
1030	**********0065	GBP	L420	Stationery
1039	**********9807	GBP	L220	Purchases Food
1040	**********9807	GBP	L220	Purchases Food
1041	**********9807	GBP	L220	Purchases Food
1043	**********9807	GBP	L220	Purchases Food
1046	**********9807	GBP	L220	Purchases Food
1067	**********1490	GBP	L220	Purchases Food
1069	**********1490	GBP	L220	Purchases Food
1070	**********1490	GBP	L220	Purchases Food
1075	**********4886	GBP	L220	Purchases Food
1102	**********6741	GBP	L100	Equip Operational
1186	**********3783	GBP	L220	Purchases Food
1233	**********5752	GBP	M900	Hospitality
1242	**********3759	GBP	L100	Equip Operational
1256	**********1339	GBP	L220	Purchases Food
1277	**********1436	GBP	MC70	Supplies & Sev Mic
1343	**********9520	GBP	L220	Purchases Food
1352	**********6976	GBP	MC70	Supplies & Sev Mic

	TRANS CAC CODE 2	TRANS CAC DESC 2	TRANS CAC CODE 3

```
981             REARM                       Wyndcliffe JI             A00
1024            REAAB       The City of Birmingham School             A00
1030            REAAB       The City of Birmingham School             A00
1039            REARK                     Wychall Farm JI             A5F
1040            REARK                     Wychall Farm JI             A5F
1041            REARK                     Wychall Farm JI             A00
1043            REARK                     Wychall Farm JI             A5F
1046            REARK                     Wychall Farm JI             A5F
1067            REAAB       The City of Birmingham School             A00
1069            REAAB       The City of Birmingham School             A00
1070            REAAB       The City of Birmingham School             A00
1075            REAAB       The City of Birmingham School             A00
1102            REAGW                  Marlborough Infant             A00
1186            REANH     St Saviour's CE Junior & Infant             A00
1233            REANC  St Paul's RC Junior & Infant  (NC)             A00
1242            REAHG           Moor Hall Junior & Infant             A00
1256            REAYH                             Mayfield            A00
1277            REAZU            Selly Oak Nursery School             A00
1343            REAFA           Highfield Junior & Infant             A00
1352            REAZM                  Marsh Hill Nursery             A00

       Directorate
981   CYP&F SCHOOLS
1024  CYP&F SCHOOLS
1030  CYP&F SCHOOLS
1039  CYP&F SCHOOLS
1040  CYP&F SCHOOLS
1041  CYP&F SCHOOLS
1043  CYP&F SCHOOLS
1046  CYP&F SCHOOLS
1067  CYP&F SCHOOLS
1069  CYP&F SCHOOLS
1070  CYP&F SCHOOLS
1075  CYP&F SCHOOLS
1102  CYP&F SCHOOLS
1186  CYP&F SCHOOLS
1233  CYP&F SCHOOLS
1242  CYP&F SCHOOLS
1256  CYP&F SCHOOLS
1277  CYP&F SCHOOLS
1343  CYP&F SCHOOLS
1352  CYP&F SCHOOLS
```

Since we do not see a clear pattern, and at the moment, the VAT (Value Added Tax) seems like an interesting feature (for instance, for a classification task), and we do not want to eliminate rows so soon (it is just a 2% of the data), we decide to establish another category as 'other'.

[11]:
```
df2014.loc[df2014['TRANS VAT DESC'].isna(),'TRANS VAT DESC'] = 'other'
```

Since the amount of missing values for the other features does not seeem considerable, we eliminate the rows.

[12]:
```
df2014 = df2014[~df2014['TRANS CAC CODE 1'].isna()]
df2014 = df2014[~df2014['TRANS CAC DESC 1'].isna()]
df2014 = df2014[~df2014['TRANS CAC CODE 2'].isna()]
df2014 = df2014[~df2014['TRANS CAC DESC 2'].isna()]
df2014 = df2014[~df2014['TRANS CAC CODE 3'].isna()]
```

[13]:
```
df2014.describe(include='all').T
```

[13]:

	count	unique	top	\
TRANS DATE	34453	253	2014-06-11 00:00:00	
TRANS VAT DESC	34453	8	VZ	
ORIGINAL GROSS AMT	34453.0	NaN	NaN	
MERCHANT NAME	34453	4716	amazon mktplce eu-uk	
CARD NUMBER	34453	993	**********5412	
BILLING CUR CODE	34453	1	GBP	
TRANS CAC CODE 1	34453	96	K020	
TRANS CAC DESC 1	34453	96	Vehicle Fuel	
TRANS CAC CODE 2	34453	816	RUF0F	
TRANS CAC DESC 2	34453	810	Illegal Money Lending T Stds Comm Inv.	
TRANS CAC CODE 3	34453	23	A00	
Directorate	34453	12	CYP&F SCHOOLS	

	freq	first	last	mean	std	\
TRANS DATE	473	2013-11-12	2015-01-01	NaN	NaN	
TRANS VAT DESC	17533	NaT	NaT	NaN	NaN	
ORIGINAL GROSS AMT	NaN	NaT	NaT	162.580459	1959.630681	
MERCHANT NAME	4113	NaT	NaT	NaN	NaN	
CARD NUMBER	325	NaT	NaT	NaN	NaN	
BILLING CUR CODE	34453	NaT	NaT	NaN	NaN	
TRANS CAC CODE 1	5396	NaT	NaT	NaN	NaN	
TRANS CAC DESC 1	5396	NaT	NaT	NaN	NaN	
TRANS CAC CODE 2	3276	NaT	NaT	NaN	NaN	
TRANS CAC DESC 2	3276	NaT	NaT	NaN	NaN	
TRANS CAC CODE 3	33971	NaT	NaT	NaN	NaN	
Directorate	15605	NaT	NaT	NaN	NaN	

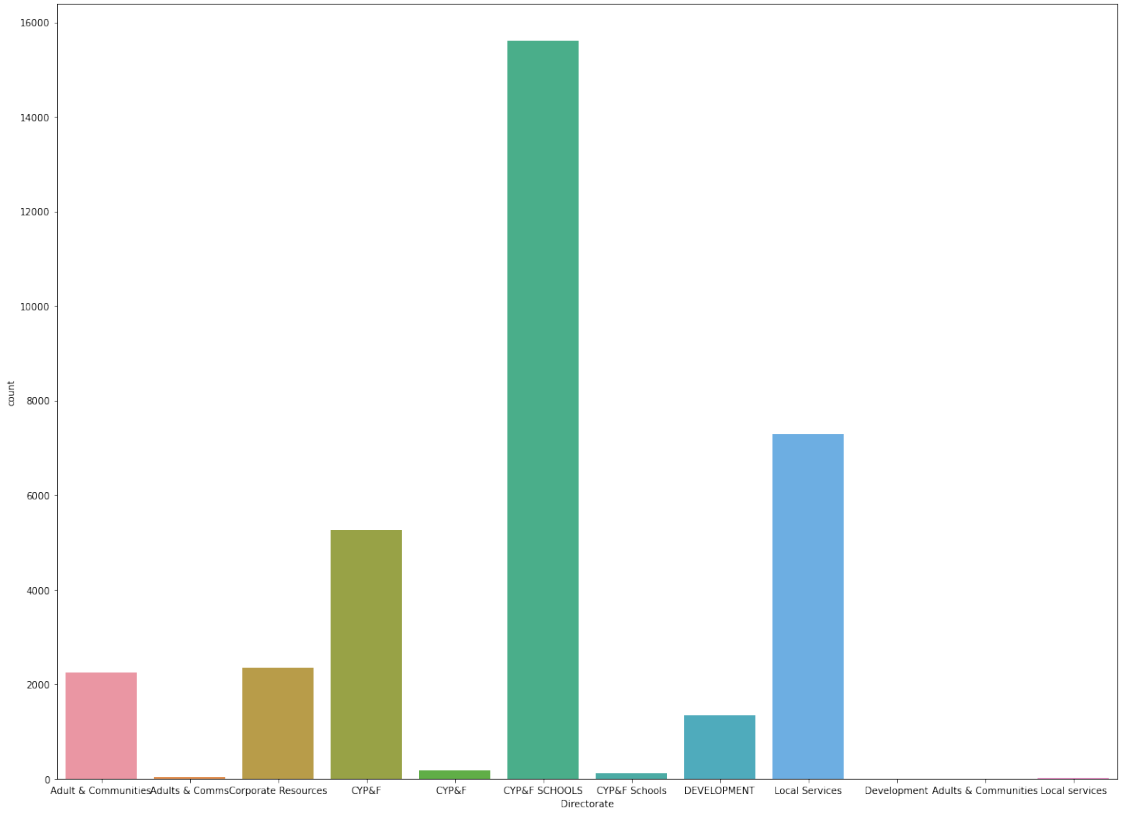
	min	25%	50%	75%	max
TRANS DATE	NaN	NaN	NaN	NaN	NaN
TRANS VAT DESC	NaN	NaN	NaN	NaN	NaN
ORIGINAL GROSS AMT	-6794.0	12.66	39.92	84.76	92379.56
MERCHANT NAME	NaN	NaN	NaN	NaN	NaN
CARD NUMBER	NaN	NaN	NaN	NaN	NaN

```
BILLING CUR CODE       NaN    NaN    NaN    NaN       NaN
TRANS CAC CODE 1       NaN    NaN    NaN    NaN       NaN
TRANS CAC DESC 1       NaN    NaN    NaN    NaN       NaN
TRANS CAC CODE 2       NaN    NaN    NaN    NaN       NaN
TRANS CAC DESC 2       NaN    NaN    NaN    NaN       NaN
TRANS CAC CODE 3       NaN    NaN    NaN    NaN       NaN
Directorate            NaN    NaN    NaN    NaN       NaN
```

We want to understand the distribution of a possible dependant categorical value if we were to do card transaction's classification, per year, related to directorates, as well as its baseline.

[14]:
```
plt.figure(figsize=(20,15))
sns.countplot(x='Directorate',data=df2014)
```

[14]:
```
<AxesSubplot:xlabel='Directorate', ylabel='count'>
```



[15]:
```
pd.crosstab(index=df2014['Directorate'],columns='count')
```

[15]:
```
col_0                 count
Directorate
Adult & Communities    2256
Adults & Comms           33
```

```
Adults & Communities      6
CYP&F                  5261
CYP&F                   183
CYP&F SCHOOLS         15605
CYP&F Schools           116
Corporate Resources    2351
DEVELOPMENT            1340
Development               2
Local Services         7289
Local services           11
```

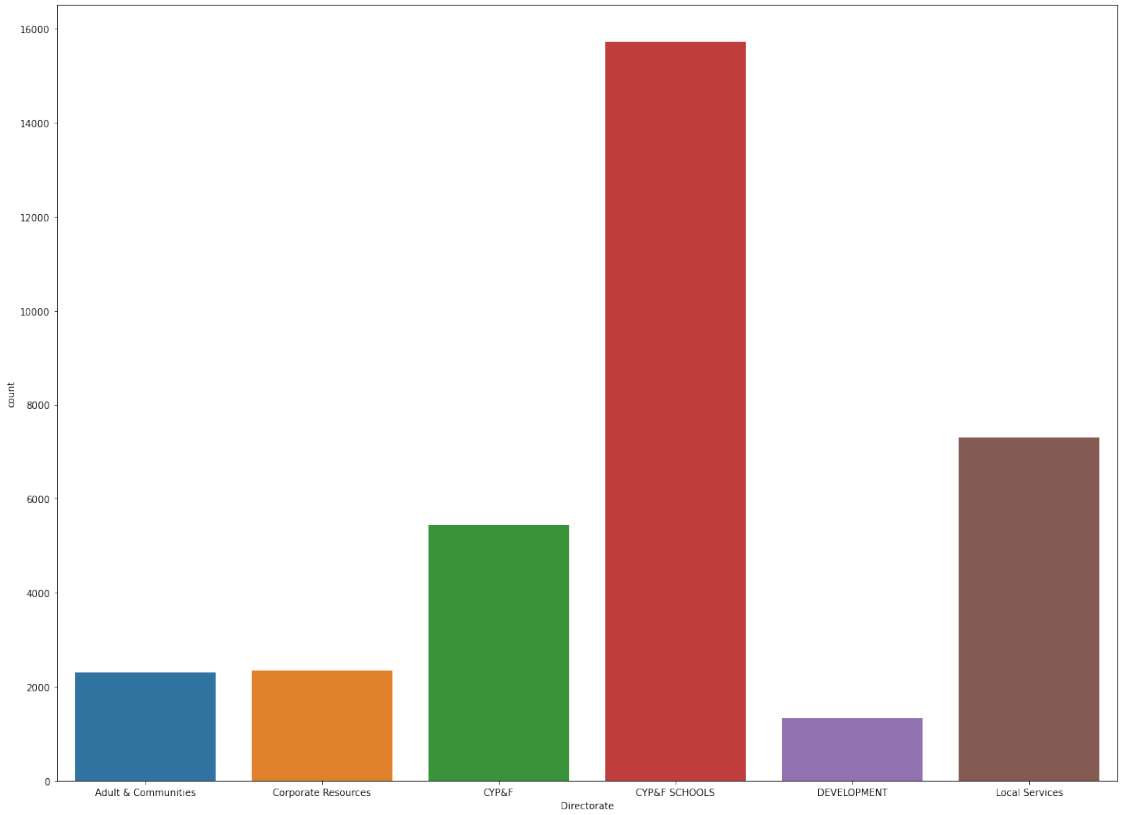
We found out that there are redundant categories that we must fusion in order to have a clearer view of the real directorates involved in the card transactions.

We check that the values are indeed related and the difference between the previous categories where typos (we checked the xls files).

[16]:
```
df2014.loc[df2014['Directorate'] == 'Adults & Comms','Directorate'] = 'Adult &␣
↪Communities'
df2014.loc[df2014['Directorate'] == 'Adults & Communities','Directorate'] =␣
↪'Adult & Communities'
df2014.loc[df2014['Directorate'] == 'CYP&F  ','Directorate'] = 'CYP&F'
df2014.loc[df2014['Directorate'] == 'CYP&F Schools','Directorate'] = 'CYP&F␣
↪SCHOOLS'
df2014.loc[df2014['Directorate'] == 'Development','Directorate'] = 'DEVELOPMENT'
df2014.loc[df2014['Directorate'] == 'Local services','Directorate'] = 'Local␣
↪Services'
```

[17]:
```
plt.figure(figsize=(20,15))
sns.countplot(x='Directorate',data=df2014)
```

[17]:
```
<AxesSubplot:xlabel='Directorate', ylabel='count'>
```



[18]:
```
pd.crosstab(index=df2014['Directorate'],columns='count')
```

[18]:
```
col_0                count
Directorate
Adult & Communities   2295
CYP&F                 5444
CYP&F SCHOOLS        15721
Corporate Resources   2351
DEVELOPMENT           1342
Local Services        7300
```

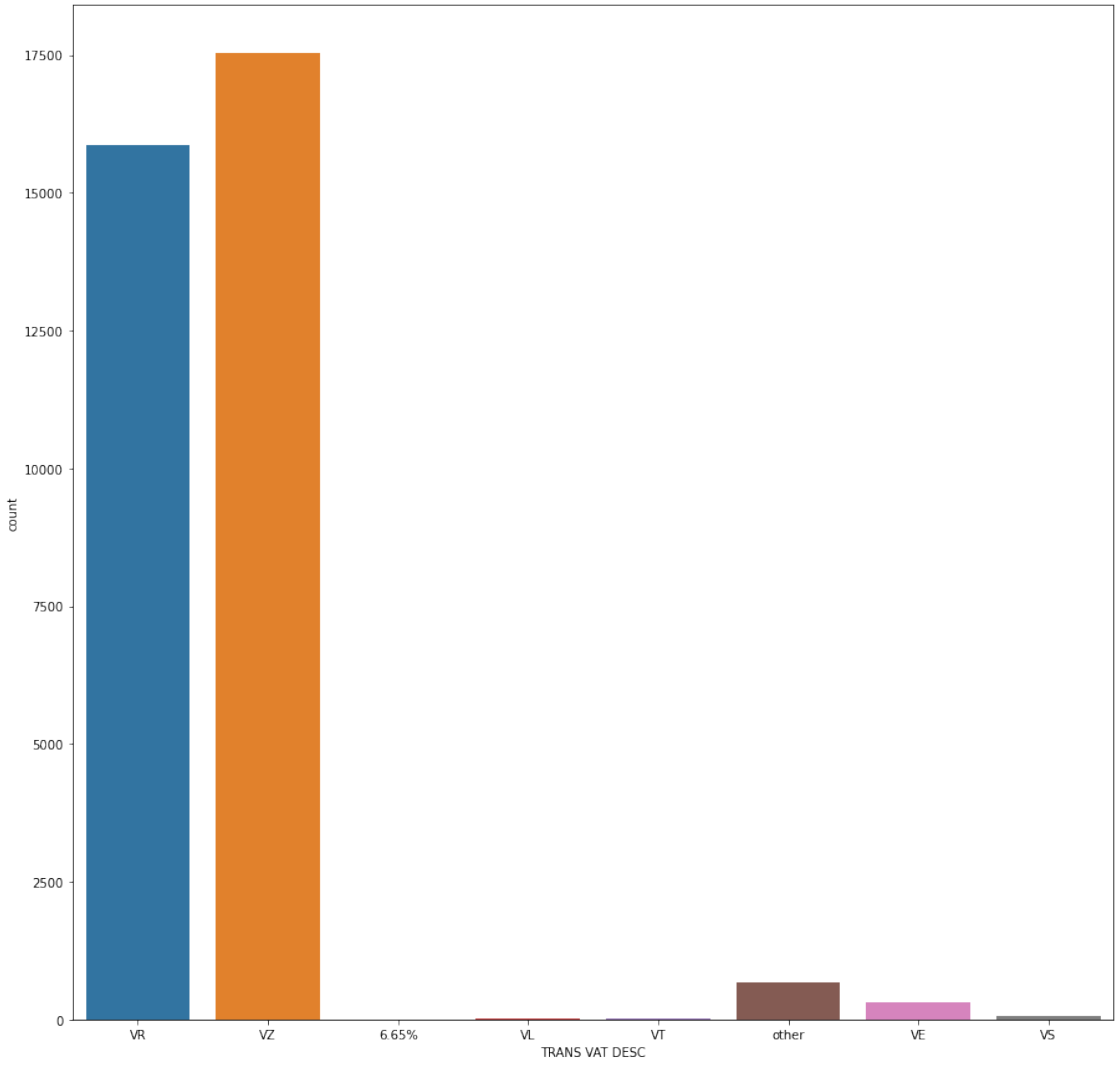
We found out a baseline of 45.63% (for directorates) for the year of 2014

[19]:
```
df2014['Directorate'][df2014['Directorate'] == 'CYP&F SCHOOLS'].count()/df2014.
↪shape[0]*100
```

[19]:
```
45.63027893071721
```

[20]:
```
plt.figure(figsize=(15,15))
sns.countplot(x='TRANS VAT DESC',data=df2014)
```

[20]:
```
<AxesSubplot:xlabel='TRANS VAT DESC', ylabel='count'>
```



We must enfasize on the importance of the meaning of each acronym mentioned in the previous bar plot.

Firstly, we would indicate that VAT is the Value-Added-Tax. In this dataset they use several categories to identify the applied tax for each transaction.

- VR: Standard rate (20%)
- VZ: Zero rate (0%)
- 6.65% rate
- VL: Lower (or reduced) rate (5%)
- VE: Exempt

This according to the document of Schools Financial Procedures Manual from the Birmingham City Council

There is no data dictionary document stating the meaning of VT and VS, and between them they sum 64 datapoints, which is just the 0.19% of the total amount of rows.

Also, since the amount of data points with VAT 6.65%, VE and VL is not significative (364, which is a 0.91%), and altogether, with the previous categories of VT and VS, it only gives us a 1.1%, we will merge them with the 'other' category.

[21]:
```
amount_of_VT_VS = df2014[df2014['TRANS VAT DESC'].isin(['VT','VS'])]['TRANS VAT␣
↪DESC'].count()
amount_of_VT_VS
```

[21]:
```
64
```

[22]:
```
amount_of_VE_VL_665 = df2014[df2014['TRANS VAT DESC'].isin(['VE','VL','6.
↪65'])]['TRANS VAT DESC'].count()
amount_of_VE_VL_665
```

[22]:
```
316
```

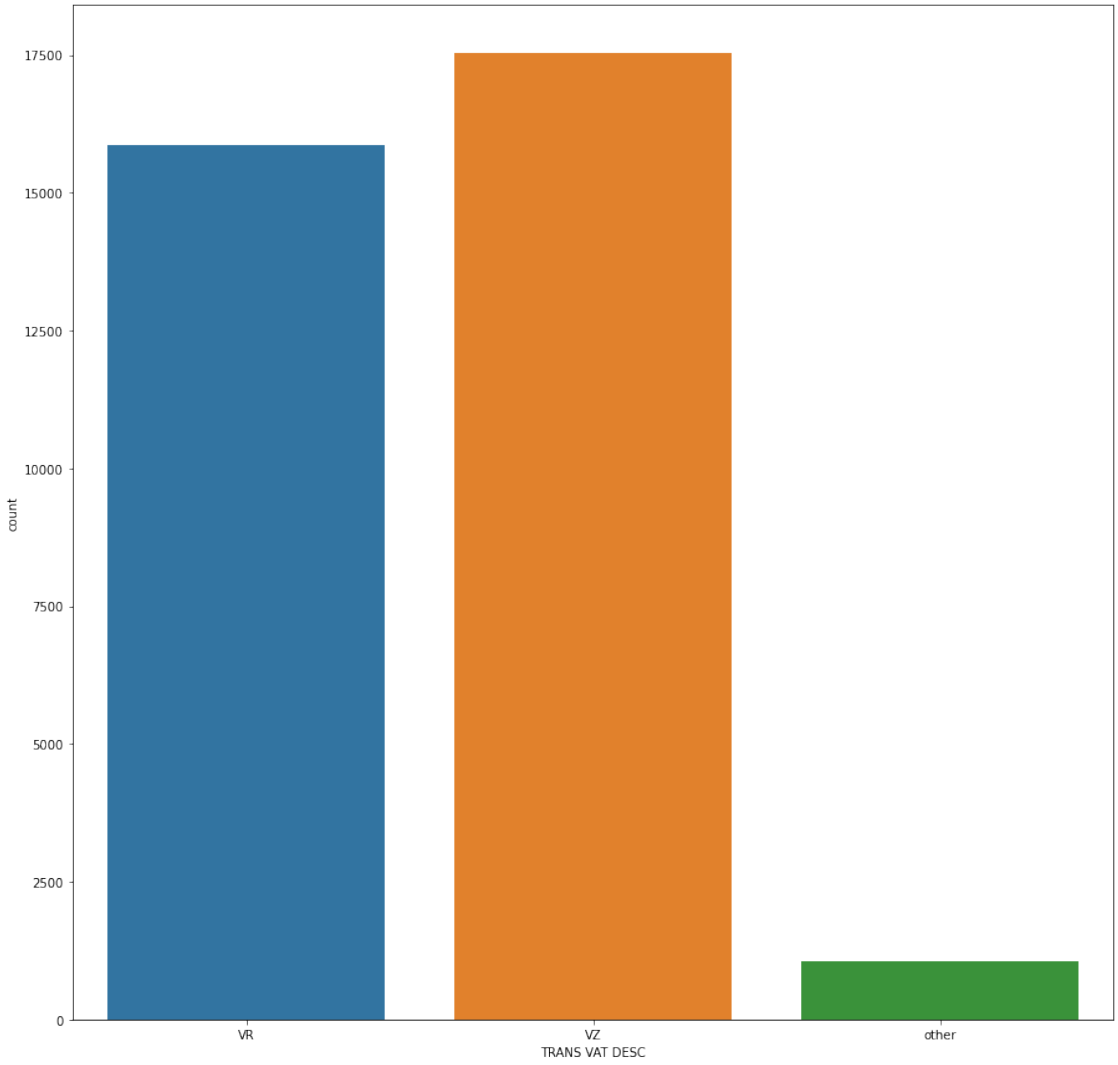
[23]:
```
df2014.loc[df2014['TRANS VAT DESC'].isin(['VT','VS','VE','VL','6.65%']),'TRANS␣
↪VAT DESC'] = 'other'
```

[24]:
```
pd.crosstab(index=df2014['TRANS VAT DESC'],columns='count')
```

[24]:
```
col_0           count
TRANS VAT DESC
VR              15858
VZ              17533
other            1062
```

[25]:
```
plt.figure(figsize=(15,15))
sns.countplot(x='TRANS VAT DESC',data=df2014)
```

[25]:
```
<AxesSubplot:xlabel='TRANS VAT DESC', ylabel='count'>
```



We found out a baseline of 50.89% (for TRANS VAT DESC) for the year of 2014

[26]:
```
df2014['TRANS VAT DESC'][df2014['TRANS VAT DESC'] == 'VZ'].count()/df2014.
↪shape[0]*100
```

[26]:
```
50.88961774010971
```

In order for us to do an exploratory analysis aiming at profiling clients and merchants we need to verify the possible typing mistakes for both categories as well as creating different dataframes grouping by each of those categories.

[27]:
```
clients = df2014['CARD NUMBER'].unique()
```

We want to check if there is a typo in the amount of asterisks and see if there are possibly repeteated card numbers (i.e. clients)

[28]:
```
num = None
differences = 0
for c in clients:
    if num == None:
        num == len(c)
    else:
        if num != len(c):
            differences += 1
print("There are", differences,"differences in length")
```

```
There are 0 differences in length
```

We want to check the amount of different merchants involved in the 2014 dataset.

[29]:
```
merchants = df2014['MERCHANT NAME'].unique()
len(merchants)
```

[29]:
```
4716
```

We define a function to verify the similarity of the names between merchants in orther to avoid repetitions. We use SequenceMatcher and its ratio.

[30]:
```
def similar(a, b):
    return SequenceMatcher(None, a, b).ratio()
```

We define a new dataframe where we will make the changes related to the merchants.

[31]:
```
df2014Merchants = df2014.copy()
```

We define a threshold of 95% and we process the column MERCHANT NAME trying to find names that are similar in more that 95% in order to reduce the amount of redundant categories. We reduce the amount of merchants to 4662.

[32]:
```
for i in range(len(merchants)):
    for j in range(i,len(merchants)):
        if similar(merchants[i],merchants[j]) != 1 and␣
↪similar(merchants[i],merchants[j]) > 0.95:
            print(merchants[i],'---',merchants[j])
            df2014Merchants.loc[df2014Merchants['MERCHANT NAME'] ==␣
↪merchants[j],'MERCHANT NAME'] = merchants[i]
```

```
amazon *mktplce eu-uk --- amazon mktplce eu-uk
w m morrison plc --- w m morrisons plc
sainsburys s/mkts --- sainsburys s/mkt
sainsburys s/mkts --- sainsburys smkts
w m morrisons --- w m morrison
w m morrisons --- wm morrisons
sainsburys smkt --- sainsburys s/mkt
sainsburys smkt --- sainsbury's smkt
```

sainsburys smkt --- sainsburys smkts
wm morrisons store --- wm morrison store
pret a manger --- pret a manager
monarch air --- monarch airl
tesco store 2624 --- tesco stores 2624
sacat marks spencer --- sacat marks & spencer
1&1 internet limited --- 11 internet limited
sainsburys s/mkt --- sainsbury's s/mkt
mcdonalds rest --- mcdonalds rest.
sainsbury's s/mkt --- sainsbury's smkt
tesco pay at pump 4203 --- tesco pay at pump 4231
morrisons petrol --- morrison petrol
the bristish associati --- the british associatio
paypal *vyka ltd --- paypal vyka ltd
pam's florist --- pams florist
paypal *pixygraphic --- paypal pixygraphic
sport leisure --- sport & leisure
ee t-mobile --- ee & t-mobile
selco trade centres --- selco trade centre
b'ham cc-payments --- bham cc-payments
mil/superstickers --- mil/superstickers-
hermolis & co ltd --- hermolis co ltd
paypal *richardbadl --- paypal richardbadl
google *adws --- google adws
salvation army trading c --- salvation army trading
holiday inn express --- holiday inn exprss
tesco store 2280 --- tesco stores 2280
m&s simply food --- ms simply food
tesco store 3145 --- tesco stores 3145
tesco store 3084 --- tesco stores 3084
welcome break starbuck --- welcome bbreak starbuc
toddington n/e costa --- toddington n/e costa k
murco petroleum ltd 12 --- murco petroleum ltd 15
welcome break waitrose --- welcome break/waitrose
holiday inns --- holiday inn
the trainline.com --- thetrainline.com
www.birmingham.gov.uk --- www.birmingham.gov.u
boots the chemist --- boots the chemists
bootsbirmingham --- boots birmingham
game retail ltd birmin --- game retail ltd birm
tesco store 3316 --- tesco stores 3316
maharaja restaurant --- maharaja restauran
the library of bh --- the library of bha
paypal irs --- paypal igrs
w m morrisons plc --- wm morrisons plc
tesco stores 2634 --- tesco store 2634
welcome break eat in --- welcome break eat in-n
welcome break subway --- welcome break subway

```
tesco store 3340 --- tesco stores 3340
paypal fancydress --- paypal fancy dress
holiday inn ipswich --- holiday inn ipswic
```

[33]:
```
len(df2014Merchants['MERCHANT NAME'].unique())
```

[33]:
```
4662
```

We save the last 2014 dataframe in a .pkl file.

[34]:
```
df2014_pkl_filename = '/Users/andresaristi/Documents/BirminghamCardTransactions/
↪pickle_files/df2014.pkl'
df2014_pkl = open(df2014_pkl_filename, 'wb')
pickle.dump(df2014Merchants, df2014_pkl)
df2014_pkl.close()
```

We load the last saved 2014 dataframe.

[35]:
```
df2014_pkl_opened = open(df2014_pkl_filename, 'rb')
df2014_pkl_loaded = pickle.load(df2014_pkl_opened)
df2014_pkl_loaded
```

[35]:

	TRANS DATE	TRANS VAT DESC	ORIGINAL GROSS AMT	MERCHANT NAME	\
0	2014-04-29	VR	52.32	shell kings 587	
1	2014-04-04	VR	65.82	shell kings 587	
2	2014-04-07	VR	41.35	tesco pfs 2484	
3	2014-04-25	VR	47.19	bush service station	
4	2014-04-24	VR	98.08	bush service station	
…	…	…	…	…	
3288	2014-12-18	VR	39.30	sainsburys s/mkts	
3289	2014-12-22	VR	349.00	dreams ltd	
3290	2014-12-22	VR	51.30	sainsburys s/mkts	
3291	2014-12-24	VZ	62.00	post office counter	
3292	2014-12-24	VZ	31.00	post office counter	

	CARD NUMBER	BILLING CUR CODE	TRANS CAC CODE 1	TRANS CAC DESC 1	\
0	**********5770	GBP	K020	Vehicle Fuel	
1	**********5770	GBP	K020	Vehicle Fuel	
2	**********8738	GBP	K020	Vehicle Fuel	
3	**********2997	GBP	K020	Vehicle Fuel	
4	**********2997	GBP	K020	Vehicle Fuel	
…	…	…	…	…	
3288	**********8673	GBP	MC70	Supplies & Sev Mic	
3289	**********8673	GBP	L100	Equip Operational	
3290	**********8673	GBP	MC70	Supplies & Sev Mic	
3291	**********8673	GBP	L700	Postage	
3292	**********8673	GBP	L700	Postage	

```
     TRANS CAC CODE 2                TRANS CAC DESC 2 TRANS CAC CODE 3  \
0               RV12N            African-Caribbean DC               A00
1               RV12N            African-Caribbean DC               A00
2               RV11Y         Marsh Lane Dce, 79, B23               A00
3               RV1K1        Elders Group - Erdington               A00
4               RV1K1        Elders Group - Erdington               A00
…                 …                               …                …
3288            RJHH8    Homeless Centre-Breedon Road               A00
3289            RJHH8    Homeless Centre-Breedon Road               A00
3290            RJHH4       Homeless Centre  Bushmere               A00
3291            RJHNB  H/Needs Hmelss Assessment Team               A00
3292            RJHNB  H/Needs Hmelss Assessment Team               A00

              Directorate
0     Adult & Communities
1     Adult & Communities
2     Adult & Communities
3     Adult & Communities
4     Adult & Communities
…                     …
3288       Local Services
3289       Local Services
3290       Local Services
3291       Local Services
3292       Local Services

[34453 rows x 12 columns]
```

We group by clients and do feature engineering by creating new numerical variables related to the different Value-Added tax category of each client.

[36]:
```
df_clients_2014 = df2014_pkl_loaded.groupby('CARD NUMBER').
↪agg(accumulated_gross_amt=('ORIGINAL GROSS AMT', 'sum'),
                                                          ␣
↪Number_of_transactions=('CARD NUMBER', 'count'),
                                                          ␣
↪VR_number=('TRANS VAT DESC', lambda val: (val == 'VR').sum()),
                                                          VZ_number=('TRANS␣
↪VAT DESC', lambda val: (val == 'VZ').sum()),
                                                          ␣
↪other_number=('TRANS VAT DESC', lambda val: (val == 'other').sum()))
df_clients_2014.head()
```

[36]:
```
                 accumulated_gross_amt  Number_of_transactions  VR_number  \
CARD NUMBER
**********0007                 214.04                       8          3
**********0015                  48.04                       1          1
```

```
**********0040                737.18                       9          8
**********0047              10855.66                      95         38
**********0057                760.81                       8          2
```

```
                  VZ_number  other_number
CARD NUMBER
**********0007          5             0
**********0015          0             0
**********0040          1             0
**********0047         51             6
**********0057          4             2
```

We create a function to do more feature engineering by creating columns related to the main description of card transactions per client.

[37]:
```
def add_trans_description(descriptions, data, variable, name):
    df = pd.DataFrame()
    i=0
    for descr in descriptions:
        df1 = pd.DataFrame()
        print(descr)
        #exec("%s = %s" % (des, descr)) (
        df1 = data.groupby(variable).agg(descr=(name, lambda val: (val ==␣
 ↪descr).sum()))
        df = pd.concat([df,df1], axis=1)
    return df
```

We create the new dataframe that will contain 96 new numerical features.

[38]:
```
descriptions = df2014Merchants['TRANS CAC DESC 1'].unique()
df_clients_2014_transf =␣
 ↪add_trans_description(descriptions,df2014_pkl_loaded,'CARD NUMBER','TRANS␣
 ↪CAC DESC 1')
```

```
Vehicle Fuel
Training Other
Computing Other
Vehicle Excise Lics
Equip Other
Vehicle R&M
Books
Staff Advert Exp
Equip Operational
Stationery
Purchases Food
Bldg RM Departmental
Travel Bus/Rail
Hospitality
```

Vehicle OthrunCosts
Ttavel Other (UK)
Photocopying
Mat'l Raw/Drct
Postage
Travel Taxis
Supplies & Sev Mic
Prof Fees other
Other Fix&Fittings
Conference Fees Subs Foreign
Conference Fees Subs UK
Legal Fee Other
Equip Office
Electricity
Car Allowances etc
Travel Foreign
N'Papers&Periodicals
Phon NonCentrx Lines
Visits Expenditure
Other Services
Personal Needs
Clothing&Uniforms
HR&M Door Entry
Equip Maintenance
Licences & Permits
Stock Misc
Cleaning Materials
Subscriptions
Grounds Maintenance
Bldg RM Fair Fund NS
Bldg RM Routine UDD
Signs & N'boards
Premises Provisions
Disinfestation
Mobiles/Radios/Pagrs
Bldg RM Emergy UDD
Bank & Goro ChgsS
Oth Indirect EmpExps
Car Parking
Advertising NonStaff
GoodsPurchforResale
Promotions/Marketing
Transport Misc
Other Agencies
Witness Expenses
In Year Credits
Consultancy Fees
Gas

```
Accomodation Hire
Purchases Othermat'l
Training EquipMat'ls
Vehicle Hire Charge
Recordings (S&V)
Family Support S17
Not'l fin Cha IT
Premises Misc
Bldg RM Fair Fund S
Catering Disposables
Security Contracts
Fire/Sec'yAlarm/Eq't
Vehicle Tyres
Equip Hire/Op Lease
Contract Meals
NonEmpAllow-General
Insurance NonPremise
Laundry
Other Third Parties
Other Grants
Transport Insurance
CSDP Act Telephones
Entertainers/Artists
Training Travel&Subs
Floral Decorations
GOVEN'MNT DEPARTMNTS
Refuse Collection
Sec. 24 CH Act 1989
IT Leasing Charges
NonEmpAllow-Training
Training Tutor Fees
Aftercare Assistance
Rents incl Svce Chgs
SchGovBrds Clerks
```

We trim the card transactions' descriptions in order to associate them to each column of the previously created dataframe.

[39]:
```
descriptions = df2014Merchants['TRANS CAC DESC 1'].unique()
descriptions_trimmed = list()
for descr in descriptions:
    des = descr
    des = descr.replace(" ","")
    des = descr.replace("&","")
    des = descr.replace("/","")
    des = descr.replace("(","")
    des = descr.replace(")","")
    des = descr.replace("'","")
```

```
    des = descr.replace("-","")
    des = descr.replace(".","")
df_clients_2014_transf.shape
len(descriptions)
df_clients_2014_transf.columns = descriptions
```

We visualize the first 5 rows of our new dataframe

[40]:
```
df_clients_2014_transf.head()
```

[40]:

CARD NUMBER	Vehicle Fuel	Training Other	Computing Other
**********0007	0	0	0
**********0015	1	0	0
**********0040	9	0	0
**********0047	0	2	1
**********0057	8	0	0

CARD NUMBER	Vehicle Excise Lics	Equip Other	Vehicle R&M	Books
**********0007	0	0	0	0
**********0015	0	0	0	0
**********0040	0	0	0	0
**********0047	2	1	0	2
**********0057	0	0	0	0

CARD NUMBER	Staff Advert Exp	Equip Operational	Stationery	…
**********0007	0	2	0	…
**********0015	0	0	0	…
**********0040	0	0	0	…
**********0047	0	32	8	…
**********0057	0	0	0	…

CARD NUMBER	Floral Decorations	GOVEN'MNT DEPARTMNTS	Refuse Collection
**********0007	0	0	0
**********0015	0	0	0
**********0040	0	0	0
**********0047	0	0	0
**********0057	0	0	0

CARD NUMBER	Sec. 24 CH Act 1989	IT Leasing Charges
**********0007	0	0
**********0015	0	0
**********0040	0	0

```
************0047                    0                    0
************0057                    0                    0

                 NonEmpAllow-Training  Training Tutor Fees  \
CARD NUMBER
************0007                     0                    0
************0015                     0                    0
************0040                     0                    0
************0047                     0                    0
************0057                     0                    0

                 Aftercare Assistance  Rents incl Svce Chgs  \
CARD NUMBER
************0007                     0                     0
************0015                     0                     0
************0040                     0                     0
************0047                     0                     0
************0057                     0                     0

                 SchGovBrds Clerks
CARD NUMBER
************0007                  0
************0015                  0
************0040                  0
************0047                  0
************0057                  0

[5 rows x 96 columns]
```

We concatenate the two dataframes in one, which we will later use to clusterize.

Previously, we will need to standardize our new numerical variables, since we are going to use unsupervised techniques based on distances.

[41]:
```
data2014 = pd.concat([df_clients_2014, df_clients_2014_transf],axis=1)
data2014.head()
```

[41]:
```
                 accumulated_gross_amt  Number_of_transactions  VR_number  \
CARD NUMBER
************0007                 214.04                       8          3
************0015                  48.04                       1          1
************0040                 737.18                       9          8
************0047                10855.66                      95         38
************0057                 760.81                       8          2

                 VZ_number  other_number  Vehicle Fuel  Training Other  \
CARD NUMBER
************0007          5             0             0               0
```

**********0015	0	0	1	0
**********0040	1	0	9	0
**********0047	51	6	0	2
**********0057	4	2	8	0

	Computing Other	Vehicle Excise Lics	Equip Other	… \
CARD NUMBER				…
**********0007	0	0	0	…
**********0015	0	0	0	…
**********0040	0	0	0	…
**********0047	1	2	1	…
**********0057	0	0	0	…

	Floral Decorations	GOVEN'MNT DEPARTMNTS	Refuse Collection \
CARD NUMBER			
**********0007	0	0	0
**********0015	0	0	0
**********0040	0	0	0
**********0047	0	0	0
**********0057	0	0	0

	Sec. 24 CH Act 1989	IT Leasing Charges \
CARD NUMBER		
**********0007	0	0
**********0015	0	0
**********0040	0	0
**********0047	0	0
**********0057	0	0

	NonEmpAllow-Training	Training Tutor Fees \
CARD NUMBER		
**********0007	0	0
**********0015	0	0
**********0040	0	0
**********0047	0	0
**********0057	0	0

	Aftercare Assistance	Rents incl Svce Chgs \
CARD NUMBER		
**********0007	0	0
**********0015	0	0
**********0040	0	0
**********0047	0	0
**********0057	0	0

	SchGovBrds Clerks
CARD NUMBER	

```
**********0007                  0
**********0015                  0
**********0040                  0
**********0047                  0
**********0057                  0

[5 rows x 101 columns]
```

We can see the difference in scales which won't be useful to us if we want use PCA to reduce dimensions and preprocess the data. We should standardize in order for the clustering methods to consume equally relevant features and be able to interpret the clusters correctly.

[42]:
```
data2014.describe()
```

[42]:
```
       accumulated_gross_amt  Number_of_transactions   VR_number   VZ_number  \
count           9.930000e+02              993.000000  993.000000  993.000000
mean            5.640871e+03               34.695871   15.969789   17.656596
std             7.110156e+04               41.762893   21.285007   31.576550
min            -6.236000e+01                1.000000    0.000000    0.000000
25%             4.653700e+02                7.000000    3.000000    1.000000
50%             1.255960e+03               19.000000    8.000000    5.000000
75%             3.279320e+03               48.000000   21.000000   21.000000
max             2.209574e+06              325.000000  165.000000  288.000000

       other_number  Vehicle Fuel  Training Other  Computing Other  \
count    993.000000    993.000000      993.000000       993.000000
mean       1.069486      5.434038        0.211480         0.472306
std        4.276507     11.459603        0.951224         2.174237
min        0.000000      0.000000        0.000000         0.000000
25%        0.000000      0.000000        0.000000         0.000000
50%        0.000000      0.000000        0.000000         0.000000
75%        0.000000      6.000000        0.000000         0.000000
max       63.000000     88.000000       15.000000        41.000000

       Vehicle Excise Lics  Equip Other  …  Floral Decorations  \
count           993.000000   993.000000  …          993.000000
mean              0.437059     0.888218  …            0.005035
std               3.830364     3.143829  …            0.095117
min               0.000000     0.000000  …            0.000000
25%               0.000000     0.000000  …            0.000000
50%               0.000000     0.000000  …            0.000000
75%               0.000000     0.000000  …            0.000000
max              99.000000    49.000000  …            2.000000

       GOVEN'MNT DEPARTMNTS  Refuse Collection  Sec. 24 CH Act 1989  \
count            993.000000         993.000000           993.000000
mean               0.003021           0.004028             0.002014
```

```
std                 0.095202            0.077667             0.063468
min                 0.000000            0.000000             0.000000
25%                 0.000000            0.000000             0.000000
50%                 0.000000            0.000000             0.000000
75%                 0.000000            0.000000             0.000000
max                 3.000000            2.000000             2.000000

       IT Leasing Charges  NonEmpAllow-Training  Training Tutor Fees  \
count          993.000000            993.000000           993.000000
mean             0.002014              0.001007             0.007049
std              0.044856              0.031734             0.145326
min              0.000000              0.000000             0.000000
25%              0.000000              0.000000             0.000000
50%              0.000000              0.000000             0.000000
75%              0.000000              0.000000             0.000000
max              1.000000              1.000000             4.000000

       Aftercare Assistance  Rents incl Svce Chgs  SchGovBrds Clerks
count             993.000000            993.000000         993.000000
mean                0.002014              0.001007           0.001007
std                 0.063468              0.031734           0.031734
min                 0.000000              0.000000           0.000000
25%                 0.000000              0.000000           0.000000
50%                 0.000000              0.000000           0.000000
75%                 0.000000              0.000000           0.000000
max                 2.000000              1.000000           1.000000

[8 rows x 101 columns]
```

We have created 101 new numerical variables by means of feature engineering.

To reduce the amount of variables we group by common elements such as:

- Vehicles
- Personal needs
- Services
- Transport
- Residence
- Work related
- Family&School
- Legal

[43]:
```
data2014_copy = data2014.copy()
data2014['Vehicles'] = data2014['Vehicle Fuel'] + data2014['Vehicle Excise␣
↪Lics'] + data2014['Vehicle R&M'] + data2014['Vehicle OthrunCosts'] +␣
↪data2014['Car Allowances etc'] + data2014['Car Parking'] + data2014['Vehicle␣
↪Hire Charge'] + data2014['Vehicle Tyres']
data2014.drop(['Vehicle Fuel'],axis=1,inplace=True)
```

```
data2014.drop(['Vehicle Excise Lics'],axis=1,inplace=True)
data2014.drop(['Vehicle R&M'],axis=1,inplace=True)
data2014.drop(['Vehicle OthrunCosts'],axis=1,inplace=True)
data2014.drop(['Car Allowances etc'],axis=1,inplace=True)
data2014.drop(['Car Parking'],axis=1,inplace=True)
data2014.drop(['Vehicle Hire Charge'],axis=1,inplace=True)
data2014.drop(['Vehicle Tyres'],axis=1,inplace=True)

data2014['Personal needs'] = data2014['Books'] + data2014['Purchases Food'] +␣
 ↪data2014['Hospitality'] + data2014['Postage'] + data2014['Personal Needs'] +␣
 ↪data2014['Clothing&Uniforms'] + data2014['Subscriptions'] +␣
 ↪data2014['Laundry'] + data2014['Entertainers/Artists'] + data2014['Mobiles/
 ↪Radios/Pagrs'] + data2014['Floral Decorations'] + data2014['Aftercare␣
 ↪Assistance'] + data2014["Mat'l Raw/Drct"] + data2014['In Year Credits'] +␣
 ↪data2014['CSDP Act Telephones']
data2014.drop(['Books'],axis=1,inplace=True)
data2014.drop(['Purchases Food'],axis=1,inplace=True)
data2014.drop(['Hospitality'],axis=1,inplace=True)
data2014.drop(['Postage'],axis=1,inplace=True)
data2014.drop(['Personal Needs'],axis=1,inplace=True)
data2014.drop(['Clothing&Uniforms'],axis=1,inplace=True)
data2014.drop(['Subscriptions'],axis=1,inplace=True)
data2014.drop(['Laundry'],axis=1,inplace=True)
data2014.drop(['Entertainers/Artists'],axis=1,inplace=True)
data2014.drop(['Mobiles/Radios/Pagrs'],axis=1,inplace=True)
data2014.drop(['Floral Decorations'],axis=1,inplace=True)
data2014.drop(['Aftercare Assistance'],axis=1,inplace=True)
data2014.drop(["Mat'l Raw/Drct"],axis=1,inplace=True)
data2014.drop(['In Year Credits'],axis=1,inplace=True)
data2014.drop(['CSDP Act Telephones'],axis=1,inplace=True)

data2014['Services'] = data2014['Photocopying'] + data2014['Gas'] +␣
 ↪data2014['Electricity'] + data2014["N'Papers&Periodicals"] + data2014['Other␣
 ↪Services'] + data2014['Disinfestation'] + data2014['Consultancy Fees'] +␣
 ↪data2014['Security Contracts'] + data2014['Phon NonCentrx Lines'] +␣
 ↪data2014['Bank & Goro ChgsS'] + data2014['Refuse Collection']
data2014.drop(['Photocopying'],axis=1,inplace=True)
data2014.drop(['Gas'],axis=1,inplace=True)
data2014.drop(['Electricity'],axis=1,inplace=True)
data2014.drop(["N'Papers&Periodicals"],axis=1,inplace=True)
data2014.drop(['Other Services'],axis=1,inplace=True)
data2014.drop(['Disinfestation'],axis=1,inplace=True)
data2014.drop(['Consultancy Fees'],axis=1,inplace=True)
data2014.drop(['Security Contracts'],axis=1,inplace=True)
data2014.drop(['Phon NonCentrx Lines'],axis=1,inplace=True)
data2014.drop(['Bank & Goro ChgsS'],axis=1,inplace=True)
data2014.drop(['Refuse Collection'],axis=1,inplace=True)
```

```
data2014['Transport'] = data2014['Travel Bus/Rail'] + data2014["Ttavel Other␣
↪(UK)"] + data2014['Travel Taxis'] + data2014['Travel Foreign'] +␣
↪data2014['Transport Misc'] + data2014['Transport Insurance']
data2014.drop(['Travel Bus/Rail'],axis=1,inplace=True)
data2014.drop(["Ttavel Other (UK)"],axis=1,inplace=True)
data2014.drop(['Travel Taxis'],axis=1,inplace=True)
data2014.drop(['Travel Foreign'],axis=1,inplace=True)
data2014.drop(['Transport Misc'],axis=1,inplace=True)
data2014.drop(['Transport Insurance'],axis=1,inplace=True)

data2014['Residence'] = data2014['Other Fix&Fittings'] + data2014['Bldg RM Fair␣
↪Fund NS'] + data2014['Bldg RM Routine UDD'] + data2014['Bldg RM Emergy UDD']␣
↪+ data2014['Bldg RM Fair Fund S'] + data2014['Rents incl Svce Chgs'] +␣
↪data2014['Bldg RM Departmental']
data2014.drop(['Other Fix&Fittings'],axis=1,inplace=True)
data2014.drop(['Bldg RM Fair Fund NS'],axis=1,inplace=True)
data2014.drop(['Bldg RM Routine UDD'],axis=1,inplace=True)
data2014.drop(['Bldg RM Emergy UDD'],axis=1,inplace=True)
data2014.drop(['Bldg RM Fair Fund S'],axis=1,inplace=True)
data2014.drop(['Rents incl Svce Chgs'],axis=1,inplace=True)
data2014.drop(['Bldg RM Departmental'],axis=1,inplace=True)

data2014['Work related'] = data2014['Training Other'] + data2014['Computing␣
↪Other'] + data2014['Equip Other'] + data2014['Staff Advert Exp'] +␣
↪data2014['Equip Operational'] + data2014['Supplies & Sev Mic'] +␣
↪data2014['Conference Fees Subs Foreign'] + data2014['Conference Fees Subs␣
↪UK'] + data2014['Equip Office'] + data2014['Equip Maintenance'] +␣
↪data2014['Licences & Permits'] + data2014['Stock Misc'] + data2014['Cleaning␣
↪Materials'] + data2014['Grounds Maintenance'] + data2014["Signs & N'boards"]␣
↪+ data2014['Premises Provisions'] + data2014['Oth Indirect EmpExps'] +␣
↪data2014['Advertising NonStaff'] + data2014['GoodsPurchforResale'] +␣
↪data2014['Promotions/Marketing'] + data2014['Other Agencies'] +␣
↪data2014['Accomodation Hire'] + data2014["Purchases Othermat'l"] +␣
↪data2014["Training EquipMat'ls"] + data2014['Catering Disposables'] +␣
↪data2014["Fire/Sec'yAlarm/Eq't"] + data2014["Equip Hire/Op Lease"] +␣
↪data2014['Contract Meals'] + data2014['NonEmpAllow-General'] +␣
↪data2014['Insurance NonPremise'] + data2014['Other Third Parties'] +␣
↪data2014['Training Travel&Subs'] + data2014['IT Leasing Charges'] +␣
↪data2014['NonEmpAllow-Training'] + data2014['Visits Expenditure'] +␣
↪data2014['Premises Misc'] + data2014["Not'l fin Cha IT"] + data2014['HR&M␣
↪Door Entry'] + data2014['Stationery'] + data2014['Recordings (S&V)']
data2014.drop(['Training Other'],axis=1,inplace=True)
data2014.drop(['Computing Other'],axis=1,inplace=True)
data2014.drop(['Equip Other'],axis=1,inplace=True)
data2014.drop(['Staff Advert Exp'],axis=1,inplace=True)
data2014.drop(['Equip Operational'],axis=1,inplace=True)
```

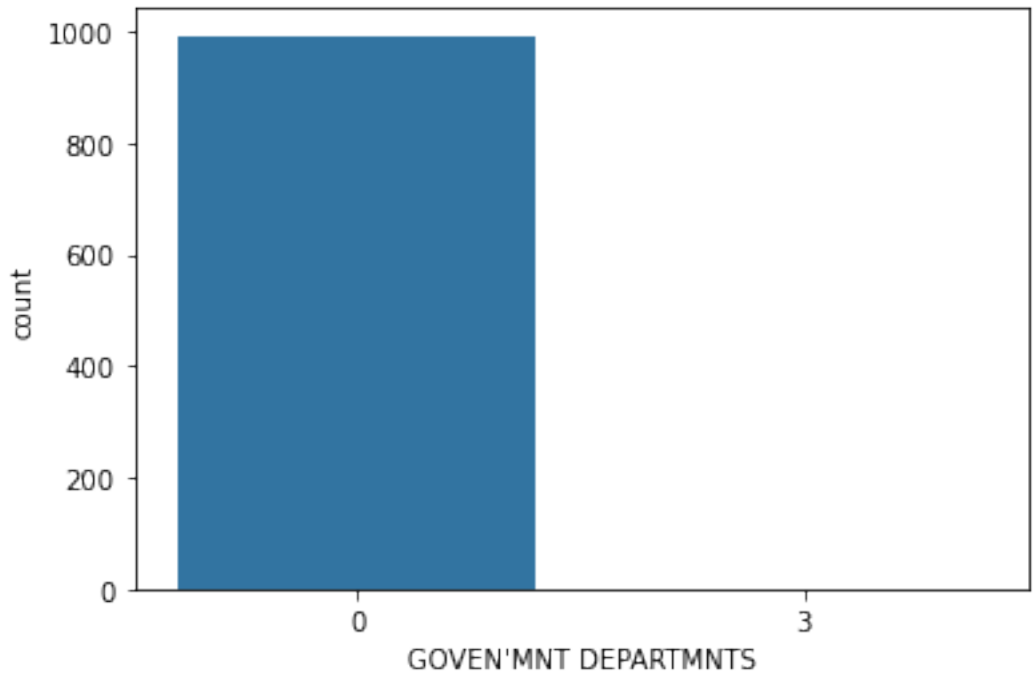
```
data2014.drop(['Supplies & Sev Mic'],axis=1,inplace=True)
data2014.drop(['Conference Fees Subs Foreign'],axis=1,inplace=True)
data2014.drop(['Conference Fees Subs UK'],axis=1,inplace=True)
data2014.drop(['Equip Office'],axis=1,inplace=True)
data2014.drop(['Equip Maintenance'],axis=1,inplace=True)
data2014.drop(['Licences & Permits'],axis=1,inplace=True)
data2014.drop(['Stock Misc'],axis=1,inplace=True)
data2014.drop(['Cleaning Materials'],axis=1,inplace=True)
data2014.drop(['Grounds Maintenance'],axis=1,inplace=True)
data2014.drop(["Signs & N'boards"],axis=1,inplace=True)
data2014.drop(['Premises Provisions'],axis=1,inplace=True)
data2014.drop(['Oth Indirect EmpExps'],axis=1,inplace=True)
data2014.drop(['Advertising NonStaff'],axis=1,inplace=True)
data2014.drop(['GoodsPurchforResale'],axis=1,inplace=True)
data2014.drop(['Promotions/Marketing'],axis=1,inplace=True)
data2014.drop(['Other Agencies'],axis=1,inplace=True)
data2014.drop(['Accomodation Hire'],axis=1,inplace=True)
data2014.drop(["Purchases Othermat'l"],axis=1,inplace=True)
data2014.drop(["Training EquipMat'ls"],axis=1,inplace=True)
data2014.drop(['Catering Disposables'],axis=1,inplace=True)
data2014.drop(["Fire/Sec'yAlarm/Eq't"],axis=1,inplace=True)
data2014.drop(["Equip Hire/Op Lease"],axis=1,inplace=True)
data2014.drop(['Contract Meals'],axis=1,inplace=True)
data2014.drop(['NonEmpAllow-General'],axis=1,inplace=True)
data2014.drop(['Insurance NonPremise'],axis=1,inplace=True)
data2014.drop(['Other Third Parties'],axis=1,inplace=True)
data2014.drop(['Training Travel&Subs'],axis=1,inplace=True)
data2014.drop(['IT Leasing Charges'],axis=1,inplace=True)
data2014.drop(['NonEmpAllow-Training'],axis=1,inplace=True)
data2014.drop(['Visits Expenditure'],axis=1,inplace=True)
data2014.drop(['Premises Misc'],axis=1,inplace=True)
data2014.drop(["Not'l fin Cha IT"],axis=1,inplace=True)
data2014.drop(['Stationery'],axis=1,inplace=True)
data2014.drop(['HR&M Door Entry'],axis=1,inplace=True)
data2014.drop(['Recordings (S&V)'],axis=1,inplace=True)

data2014["Family&School"] = data2014['Prof Fees other'] + data2014['Family␣
 ↪Support S17'] + data2014['Other Grants'] + data2014['Training Tutor Fees'] +␣
 ↪data2014['SchGovBrds Clerks'] + data2014['Sec. 24 CH Act 1989']
data2014.drop(['Prof Fees other'],axis=1,inplace=True)
data2014.drop(['Family Support S17'],axis=1,inplace=True)
data2014.drop(['Other Grants'],axis=1,inplace=True)
data2014.drop(['Training Tutor Fees'],axis=1,inplace=True)
data2014.drop(['SchGovBrds Clerks'],axis=1,inplace=True)
data2014.drop(['Sec. 24 CH Act 1989'],axis=1,inplace=True)

data2014["Legal"] = data2014['Legal Fee Other'] + data2014['Witness Expenses']
```

```
data2014.drop(['Legal Fee Other'],axis=1,inplace=True)
data2014.drop(['Witness Expenses'],axis=1,inplace=True)
```

Since 992 datapoints are from category 0 in the new numerical feature GOVEN'MNT DEPARTMNTS, (99.9% of the data) we will drop that column. It will not give us anything new to profile the clusters of clients.

[44]:
```
sns.countplot(x="GOVEN'MNT DEPARTMNTS",data=data2014)
```

[44]:
```
<AxesSubplot:xlabel="GOVEN'MNT DEPARTMNTS", ylabel='count'>
```



[45]:
```
pd.crosstab(index=data2014["GOVEN'MNT DEPARTMNTS"],columns='count')
```

[45]:
```
col_0                  count
GOVEN'MNT DEPARTMNTS
0                        992
3                          1
```

[46]:
```
data2014.drop(["GOVEN'MNT DEPARTMNTS"],axis=1,inplace=True)
```

[47]:
```
data2014.shape
```

[47]:
```
(993, 13)
```

We standardize the new numerical features

[48]:
```
data2014_std = data2014.copy()
data2014_std = StandardScaler().fit_transform(data2014_std)
data2014_std = pd.DataFrame(data2014_std)
data2014_std.columns = data2014.columns
data2014_std = data2014_std.set_index(data2014.index)
```

We have 993 data points and 13 numerical features that we will need to reduce, even more, in order to understand our clustering process.

[49]:
```
data2014_std
```

[49]:

CARD NUMBER	accumulated_gross_amt	Number_of_transactions	VR_number	\
**********0007	-0.076364	-0.639547	-0.609646	
**********0015	-0.078699	-0.807244	-0.703656	
**********0040	-0.069002	-0.615590	-0.374621	
**********0047	0.073380	1.444692	1.035532	
**********0057	-0.068670	-0.639547	-0.656651	
…	…	…	…	
**********9957	-0.018586	0.630162	1.975634	
**********9963	-0.071055	-0.735374	-0.562641	
**********9968	-0.070117	-0.567676	-0.468631	
**********9971	-0.056846	0.031243	-0.468631	
**********9989	-0.038510	1.444692	0.236446	

CARD NUMBER	VZ_number	other_number	Vehicles	Personal needs	Services	\
**********0007	-0.401025	-0.250210	-0.540526	-0.369930	0.571999	
**********0015	-0.559450	-0.250210	-0.459606	-0.491849	-0.278718	
**********0040	-0.527765	-0.250210	0.187754	-0.491849	-0.278718	
**********0047	1.056487	1.153511	-0.378686	0.605419	1.139144	
**********0057	-0.432710	0.217697	0.106834	-0.491849	-0.278718	
…	…	…	…	…	…	
**********9957	-0.464395	-0.250210	-0.055006	0.686698	-0.278718	
**********9963	-0.559450	-0.250210	-0.540526	-0.451209	-0.278718	
**********9968	-0.401025	-0.250210	0.349594	-0.491849	-0.278718	
**********9971	0.391101	-0.250210	-0.540526	0.605419	-0.278718	
**********9989	0.898062	6.300488	-0.459606	1.905884	-0.278718	

CARD NUMBER	Transport	Residence	Work related	Family&School	Legal
**********0007	-0.220214	-0.260819	-0.484121	-0.208617	-0.048902
**********0015	-0.220214	-0.260819	-0.577070	-0.208617	-0.048902
**********0040	-0.220214	-0.260819	-0.577070	-0.208617	-0.048902
**********0047	-0.220214	0.121177	2.164927	0.277666	-0.048902
**********0057	-0.220214	-0.260819	-0.577070	-0.208617	-0.048902

```
…                      …         …           …              …        …
************9957   0.221996   0.121177    0.491844     -0.208617 -0.048902
************9963  -0.220214  -0.260819   -0.437647     -0.208617 -0.048902
************9968  -0.220214  -0.260819   -0.577070     -0.208617 -0.048902
************9971   0.000891  -0.260819   -0.251749      0.277666 -0.048902
************9989  -0.220214   1.267163    0.863640     -0.208617 -0.048902

[993 rows x 13 columns]
```

1.4 4. Modelling

1.4.1 4.1 PCA

We are going to make a space projection of our principal components in order to better understand the data.

[50]:
```
pca2014 = PCA()
data2014_std_projected = pca2014.fit_transform(data2014_std)
```

We save our PCA object.

[51]:
```
df2014_pca_pkl_filename = '/Users/andresaristi/Documents/
 ↪BirminghamCardTransactions/pickle_files/df2014_pca.pkl'
df2014_pca_pkl = open(df2014_pca_pkl_filename, 'wb')
pickle.dump(data2014_std_projected, df2014_pca_pkl)
df2014_pca_pkl.close()
```

After fitting the PCA object to the standardized dataset we will be able to see 13 principal components as a result of our tranformation. We will get a linear combination of the original variables and we will see the loadings that correspond to each of them.

[52]:
```
data2014_std.columns
```

[52]:
```
Index(['accumulated_gross_amt', 'Number_of_transactions', 'VR_number',
       'VZ_number', 'other_number', 'Vehicles', 'Personal needs', 'Services',
       'Transport', 'Residence', 'Work related', 'Family&School', 'Legal'],
      dtype='object')
```

We want to see the explained variance ratio of each PC.

[53]:
```
exp_var=pca2014.explained_variance_ratio_
cum_exp_var = np.cumsum(exp_var)
exp_var
```

[53]:
```
array([2.89304643e-01, 1.10939433e-01, 8.95245750e-02, 8.18863371e-02,
       8.04666402e-02, 7.41262180e-02, 6.88510705e-02, 6.68901658e-02,
       5.85903585e-02, 4.86904913e-02, 3.07297481e-02, 3.19226661e-07,
```

```
       5.11252406e-33])
```

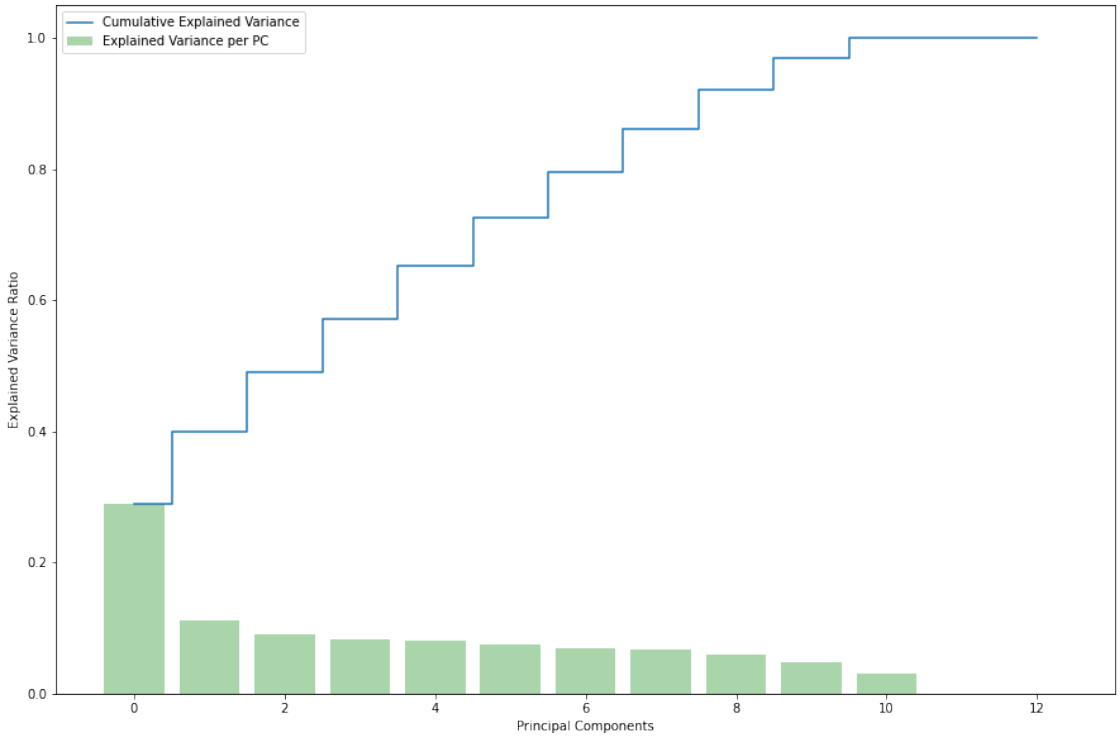
We can change the representation from the original dimensions (or features) to the new space made out by the principal componenents, by means of the transform method which belongs to the PCA object, we have already created.

[54]:
```
dataPca2014 = pca2014.transform(data2014_std)
```

To better understand the amount of information belonging to each component we are going to show a graph which will tell us the most important PCs. These new features will help us to visualize when we try to profile the clusters of clients, transactions or merchants.

We can see that this plot shows too much information which is not needed.

[55]:
```
plt.figure(figsize=(15, 10))
plt.bar(range(len(exp_var)), exp_var, alpha=0.3333, align='center',␣
 ↪label='Explained Variance per PC', color = 'g')
plt.step(range(len(cum_exp_var)), cum_exp_var, where='mid',label='Cumulative␣
 ↪Explained Variance')
plt.ylabel('Explained Variance Ratio')
plt.xlabel('Principal Components')
plt.legend(loc='best')
plt.show()
```



We focus on just the 8 first principal components. That is why we just take out the first 8 elements

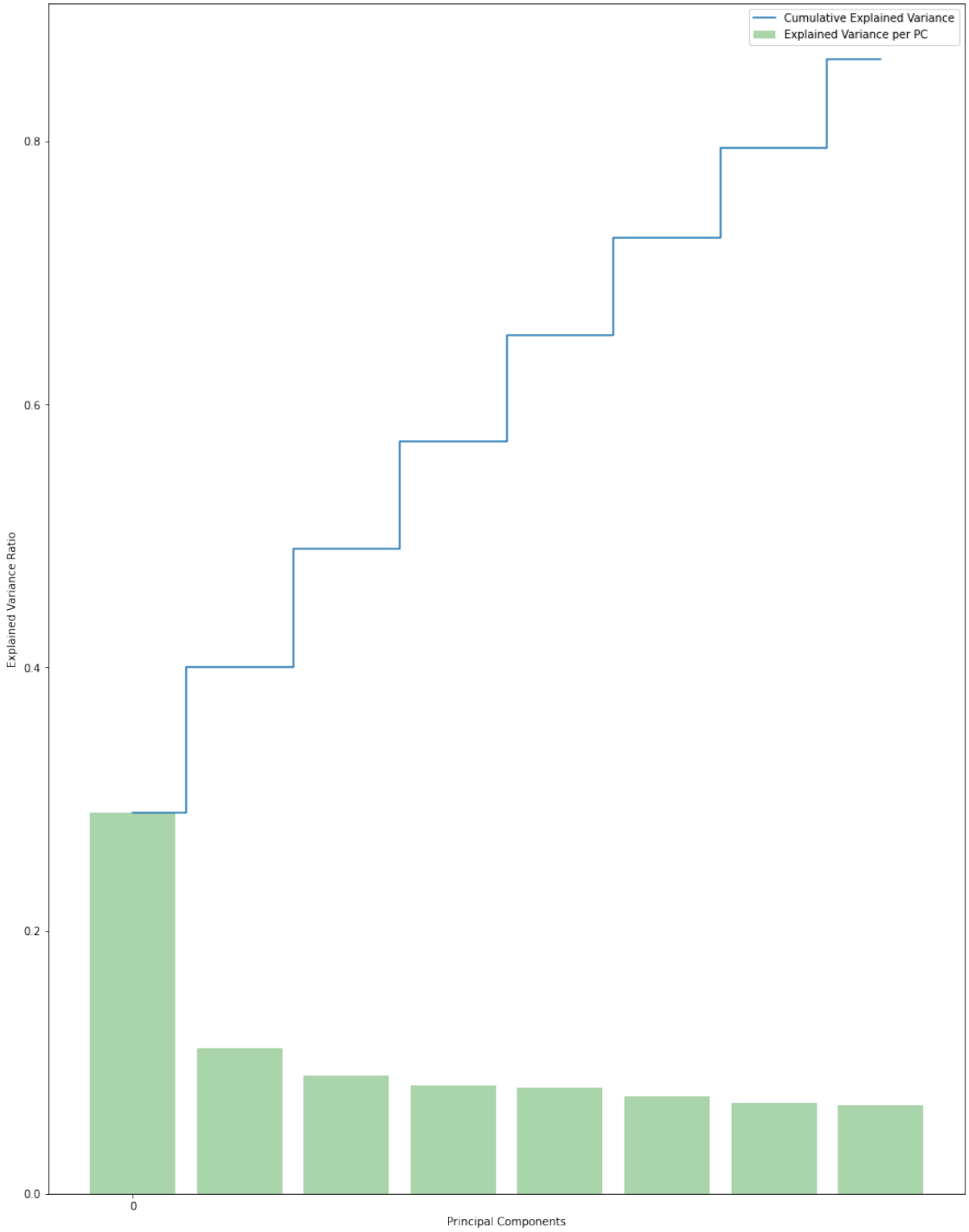
in the array of explained variance and cumulative explained variance.

[56]:
```
exp_var8 = exp_var[:8]
cum_exp_var8 = cum_exp_var[:8]
```

We focus on this bar plot and we are able to see the importance of the first principal component. We are also able to see that the next 7 PCs increase the cumulative variance much more than the following PCs (86.19%).

That is why we will focus on these components in order to analize, understand and characterize their meaning so when applying a clustering technique, we will be able to interpret each segment and give an accurate profile of the clients.

[57]:
```
plt.figure(figsize=(15, 20))
plt.bar(range(len(exp_var8)), exp_var8, alpha=0.3333, align='center',␣
↪label='Explained Variance per PC', color = 'g')
plt.step(range(len(cum_exp_var8)), cum_exp_var8, where='mid',label='Cumulative␣
↪Explained Variance')
plt.ylabel('Explained Variance Ratio')
plt.xlabel('Principal Components')
plt.xticks(np.arange(0, 5, step=5))
plt.legend(loc='best')
plt.show()
```



Some criteria say that the total variance explained by the selected principal components should be between 70% to 80% in explained variance, which in this case would mean between 6 and 8 PCS.

Therefore, we will try using 6 PCS to understand the data in order to clusterize correctly.

[58]:
```
print(cum_exp_var[5]*100,'%')
print(cum_exp_var[6]*100,'%')
print(cum_exp_var[7]*100,'%')
```

```
72.62478464753734 %
79.50989169556566 %
86.198908278766 %
```

We are able to distinguish the PC1 as the most important feature with 28.93% of the explained variance.

By sorting the PC1 we are able to understand that is a balanced linear combination in which several original features contribute.

According to our analysis the five most important original features, taking into account their loads, are, 'Number_of_transactions', 'VZ_number', 'Personal needs', 'Work related' and 'VR_number' All with more than 0.30 of loadings.

This component describes clients that make several transactions (number of transactions) for personal needs but also related to their works. VR means standard rate, which coincides with the fact that they make purchases related to personal items. VZ means zero rated items which will include books belonging to the personal needs category. All original features have a positive relation with PC1.

PC1: High number of work/personal transactions

[59]:
```
print(exp_var[0]*100,'%')
```

```
28.930464334029114 %
```

[60]:
```
pc1 = [(pca2014.components_[0][i],i) for i in range(len(pca2014.
 ↪components_[0]))]
pc1.sort(key=lambda x:abs(x[0]),reverse=True)
pc1
```

[60]:
```
[(0.5082241526523965, 1),
 (0.437618513209619, 3),
 (0.39871780870876805, 6),
 (0.39592148479729905, 10),
 (0.3073322875733205, 2),
 (0.2022347529354581, 4),
 (0.165424777559905, 7),
 (0.16524302985817668, 9),
 (0.15921979936827713, 8),
 (0.08572105913629058, 11),
 (0.07800044701529958, 12),
 (0.07042392009228982, 0),
 (0.02651877595012873, 5)]
```

[61]:
```
data2014_std.columns[[1,3,6,10,2]]
```

[61]:
```
Index(['Number_of_transactions', 'VZ_number', 'Personal needs', 'Work related',
       'VR_number'],
      dtype='object')
```

The PC2 explains the 11.09% of the variance.

By sorting the PC2 we are able to understand that is a balanced linear combination in which several original features contribute.

According to our analysis the three most important original features, taking into account their loads, are 'Vehicles', 'VR_number', 'VZ_number'. Their loadings are greater that 0.27.

This component describes clients that make card transactions related to their vehicles and, obviously, their tax-added value is VR, which is the standard one. VZ, which is zero added value tax has a negative relation, so it decreases in this group.

PC2: Vehicle and standard VAT card purchases

[62]:
```
print(exp_var[1]*100,'%')
```

```
11.09394328166093 %
```

[63]:
```
pc2 = [(pca2014.components_[1][i],i) for i in range(len(pca2014.
 ↪components_[1]))]
pc2.sort(key=lambda x:abs(x[0]),reverse=True)
pc2
```

[63]:
```
[(0.6990879235822429, 5),
 (0.5713977271792321, 2),
 (-0.2793255129376988, 3),
 (-0.18824732435531133, 11),
 (-0.15183399410393725, 6),
 (-0.13967804373663634, 12),
 (-0.09719938896620683, 7),
 (0.0735445986134687, 1),
 (0.07254759935427418, 9),
 (-0.06328366652304614, 4),
 (-0.05940598527916781, 8),
 (0.032698997213615834, 0),
 (0.005881851437448981, 10)]
```

[64]:
```
data2014_std.columns[[5,2,3]]
```

[64]:
```
Index(['Vehicles', 'VR_number', 'VZ_number'], dtype='object')
```

The amount of explained variance of PC3 is 8.95%.

When sorting the PC3, we find out the five most important original features are 'Legal', 'accumulated_gross_amt', 'other_number', 'Residence'and'Transport' which have more that 0.2 on their loadings.

This PC3 describes clients that make card purchases related mostly to legal expenses, they have a high accumulated gross amount, do not include many transctions with other TAV and do not spend much in residence (negative relation for the previous two features) but they do more transactions in transport.

PC3: High legal expenses and transport transactions

[65]:
```
print(exp_var[2]*100,'%')
```

```
8.952457500448357 %
```

[66]:
```
pc3 = [(pca2014.components_[2][i],i) for i in range(len(pca2014.
 ↪components_[2]))]
pc3.sort(key=lambda x:abs(x[0]),reverse=True)
pc3
```

[66]:
```
[(0.6022399571541859, 12),
 (0.515153104485849, 0),
 (-0.36321348098256906, 4),
 (-0.27868032195521214, 9),
 (0.23915619354672354, 8),
 (0.19924595287778463, 11),
 (-0.15841655767572563, 6),
 (0.1482149560280711, 5),
 (0.0988608623472649, 3),
 (-0.07436192195265208, 10),
 (0.04740614130512263, 1),
 (-0.021338394921974656, 7),
 (0.019328990532167304, 2)]
```

[67]:
```
data2014_std.columns[[12,0,4,9,8]]
```

[67]:
```
Index(['Legal', 'accumulated_gross_amt', 'other_number', 'Residence',
       'Transport'],
      dtype='object')
```

The PC4 explains the 8.19% of the variance.

By sorting the PC4 we are able to understand that is a balanced linear combination in which several original features contribute.

According to our analysis the six most important original features, taking into account their loads, are 'Transport', 'accumulated_gross_amt', 'Residence', 'Legal', 'other_number' and 'Family&School'. Their loadings are greater that 0.21.

This component describes clients that do not make purchases related to transport (negative relation), the accumulated value of their transactions is high. Their transactions are mainly related to residence and legal aspects. Their purchases involve other VAT category. They do not make purchases related to Family&School (negative relation)

PC4: High expenses in residence and legal aspects

[68]:
```
print(exp_var[3]*100,'%')
```

```
8.188633706200855 %
```

[69]:
```
pc4 = [(pca2014.components_[3][i],i) for i in range(len(pca2014.
 ↪components_[3]))]
pc4.sort(key=lambda x:abs(x[0]),reverse=True)
pc4
```

[69]:
```
[(-0.5856267601900639, 8),
 (0.4706277728465184, 0),
 (0.3725398807866433, 9),
 (0.2758482865928485, 12),
 (0.2357856794253863, 4),
 (-0.2143402864035548, 11),
 (-0.20684864621694832, 5),
 (0.1802883445858389, 10),
 (-0.13457778007691718, 7),
 (-0.10852299223929002, 3),
 (-0.10461703159586065, 6),
 (0.040109643615652196, 2),
 (-0.03746647839388396, 1)]
```

[70]:
```
data2014_std.columns[[8,0,9,12,4,11]]
```

[70]:
```
Index(['Transport', 'accumulated_gross_amt', 'Residence', 'Legal',
       'other_number', 'Family&School'],
      dtype='object')
```

The amount of explained variance of PC5 is 8.05%.

When sorting the PC5, we find out the four most important original features 'Legal', 'VR_number', 'accumulated_gross_amt' and 'Residence' which have more than 0.31 on their loadings.

This PC5 describes clients that do not make card purchases related to legal expenses, but most of the transactions include a standard VAT. They do not have a high accumulated gross amount and do not include many transctions concerning elements for their residence.

PC5: Non expensive transactions not related to legal expenses nor residence

[71]:
```
print(exp_var[4]*100,'%')
```

```
8.046664020469809 %
```

[72]:
```
pc5 = [(pca2014.components_[4][i],i) for i in range(len(pca2014.
 ↪components_[4]))]
pc5.sort(key=lambda x:abs(x[0]),reverse=True)
pc5
```

[72]:
```
[(-0.6065642121135595, 11),
 (0.44490701227685275, 12),
 (-0.3335588400271833, 0),
 (-0.31750876273708, 9),
 (0.22957124869365136, 6),
 (-0.20795640138837326, 10),
 (-0.20152190453673513, 8),
 (0.1744233137920079, 3),
 (0.15988266832468684, 5),
 (0.10481476554730215, 4),
 (-0.10093063251133125, 2),
 (0.09117228580267787, 1),
 (-0.005737450861367007, 7)]
```

[73]:
```
data2014_std.columns[[12,2,0,9]]
```

[73]:
```
Index(['Legal', 'VR_number', 'accumulated_gross_amt', 'Residence'],
dtype='object')
```

The amount of explained variance of PC6 is 7.41%.

When sorting the PC6, we find out the four most important original features 'Services', 'Residence', 'other_number', and 'Family&School' which have more than 0.24 on their loadings.

The PC6 describes clients that make card purchases related, mainly to services and residence, their transacions do not include other number VAT and do not make transactions related to family and shool.

PC6: Transactions related to services and residence

[74]:
```
print(exp_var[5]*100,'%')
```

```
7.412621804728281 %
```

[75]:
```
pc6 = [(pca2014.components_[5][i],i) for i in range(len(pca2014.
 ↪components_[5]))]
pc6.sort(key=lambda x:abs(x[0]),reverse=True)
pc6
```

[75]:
```
[(0.6283284681998537, 7),
 (0.5032373500287186, 9),
 (-0.4413665052478173, 4),
 (-0.24967756658381463, 11),
 (-0.16173608895780867, 0),
 (0.15589584205432222, 12),
 (-0.15226738514636756, 6),
 (-0.09339536409485272, 5),
 (0.08417436711810528, 8),
 (-0.050496292402217674, 10),
```

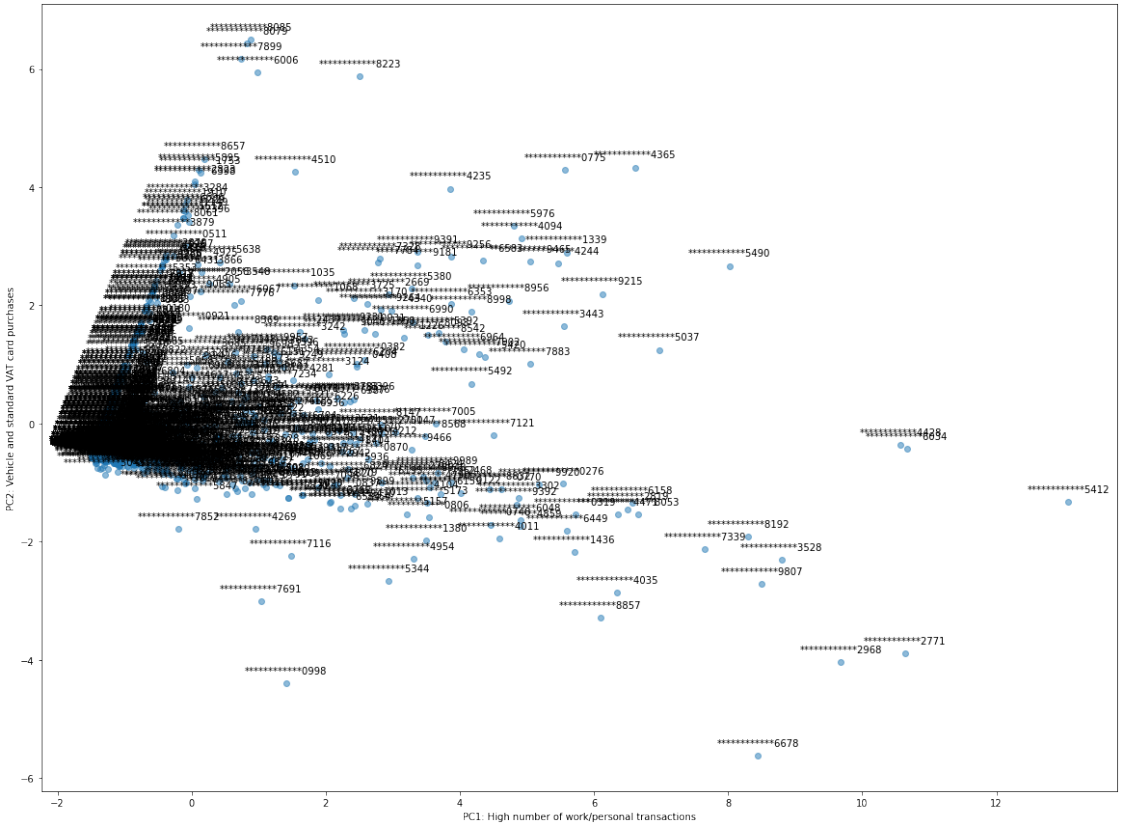
```
(0.041570833807409294, 2),
(-0.0313688539259491, 1),
(-0.009734519747798822, 3)]
```

[76]:
```
data2014_std.columns[[7,9,4,11]]
```

[76]:
```
Index(['Services', 'Residence', 'other_number', 'Family&School'],
dtype='object')
```

We visualize the projected data using the two first principal components in order to better understand the distribution of the data.

[77]:
```
plt.figure(figsize=(20, 15))
plt.scatter(data2014_std_projected[:, 0], data2014_std_projected[:, 1], alpha=0.
↪5)
plt.xlabel('PC1: High number of work/personal transactions')
plt.ylabel('PC2: Vehicle and standard VAT card purchases')
for x,y,label in zip(data2014_std_projected[:, 0], data2014_std_projected[:,␣
↪1], data2014_std.index):
    plt.annotate(label,
                 (x,y),
                 textcoords="offset points",
                 xytext=(0,10),
                 ha='center')
```



We create the new dataframe with the 6 Principal Components that we will use to clusterize our clients.

[78]:
```
data2014afterPCA = pd.DataFrame(data2014_std_projected)
data2014afterPCA = data2014afterPCA.loc[:,[0,1,2,3,4,5]]
data2014afterPCA.columns = ['High number of work/personal transactions',␣
 ↪'Vehicle and standard VAT card purchases', 'High legal expenses and␣
 ↪transport transactions', 'High expenses in residence and legal aspects',␣
 ↪'Non expensive transactions not related to legal expenses nor residence',␣
 ↪'Transactions related to services and residence']
data2014afterPCA.set_index(data2014_std.index,inplace=True)
data2014afterPCA
```

[78]:

CARD NUMBER	High number of work/personal transactions	Vehicle and standard VAT card purchases	High legal expenses and transport transactions
**********0007	-1.102613	-0.609923	-0.078921
**********0015	-1.510223	-0.474573	-0.049181
**********0040	-1.279980	0.171561	
**********0047	3.035147	-0.145967	
**********0057	-1.244730	-0.104075	
…	…	…	
**********9957	1.126939	1.249133	
**********9963	-1.360561	-0.450382	
**********9968	-1.224845	0.199069	
**********9971	0.046710	-0.869348	
**********9989	3.667547	-0.836873	

************0040	0.070341
************0047	-0.586113
************0057	-0.108621
…	…
************9957	-0.095804
************9963	-0.067909
************9968	0.106738
************9971	0.040100
************9989	-3.051555

CARD NUMBER	High expenses in residence and legal aspects \
************0007	-0.002634
************0015	0.109719
************0040	-0.017045
************0047	0.532373
************0057	0.089445
…	…
************9957	0.050369
************9963	0.153903
************9968	-0.070366
************9971	-0.278259
************9989	2.046489

CARD NUMBER	Non expensive transactions not related to legal expenses nor residence \
************0007	0.090542
************0015	0.067047
************0040	0.157104
************0047	-0.254677
************0057	0.235960
…	…
************9957	-0.173236
************9963	0.024214
************9968	0.219315
************9971	0.110072
************9989	0.870602

CARD NUMBER	Transactions related to services and residence
************0007	0.506741
************0015	-0.008816
************0040	-0.063487
************0047	-0.018432
************0057	-0.274399
…	…

```
**********9957                                        0.005066
**********9963                                       -0.012115
**********9968                                       -0.085066
**********9971                                       -0.316886
**********9989                                       -2.621232

[993 rows x 6 columns]
```

We save the resulting dataframe from our PCA

[79]:
```
df2014_after_pca_pkl_filename = '/Users/andresaristi/Documents/
 ↪BirminghamCardTransactions/pickle_files/df2014_after_pca.pkl'
df2014_after_pca_pkl = open(df2014_after_pca_pkl_filename, 'wb')
pickle.dump(data2014afterPCA, df2014_after_pca_pkl)
df2014_after_pca_pkl.close()
```

1.4.2 4.2 Clustering

By clustering the data using K-means, Hierarchical Clustering and DBSCAN we try to answer the question, which are the credit card customers' profiles for the year 2014?

K-means We are going to try first with k=3, adding a 'cluster' column with the corresponding segment.

[80]:
```
kmeans2014 = KMeans(n_clusters=3, random_state=0, n_init=10)
kmeans2014.fit(data2014afterPCA)
```

[80]:
```
KMeans(n_clusters=3, random_state=0)
```

[81]:
```
print("It took K-means", kmeans2014.n_iter_, "iterations to converge, with a␣
 ↪final WSS (Within Sum of Squares) of:",
      kmeans2014.inertia_, "and the following centroids:", kmeans2014.
 ↪cluster_centers_)
```

```
It took K-means 9 iterations to converge, with a final WSS (Within Sum of
Squares) of: 5867.0656223600945 and the following centroids: [[-6.62577522e-01
1.48022839e-03  1.51781172e-02 -1.97696484e-02
   1.39822263e-02 -1.40941235e-02]
 [ 3.81191507e+00  2.72299933e-02 -3.99908370e-01 -7.79427913e-02
  -1.37313764e-01  7.77915256e-02]
 [ 5.36643524e+00 -1.71588083e+00  1.47770105e+01  9.29690379e+00
   2.59764087e+00  2.71178143e-01]]
```

We can see the amount of observations per cluster, with unbalanced clusters, with most data points in cluster 0

[82]:
```
counter = Counter(kmeans2014.labels_)
counter
```

[82]:
```
Counter({0: 847, 1: 143, 2: 3})
```

We add a column to represent the cluster to which each row belongs.

[83]:
```
data2014_kmeans = data2014afterPCA.copy()
data2014_kmeans['Cluster'] = kmeans2014.labels_
data2014_kmeans
```

[83]:

CARD NUMBER	High number of work/personal transactions	Vehicle and standard VAT card purchases	High legal expenses and transport transactions
************0007	-1.102613	-0.609923	-0.078921
************0015	-1.510223	-0.474573	-0.049181
************0040	-1.279980	0.171561	0.070341
************0047	3.035147	-0.145967	-0.586113
************0057	-1.244730	-0.104075	-0.108621
…	…	…	…
************9957	1.126939	1.249133	
************9963	-1.360561	-0.450382	
************9968	-1.224845	0.199069	
************9971	0.046710	-0.869348	
************9989	3.667547	-0.836873	

```
************9957                                    -0.095804
************9963                                    -0.067909
************9968                                     0.106738
************9971                                     0.040100
************9989                                    -3.051555

                 High expenses in residence and legal aspects  \
CARD NUMBER
************0007                                   -0.002634
************0015                                    0.109719
************0040                                   -0.017045
************0047                                    0.532373
************0057                                    0.089445
…                                                      …
************9957                                    0.050369
************9963                                    0.153903
************9968                                   -0.070366
************9971                                   -0.278259
************9989                                    2.046489

                 Non expensive transactions not related to legal expenses nor
residence  \
CARD NUMBER
************0007                                     0.090542
************0015                                     0.067047
************0040                                     0.157104
************0047                                    -0.254677
************0057                                     0.235960
…                                                        …
************9957                                    -0.173236
************9963                                     0.024214
************9968                                     0.219315
************9971                                     0.110072
************9989                                     0.870602

                 Transactions related to services and residence  Cluster
CARD NUMBER
************0007                                     0.506741        0
************0015                                    -0.008816        0
************0040                                    -0.063487        0
************0047                                    -0.018432        1
************0057                                    -0.274399        0
…                                                        …       …
************9957                                     0.005066        0
************9963                                    -0.012115        0
************9968                                    -0.085066        0
************9971                                    -0.316886        0
```

```
**********9989                                    -2.621232        1

[993 rows x 7 columns]
```

We save the resulting dataframe from our k-means with k = 3.

[84]:
```
df2014_kmeans3_pkl_filename = '/Users/andresaristi/Documents/
 ↪BirminghamCardTransactions/pickle_files/df2014_kmeans3.pkl'
df2014_kmeans3_pkl = open(df2014_kmeans3_pkl_filename, 'wb')
pickle.dump(data2014_kmeans, df2014_kmeans3_pkl)
df2014_kmeans3_pkl.close()
```

Interpretation of k-means using k=3 We, by using the density and scatter plots, are going to understand the different characteristics of each cluster according to the observations which belong to each of them.

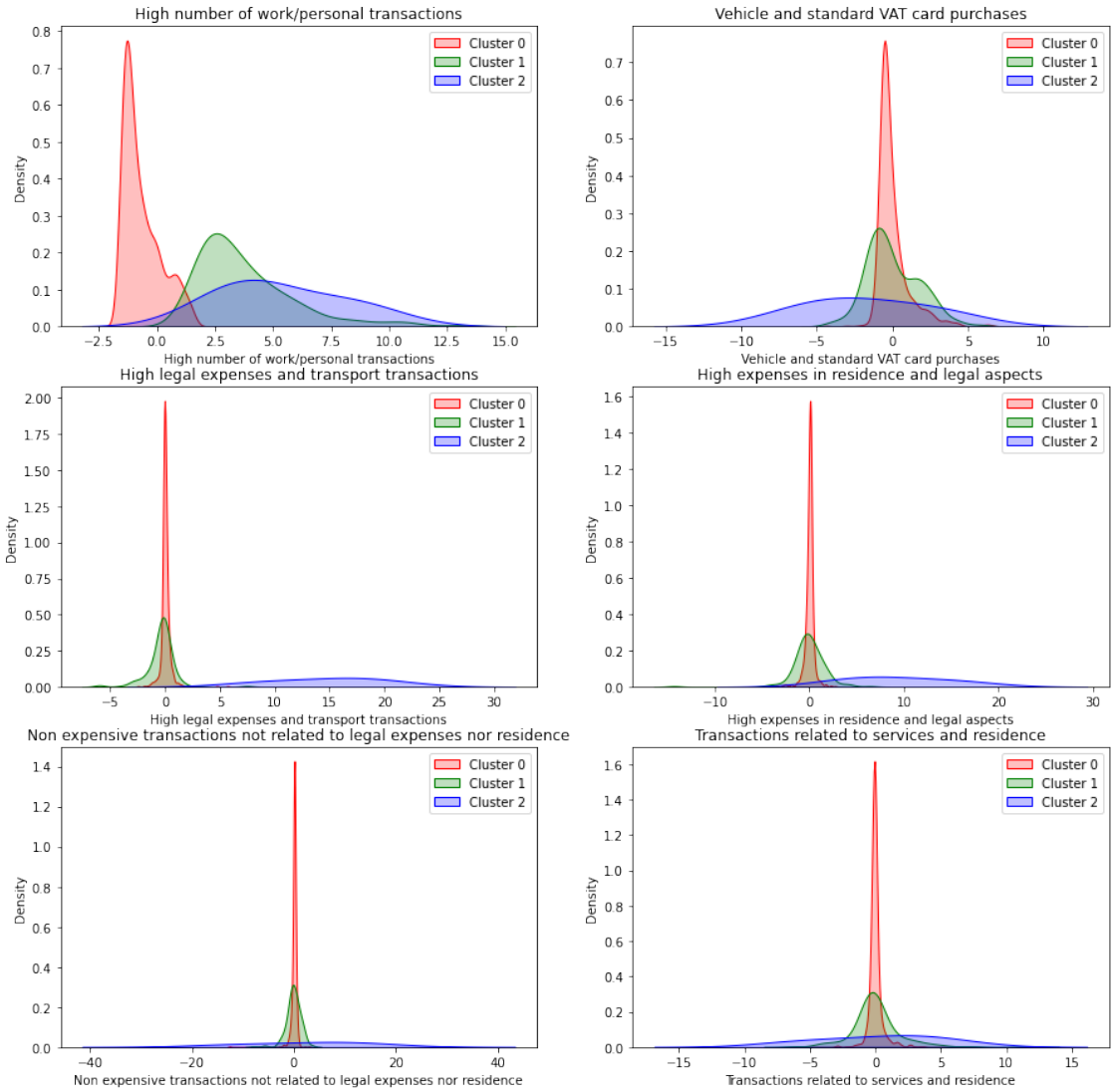
By observing the plots we are able to understand the type and amount of card transactions' purchases of each group of clients.

Cluster 0: the most densely populated. 847 data points. The clients belonging to this segment do not make high number of card purchases related to work, personal needs, legal expenses nor affairs related to their residences. They have average transactions that are non expensive and not related to legal expenses nor residences. They have average transactions on services.

Cluster 1: 143 data points. These clients have less than average on high number of work and personal transactions. The purchases related to these clients increase when talking about vehicles and standard items (which have the added value tax VR). They do not expend much in legal aspects and transport. They have less than average transactions on services. On average make non expensive transactions. These clients make transactions, on average, on residence matters.

Cluster 2: the least densely populated cluster with only 3 data points. Since there are so few clients the distribution is broader over all the new features.

[85]:
```
var_num = data2014_kmeans.columns[0:6]
fig = plt.figure(figsize=(15,15))
i=1
for var in var_num:
    ax = fig.add_subplot(math.ceil(len(var_num)/2), 2, i)
    sns.kdeplot(data2014_kmeans.loc[data2014_kmeans.Cluster==0][var],␣
 ↪shade=True, color='r', ax=ax);
    sns.kdeplot(data2014_kmeans.loc[data2014_kmeans.Cluster==1][var],␣
 ↪shade=True, color='g', ax=ax);
    sns.kdeplot(data2014_kmeans.loc[data2014_kmeans.Cluster==2][var],␣
 ↪shade=True, color='b', ax=ax);
    plt.title(var)
    plt.legend(['Cluster 0', 'Cluster 1', 'Cluster 2'])
    i+=1
```

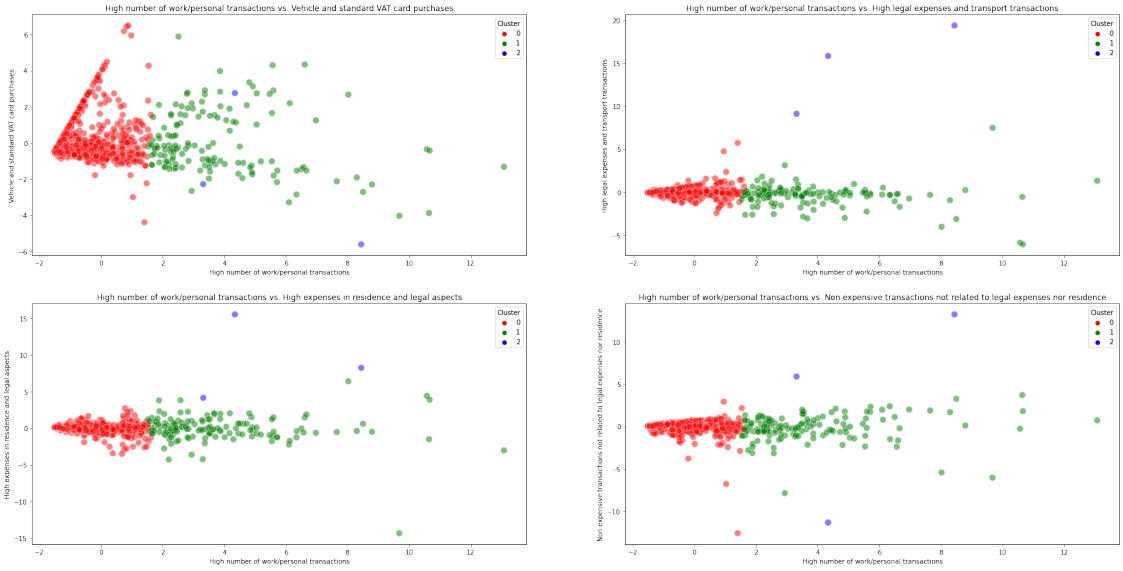


We will see the scatterplots to better understand the differences

The other elements observed in the scatter plots confirm our description of the two other clusters.

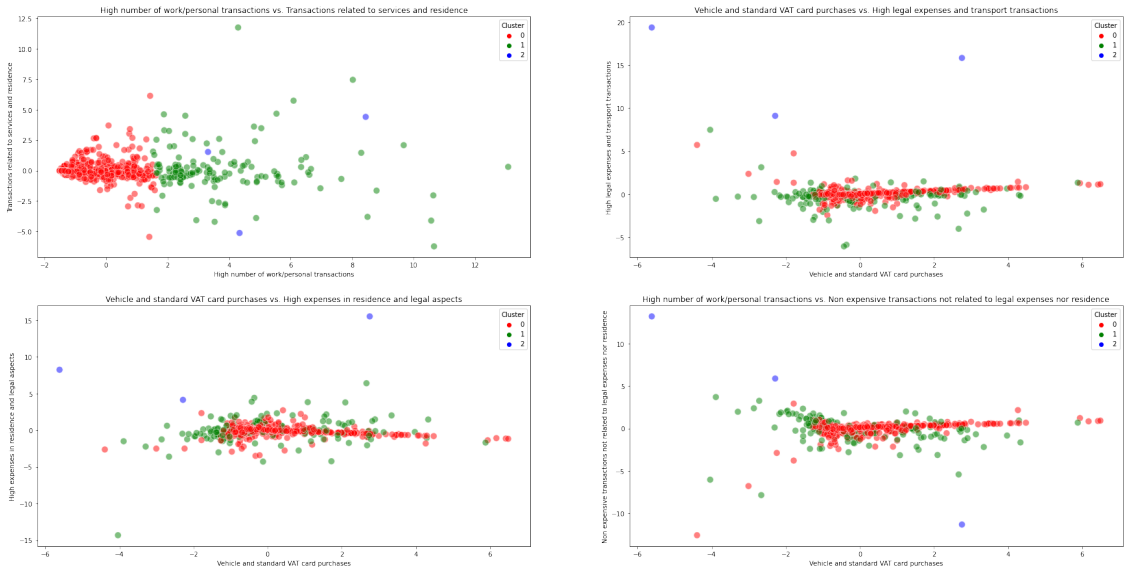
[86]:
```
fig = plt.figure(figsize=(30,15))
colorPalette = ["r", "g", "b"]
ax = fig.add_subplot(2, 2, 1)
sns.scatterplot(x='High number of work/personal transactions', y='Vehicle and␣
 ↪standard VAT card purchases', hue="Cluster", data=data2014_kmeans, ax=ax,␣
 ↪palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. Vehicle and standard␣
 ↪VAT card purchases")
ax = fig.add_subplot(2, 2, 2)
```

```
sns.scatterplot(x="High number of work/personal transactions", y="High legal␣
 ↪expenses and transport transactions", hue="Cluster", data=data2014_kmeans,␣
 ↪ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. High legal expenses␣
 ↪and transport transactions")
ax = fig.add_subplot(2, 2, 3)
sns.scatterplot(x="High number of work/personal transactions", y="High expenses␣
 ↪in residence and legal aspects", hue="Cluster", data=data2014_kmeans, ax=ax,␣
 ↪palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. High expenses in␣
 ↪residence and legal aspects")
ax = fig.add_subplot(2, 2, 4)
sns.scatterplot(x='High number of work/personal transactions', y='Non expensive␣
 ↪transactions not related to legal expenses nor residence', hue="Cluster",␣
 ↪data=data2014_kmeans, ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. Non expensive␣
 ↪transactions not related to legal expenses nor residence")
plt.show()
```



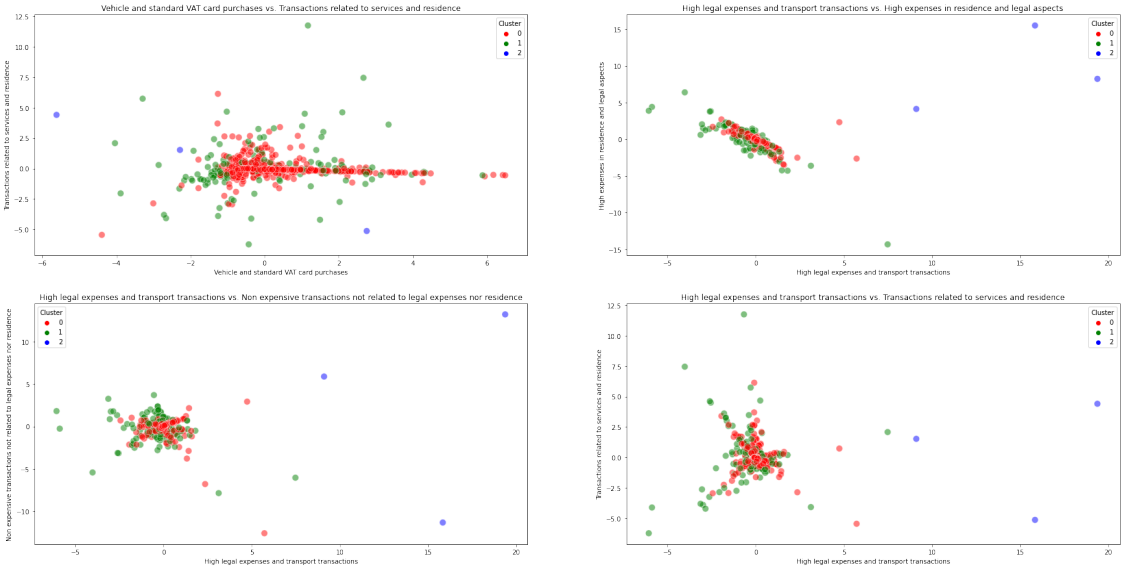
[87]:
```
fig = plt.figure(figsize=(30,15))
colorPalette = ["r", "g", "b"]
ax = fig.add_subplot(2, 2, 1)
sns.scatterplot(x="High number of work/personal transactions", y="Transactions␣
 ↪related to services and residence", hue="Cluster", data=data2014_kmeans,␣
 ↪ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. Transactions related␣
 ↪to services and residence")
```

```
ax = fig.add_subplot(2, 2, 2)
sns.scatterplot(x="Vehicle and standard VAT card purchases", y="High legal␣
 ↪expenses and transport transactions", hue="Cluster", data=data2014_kmeans,␣
 ↪ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("Vehicle and standard VAT card purchases vs. High legal expenses and␣
 ↪transport transactions")
ax = fig.add_subplot(2, 2, 3)
sns.scatterplot(x='Vehicle and standard VAT card purchases', y='High expenses␣
 ↪in residence and legal aspects', hue="Cluster", data=data2014_kmeans, ax=ax,␣
 ↪palette=colorPalette, s=100, alpha=0.5)
plt.title("Vehicle and standard VAT card purchases vs. High expenses in␣
 ↪residence and legal aspects")
ax = fig.add_subplot(2, 2, 4)
sns.scatterplot(x="Vehicle and standard VAT card purchases", y="Non expensive␣
 ↪transactions not related to legal expenses nor residence", hue="Cluster",␣
 ↪data=data2014_kmeans, ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. Non expensive␣
 ↪transactions not related to legal expenses nor residence")
plt.show()
```



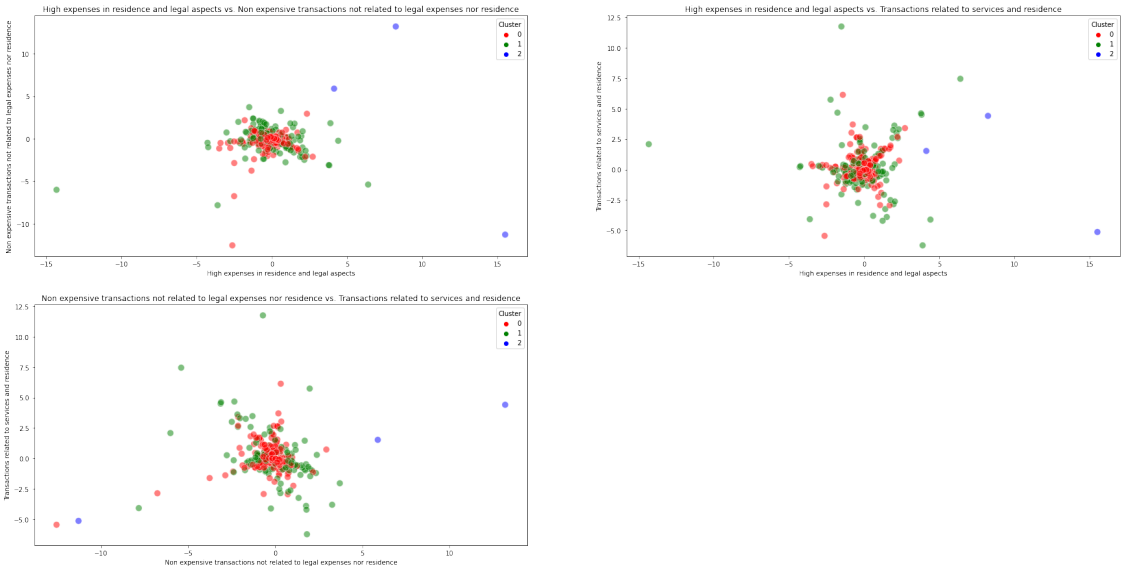
[88]:
```
fig = plt.figure(figsize=(30,15))
colorPalette = ["r", "g", "b"]
ax = fig.add_subplot(2, 2, 1)
sns.scatterplot(x="Vehicle and standard VAT card purchases", y="Transactions␣
 ↪related to services and residence", hue="Cluster", data=data2014_kmeans,␣
 ↪ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("Vehicle and standard VAT card purchases vs. Transactions related to␣
 ↪services and residence")
```

```
ax = fig.add_subplot(2, 2, 2)
sns.scatterplot(x='High legal expenses and transport transactions', y='High␣
 ↪expenses in residence and legal aspects', hue="Cluster",␣
 ↪data=data2014_kmeans, ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High legal expenses and transport transactions vs. High expenses in␣
 ↪residence and legal aspects")
ax = fig.add_subplot(2, 2, 3)
sns.scatterplot(x="High legal expenses and transport transactions", y="Non␣
 ↪expensive transactions not related to legal expenses nor residence",␣
 ↪hue="Cluster", data=data2014_kmeans, ax=ax, palette=colorPalette, s=100,␣
 ↪alpha=0.5)
plt.title("High legal expenses and transport transactions vs. Non expensive␣
 ↪transactions not related to legal expenses nor residence")
ax = fig.add_subplot(2, 2, 4)
sns.scatterplot(x="High legal expenses and transport transactions",␣
 ↪y="Transactions related to services and residence", hue="Cluster",␣
 ↪data=data2014_kmeans, ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High legal expenses and transport transactions vs. Transactions␣
 ↪related to services and residence")
plt.show()
```



[89]:
```
fig = plt.figure(figsize=(30,15))
colorPalette = ["r", "g", "b"]
ax = fig.add_subplot(2, 2, 1)
sns.scatterplot(x="High expenses in residence and legal aspects", y="Non␣
 ↪expensive transactions not related to legal expenses nor residence",␣
 ↪hue="Cluster", data=data2014_kmeans, ax=ax, palette=colorPalette, s=100,␣
 ↪alpha=0.5)
```

```
plt.title("High expenses in residence and legal aspects vs. Non expensive␣
 ↪transactions not related to legal expenses nor residence")
ax = fig.add_subplot(2, 2, 2)
sns.scatterplot(x="High expenses in residence and legal aspects",␣
 ↪y="Transactions related to services and residence", hue="Cluster",␣
 ↪data=data2014_kmeans, ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High expenses in residence and legal aspects vs. Transactions␣
 ↪related to services and residence")
ax = fig.add_subplot(2, 2, 3)
sns.scatterplot(x="Non expensive transactions not related to legal expenses nor␣
 ↪residence", y="Transactions related to services and residence",␣
 ↪hue="Cluster", data=data2014_kmeans, ax=ax, palette=colorPalette, s=100,␣
 ↪alpha=0.5)
plt.title("Non expensive transactions not related to legal expenses nor␣
 ↪residence vs. Transactions related to services and residence")
plt.show()
```



Determining the right k

Elbow method By using the Within Sum of Squares (WSS) values we will create a plot by which we will apply the elbow method, and cluster using different k values.

[90]:
```
WSSs = []
for i in range(1,15) :
    km = KMeans(n_clusters=i, random_state=0)
    km.fit(data2014_kmeans)
```

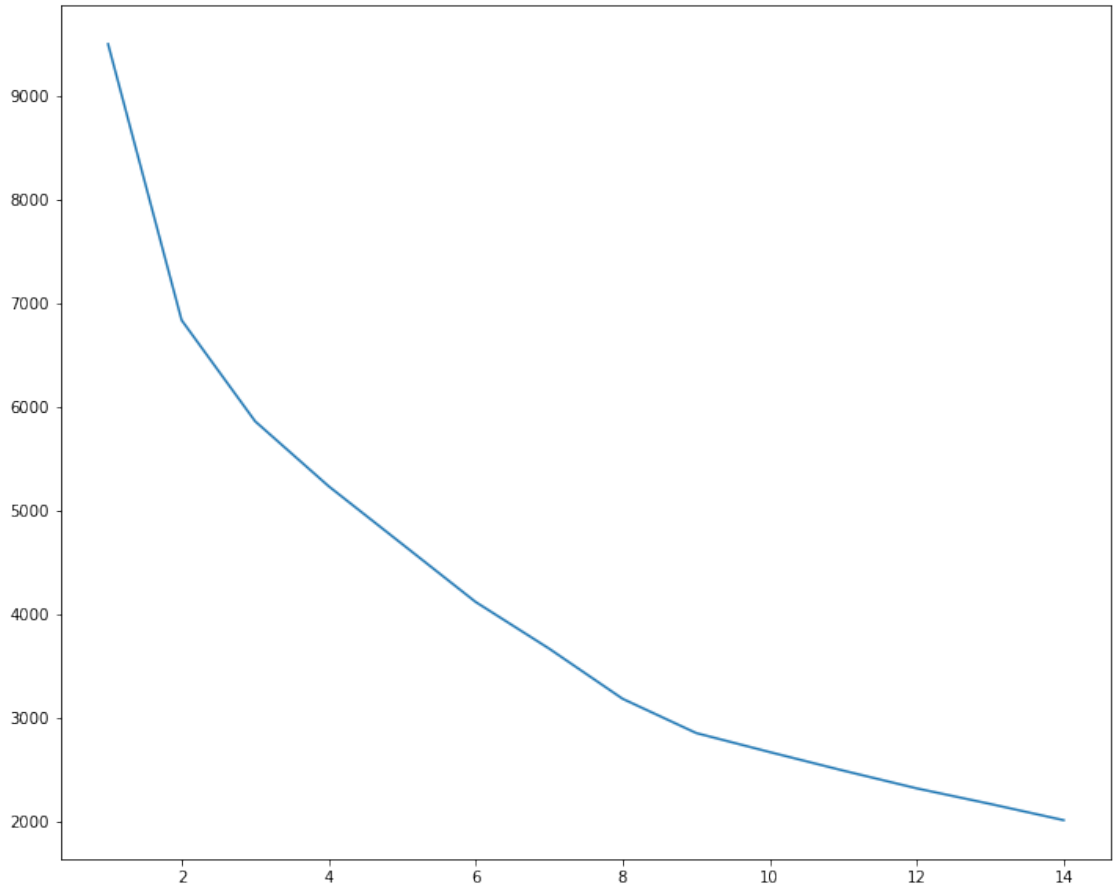
```
    WSSs.append(km.inertia_)
WSSs
```

[90]:
```
[9507.775947632981,
 6843.184481678275,
 5867.0656223600945,
 5240.250303442649,
 4682.01763222187,
 4122.937874083134,
 3672.4379914210904,
 3187.8810816845394,
 2857.6306860727555,
 2673.814534089913,
 2495.1018373154566,
 2323.469936835077,
 2173.192451369397,
 2016.1816118789964]
```

We found out that with this technique the ideal k seems to be 2, 3, 8 or 9.

[91]:
```
plt.figure(figsize=(12,10))
plt.plot(range(1, 15), WSSs)
```

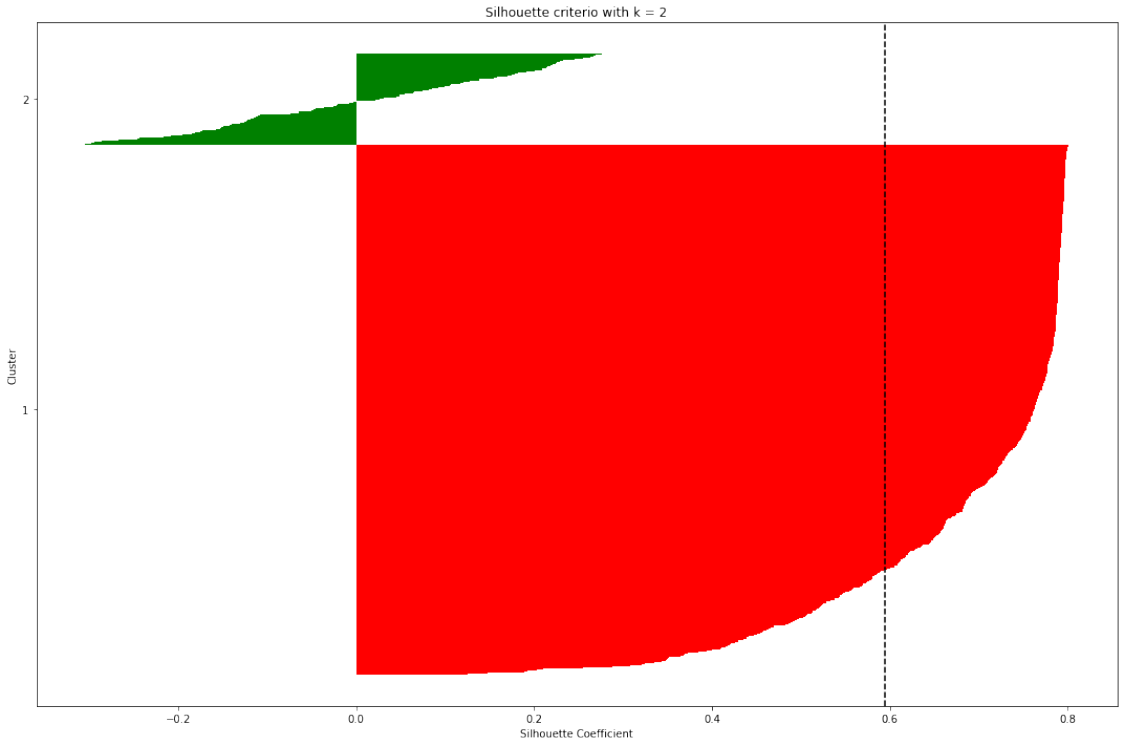
[91]:
```
[<matplotlib.lines.Line2D at 0x7f9ce2ae0be0>]
```

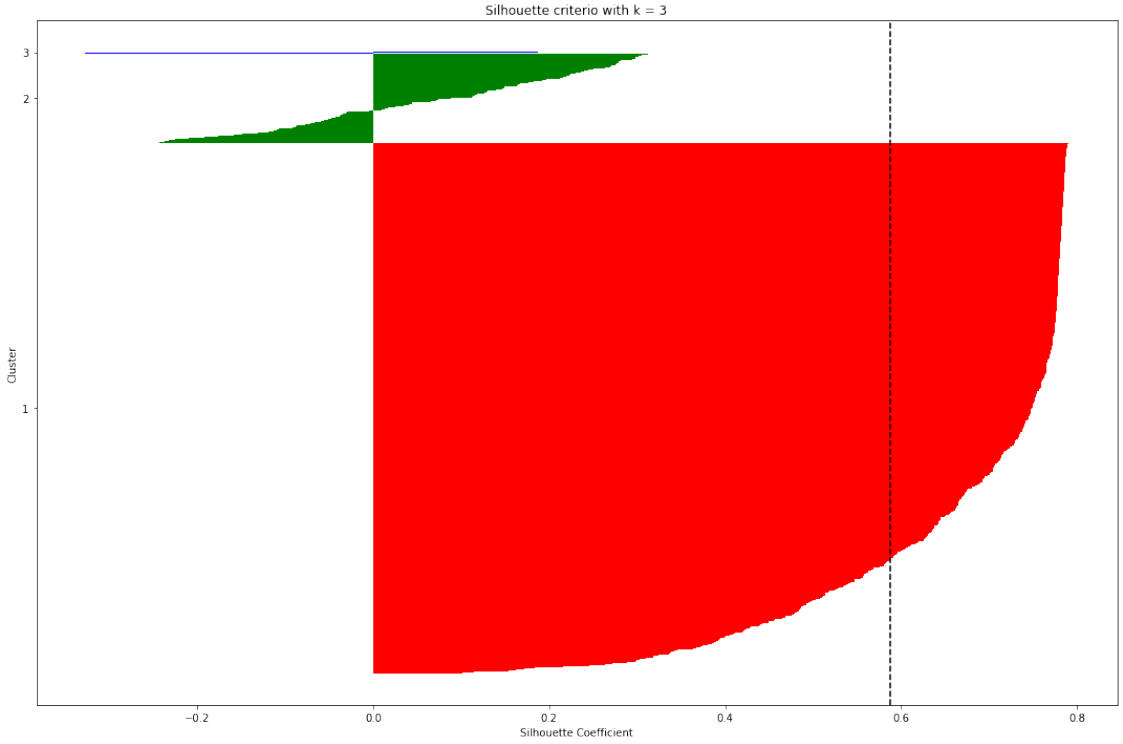


Silhouette method By using the silhoutte method we will try to find a different approach to the number of clusters that we will need for our dataset.

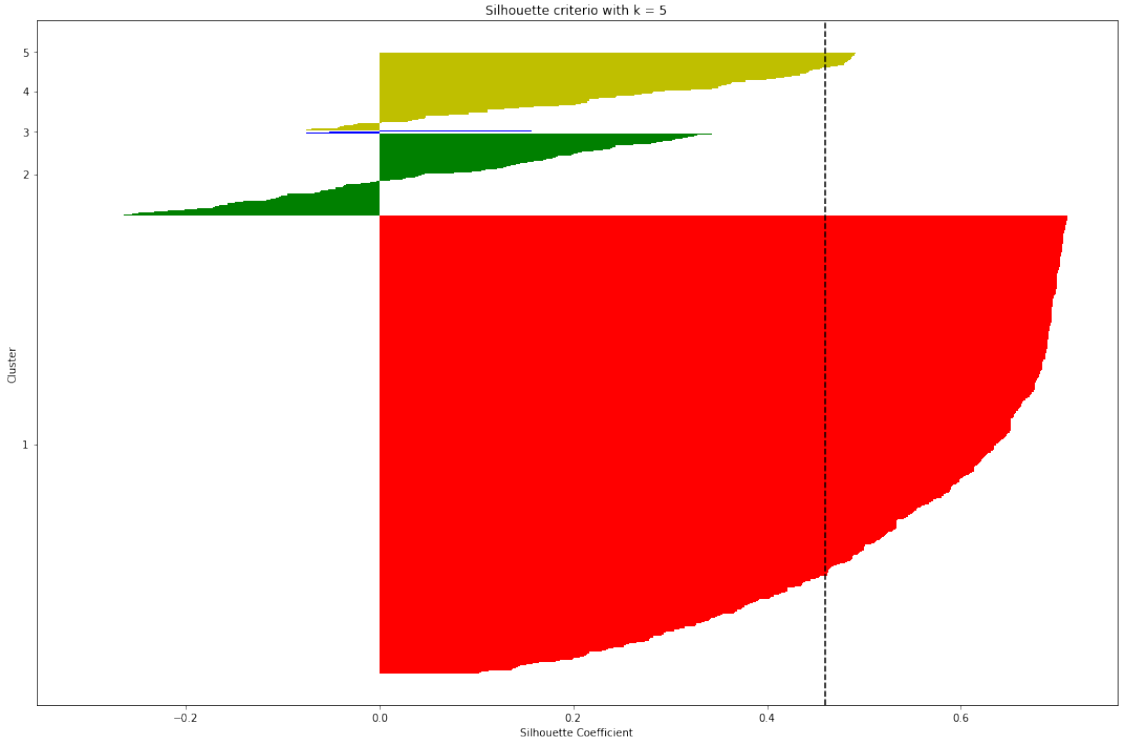
[92]:
```
k_values = [2,3,4,5,6,7,8,9,10]
for k in k_values:

    plt.figure(figsize=(15,10))

    kmeans = KMeans(n_clusters=k, random_state=0, n_init=10)
    kmeans.fit(data2014_kmeans)
    y_clusters = kmeans.labels_
    cluster_labels = np.unique(y_clusters)

    silhouette_points= silhouette_samples(data2014_kmeans, y_clusters,␣
↪metric='euclidean')

    y_ax_lower, y_ax_upper = 0, 0
    yticks = []
```

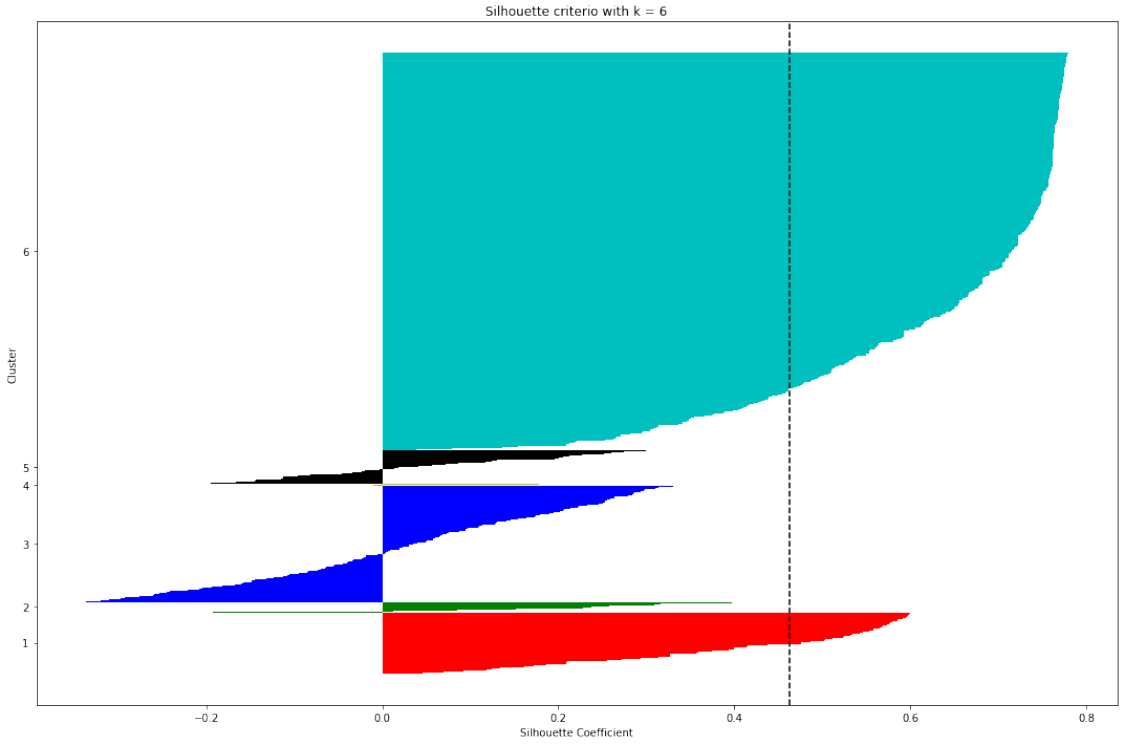
```
    colors = ['r', 'g', 'b', 'y', '0', 'c', 'm', 'k','aquamarine','tab:orange']
    for i, c in enumerate(cluster_labels):
        silhouette_points_c = silhouette_points[y_clusters == c]
        silhouette_points_c.sort()
        y_ax_upper += len(silhouette_points_c)
        color = colors[i]
        plt.barh(range(y_ax_lower, y_ax_upper), silhouette_points_c, height=1.0,
                 edgecolor='none', color=color)

        yticks.append((y_ax_lower + y_ax_upper) / 2.)
        y_ax_lower += len(silhouette_points_c)

    silhouette_mean = np.mean(silhouette_points)
    plt.axvline(silhouette_mean, color="black", linestyle="--")

    plt.yticks(yticks, cluster_labels + 1)
    plt.ylabel('Cluster')
    plt.xlabel('Silhouette Coefficient')
    plt.title(f'Silhouette criterio with k = {k}')
    plt.tight_layout()
    plt.show()
```

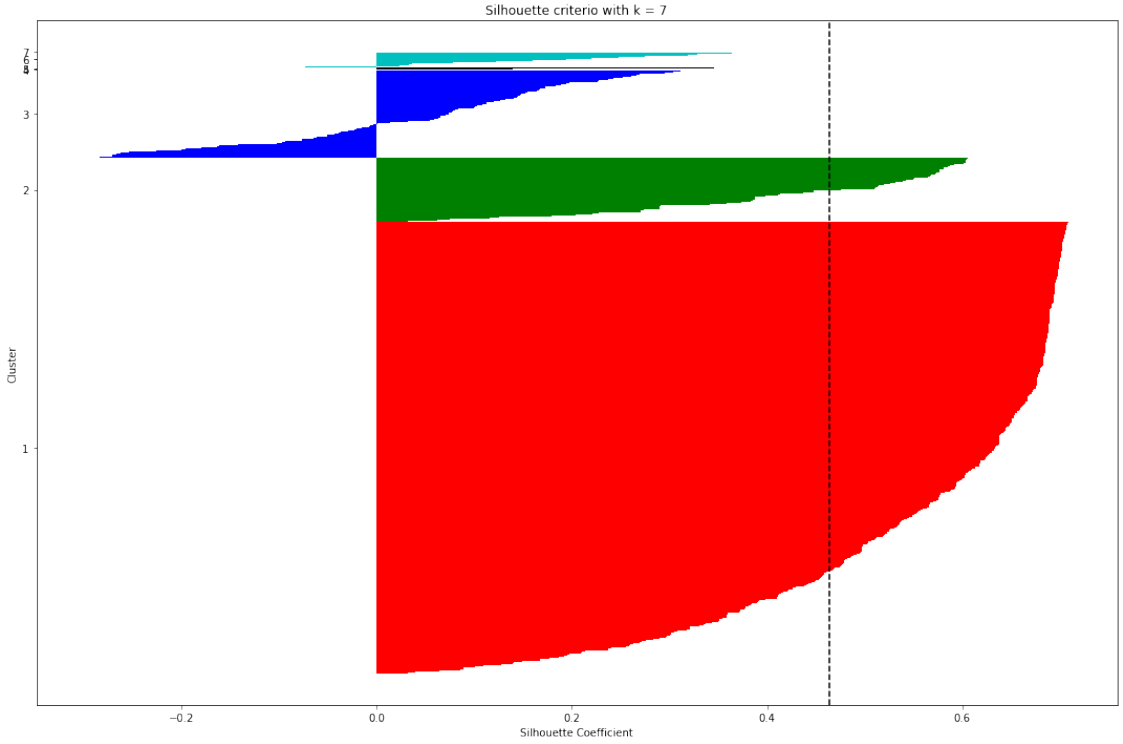


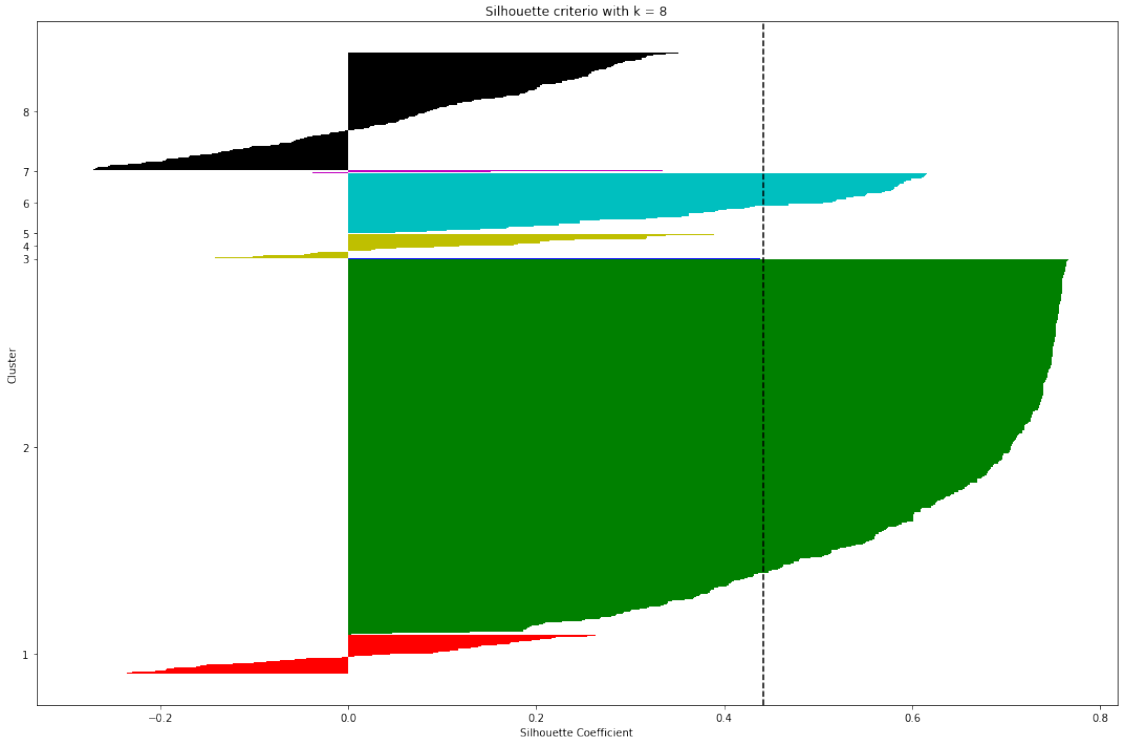

Silhouette criterio with k = 3
Cluster
3
2
1
−0.2
0.0
0.2
0.4
0.6
0.8
Silhouette Coefficient

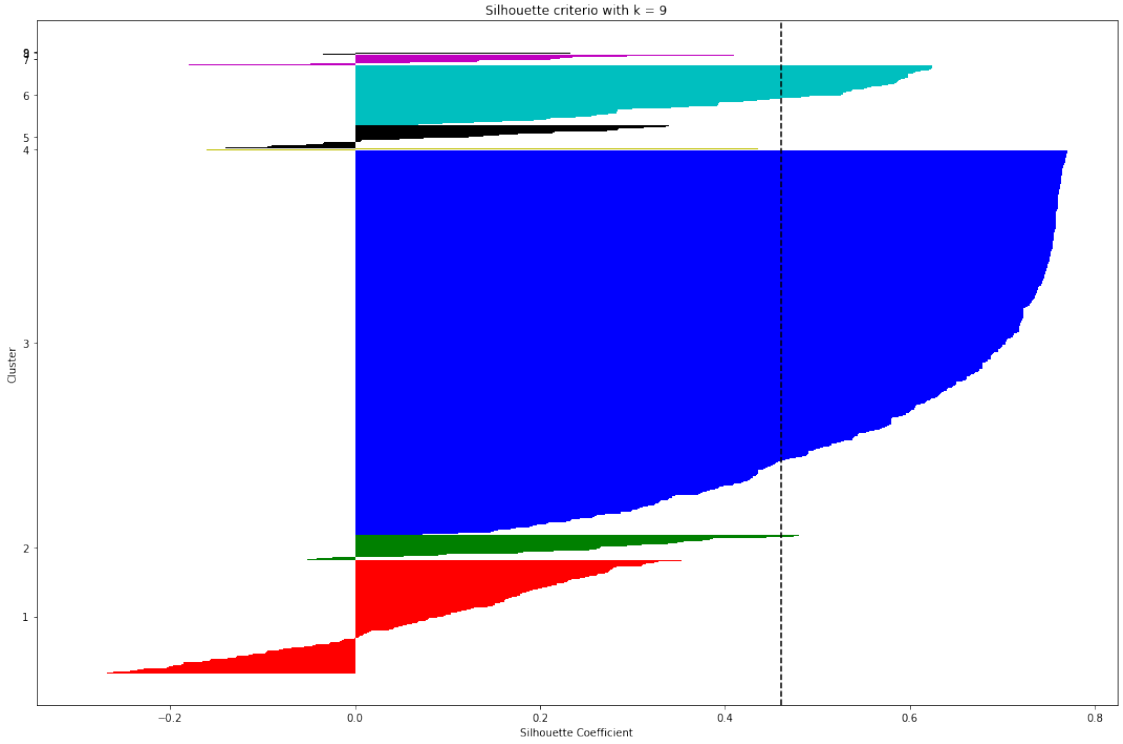

Silhouette criterio with k = 4
Cluster
4
3
2
1
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
Silhouette Coefficient

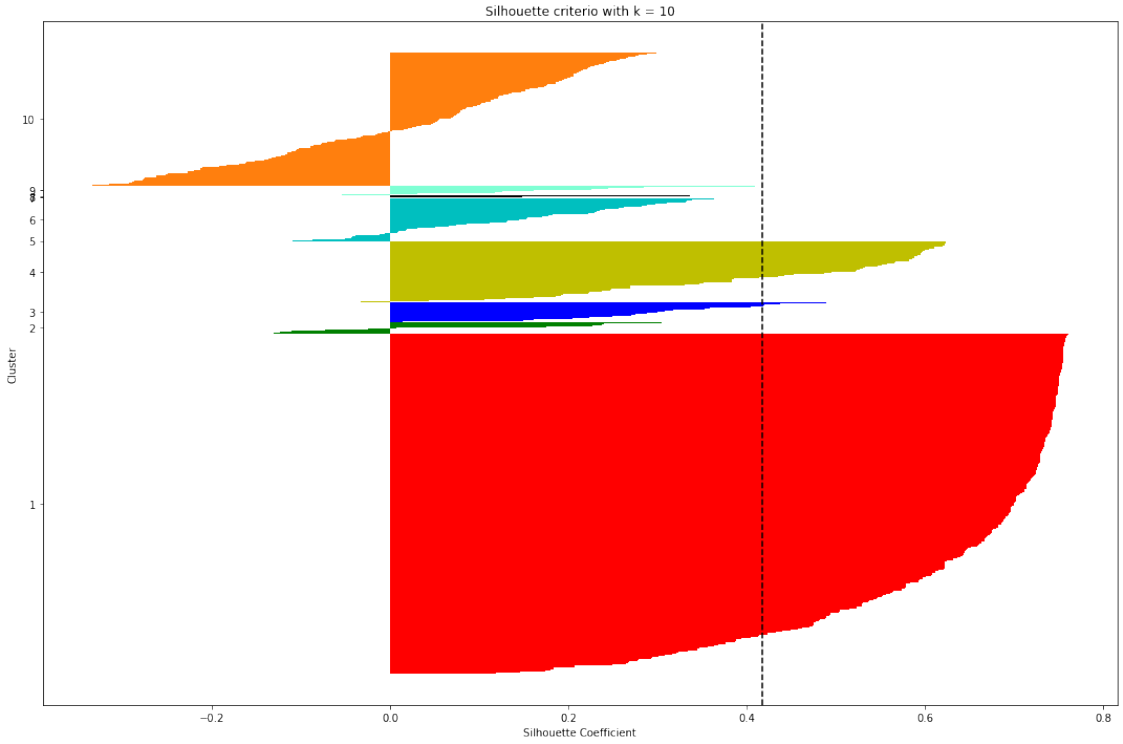

Silhouette criterio with k = 5
Cluster
5
4
3
2
1
−0.2
0.0
0.2
0.4
0.6
Silhouette Coefficient


Silhouette criterio with k = 6
Cluster
6
5
4
3
2
1
−0.2
0.0
0.2
0.4
0.6
0.8
Silhouette Coefficient


Silhouette criterio with k = 7
Cluster
Silhouette Coefficient
−0.2
0.0
0.2
0.4
0.6


Silhouette criterio with k = 8
Cluster
Silhouette Coefficient
−0.2
0.0
0.2
0.4
0.6
0.8


Silhouette criterio with k = 9
Cluster
Silhouette Coefficient
−0.2
0.0
0.2
0.4
0.6
0.8
1
2
3
4
5
6
7
8
9


Silhouette criterio with k = 10
Cluster
Silhouette Coefficient
−0.2
0.0
0.2
0.4
0.6
0.8
1
2
3
4
5
6
9
10

After analyzing the results from the silhouette method we found out that in each possibility at least a cluster has a negative value for their Silhouette coefficients and not all of them surpass de average silhouette score.

Nevertheless, the best one is k=9 where more clusters surpass the average silhuette coefficient. k=2 does not seem like a bad choice. Although, by taking into account the business intelligence criterion, it may not be the right approach. We will look at it later after analyzing the results of the next heuristic.

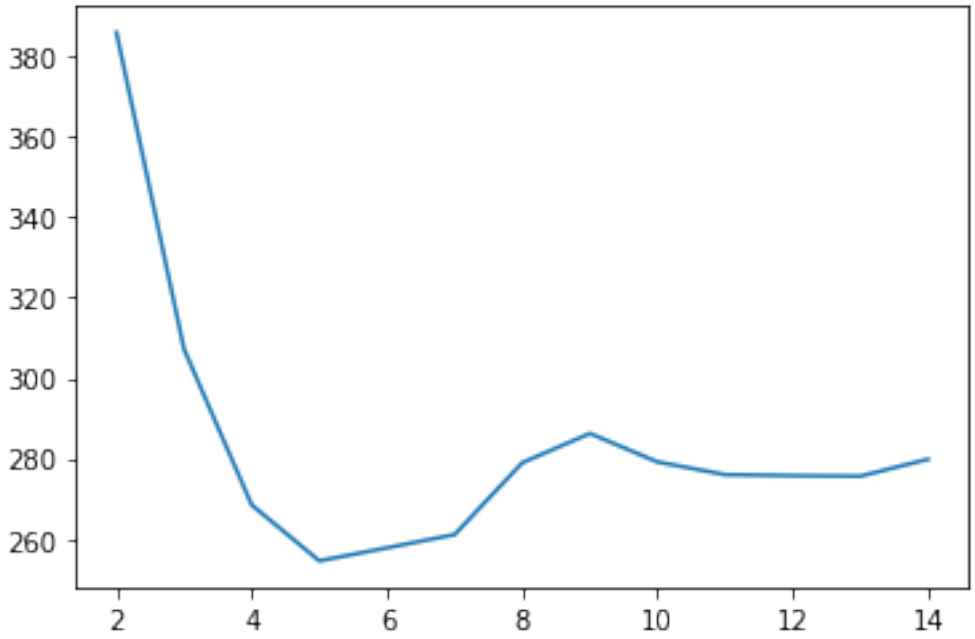
Calinski-Harabasz Finally, in order to be thorough, we will apply the Calinski-Harabasz criterion to have a new tool in order to choose the right k.

[93]:
```
CHs = []
for i in range(2,15) :
    km = KMeans(n_clusters=i, random_state=0)
    km.fit(data2014_kmeans)
    CH = calinski_harabasz_score(data2014_kmeans, km.labels_)
    CHs.append(CH)
CHs
```

[93]:
```
[385.8744638305435,
 307.1640453690942,
 268.4720905622556,
 254.58304464797408,
 257.8193225494758,
 261.1182379963743,
 278.96256763172346,
 286.23988087699394,
 279.1603478524493,
 275.9985934578568,
 275.75543899236544,
 275.6272161495632,
 279.82742431653276]
```

[94]:
```
plt.plot(range(2, 15), CHs)
```

[94]:
```
[<matplotlib.lines.Line2D at 0x7f9ce394dee0>]
```



Accoring to de Calinski-Harabasz criterion we found out that the ideal k would be 2 or 3. Even 9 seems like a good candidate. 2 and 9 agree with the two previous heuristics.

We will use k = 9 and k = 2 (2 clusters seem to little to profile clients but we will observe the results.

From the business point it would be advisable to identify more clusters (i.e. 9) in order, for example, to profile clients and be able to establish specific marketing campaigns for each of these clients in order for them to increase their purchases (not only related to the amount of transactions, but to the quantity paid for a product). It is important to find out different groups so the company or companies may offer special deals for products that clients may want.

1.4.3 K-means with k = 9

[95]:
```
kmeans2014_k9 = KMeans(n_clusters=9, random_state=0, n_init=10)
kmeans2014_k9.fit(data2014_kmeans)
```

[95]:
```
KMeans(n_clusters=9, random_state=0)
```

[96]:
```
print("It took K-means", kmeans2014_k9.n_iter_, "iterations to converge, with a␣
↪final WSS (Within Sum of Squares) of:",
      kmeans2014_k9.inertia_, "and the following centroids:", kmeans2014_k9.
↪cluster_centers_)
```

```
It took K-means 13 iterations to converge, with a final WSS (Within Sum of
Squares) of: 2857.6306860727555 and the following centroids: [[ 1.18346013e+00
```

```
-3.75628041e-01 -7.22358052e-02 -1.40538629e-01
  -1.30905455e-01 -4.75766606e-02  2.91208791e-01]
 [ 3.78139735e+00  2.08763926e+00 -1.09291610e-01 -1.03913968e-01
  -2.06301882e-01 -2.53664493e-01  1.00000000e+00]
 [-1.03169380e+00 -3.00195552e-01 -5.02387760e-02  7.07925146e-02
   1.79130070e-02  2.12711969e-02 -4.16333634e-16]
 [ 5.87904684e+00 -3.95120534e+00  1.42443921e+01  6.18770172e+00
   9.54445792e+00  2.97146617e+00  2.00000000e+00]
 [ 5.94488755e+00 -1.52360538e+00 -9.23368680e-01 -7.87202512e-02
   1.01344728e+00 -8.26816911e-01  1.00000000e+00]
 [-2.97341231e-01  2.39806088e+00  4.51899933e-01 -4.58148061e-01
   4.33690359e-01 -2.41249162e-01  5.55111512e-17]
 [ 3.08460750e+00  9.11092093e-01 -1.52391570e+00  1.78918718e+00
  -2.08271735e+00  4.30909709e+00  7.50000000e-01]
 [ 3.76392059e+00 -3.52542517e+00  4.67202596e+00 -5.76785407e+00
  -8.29522307e+00 -2.57390302e+00  5.00000000e-01]
 [ 4.34121203e+00  2.75476818e+00  1.58422474e+01  1.55153079e+01
  -1.12959932e+01 -5.12939791e+00  2.00000000e+00]]
```

We can see by counting the labels resulting from the k-means, that there exist 3 clusters with not many observations, so it would be difficult to profile those clients. Later on we will compare the results.

[97]:
```
counter = Counter(kmeans2014_k9.labels_)
counter
```

[97]:
```
Counter({2: 615, 0: 182, 4: 37, 5: 97, 6: 16, 1: 39, 7: 4, 3: 2, 8: 1})
```

[98]:
```
data2014_kmeans['Cluster'] = kmeans2014_k9.labels_
data2014_kmeans
```

[98]:

CARD NUMBER	High number of work/personal transactions	\
************0007	-1.102613	
************0015	-1.510223	
************0040	-1.279980	
************0047	3.035147	
************0057	-1.244730	
…	…	
************9957	1.126939	
************9963	-1.360561	
************9968	-1.224845	
************9971	0.046710	
************9989	3.667547	

CARD NUMBER	Vehicle and standard VAT card purchases	\
************0007	-0.609923	

************0015	-0.474573
************0040	0.171561
************0047	-0.145967
************0057	-0.104075
…	…
************9957	1.249133
************9963	-0.450382
************9968	0.199069
************9971	-0.869348
************9989	-0.836873

CARD NUMBER	High legal expenses and transport transactions \
************0007	-0.078921
************0015	-0.049181
************0040	0.070341
************0047	-0.586113
************0057	-0.108621
…	…
************9957	-0.095804
************9963	-0.067909
************9968	0.106738
************9971	0.040100
************9989	-3.051555

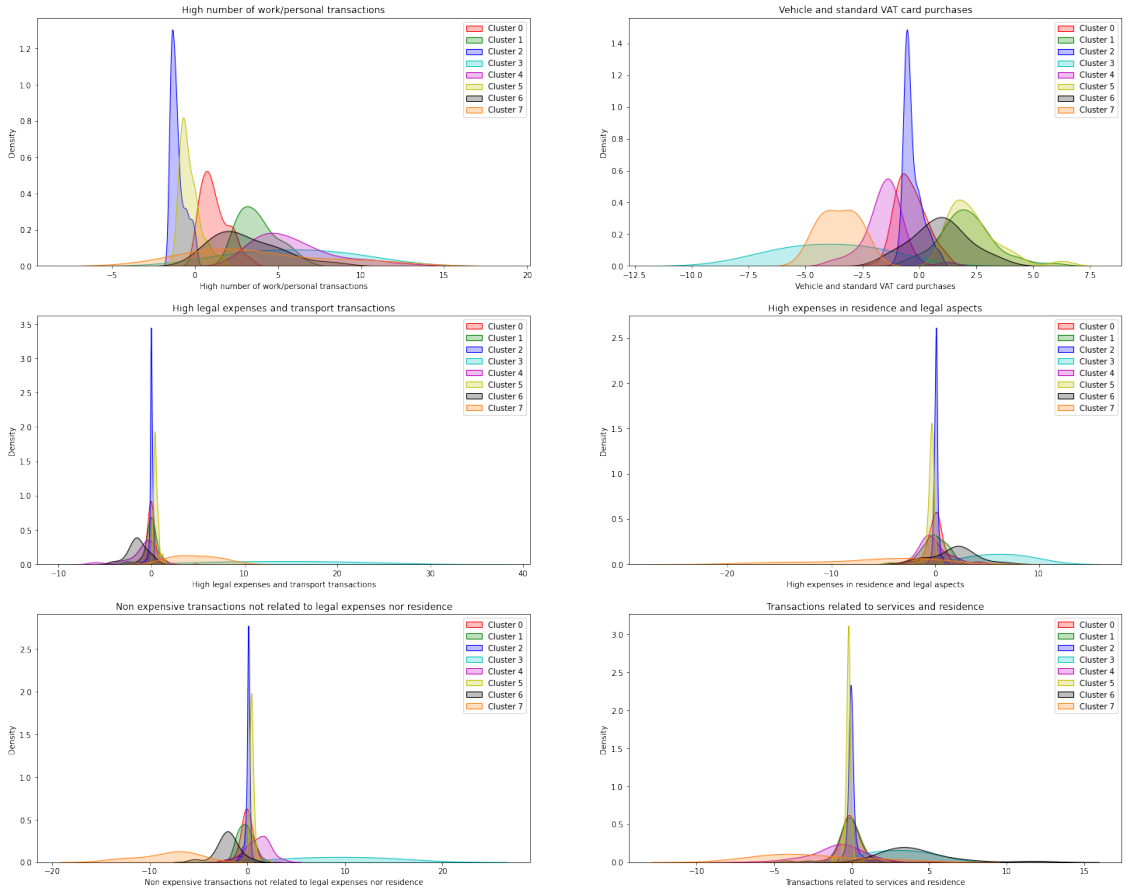
CARD NUMBER	High expenses in residence and legal aspects \
************0007	-0.002634
************0015	0.109719
************0040	-0.017045
************0047	0.532373
************0057	0.089445
…	…
************9957	0.050369
************9963	0.153903
************9968	-0.070366
************9971	-0.278259
************9989	2.046489

CARD NUMBER	Non expensive transactions not related to legal expenses nor residence \
************0007	0.090542
************0015	0.067047
************0040	0.157104
************0047	-0.254677
************0057	0.235960

```
…                                                        …
************9957                                      -0.173236
************9963                                       0.024214
************9968                                       0.219315
************9971                                       0.110072
************9989                                       0.870602

                 Transactions related to services and residence  Cluster
CARD NUMBER
************0007                                       0.506741        2
************0015                                      -0.008816        2
************0040                                      -0.063487        2
************0047                                      -0.018432        0
************0057                                      -0.274399        2
…                                                          …        …
************9957                                       0.005066        0
************9963                                      -0.012115        2
************9968                                      -0.085066        2
************9971                                      -0.316886        2
************9989                                      -2.621232        4

[993 rows x 7 columns]
```

[99]:
```
var_num = data2014_kmeans.columns[0:6]
fig = plt.figure(figsize=(25,20))
i=1
for var in var_num:
    ax = fig.add_subplot(math.ceil(len(var_num)/2), 2, i)
    sns.kdeplot(data2014_kmeans.loc[data2014_kmeans.Cluster==0][var],␣
↪shade=True, color='r', ax=ax);
    sns.kdeplot(data2014_kmeans.loc[data2014_kmeans.Cluster==1][var],␣
↪shade=True, color='g', ax=ax);
    sns.kdeplot(data2014_kmeans.loc[data2014_kmeans.Cluster==2][var],␣
↪shade=True, color='b', ax=ax);
    sns.kdeplot(data2014_kmeans.loc[data2014_kmeans.Cluster==3][var],␣
↪shade=True, color='c', ax=ax);
    sns.kdeplot(data2014_kmeans.loc[data2014_kmeans.Cluster==4][var],␣
↪shade=True, color='m', ax=ax);
    sns.kdeplot(data2014_kmeans.loc[data2014_kmeans.Cluster==5][var],␣
↪shade=True, color='y', ax=ax);
    sns.kdeplot(data2014_kmeans.loc[data2014_kmeans.Cluster==6][var],␣
↪shade=True, color='k', ax=ax);
    sns.kdeplot(data2014_kmeans.loc[data2014_kmeans.Cluster==7][var],␣
↪shade=True, color='tab:orange', ax=ax);
    sns.kdeplot(data2014_kmeans.loc[data2014_kmeans.Cluster==8][var],␣
↪shade=True, color='tab:olive', ax=ax);
```

```
    plt.title(var)
    plt.legend(['Cluster 0', 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster␣
↪4', 'Cluster 5', 'Cluster 6', 'Cluster 7','Cluster 8' ])
    i+=1
```

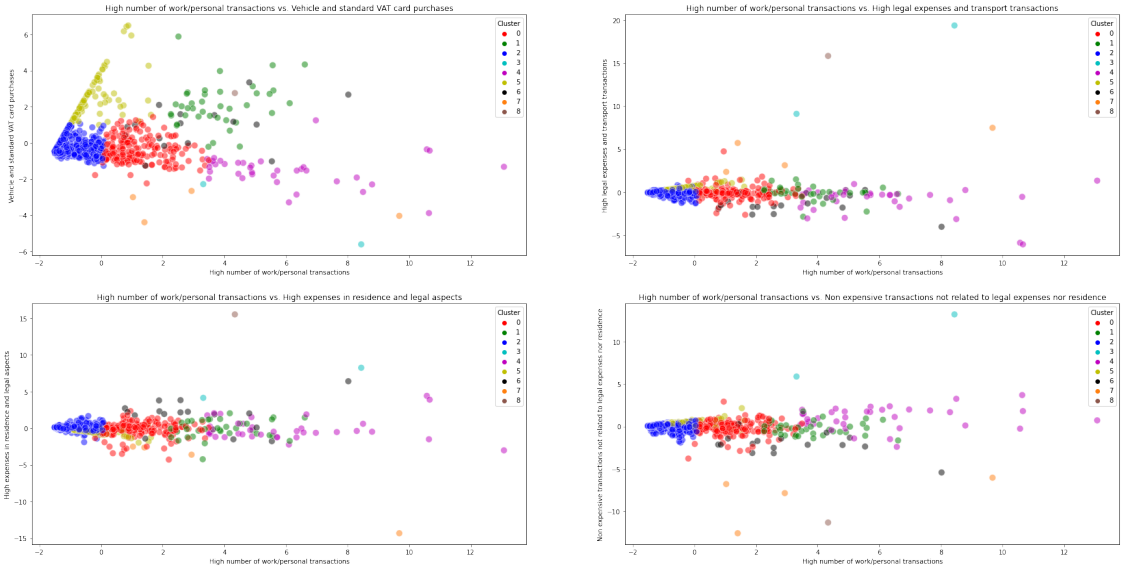


We will see the scatterplots to better understand the differences

The other elements observed in the scatter plots confirm our description of the two other clusters.

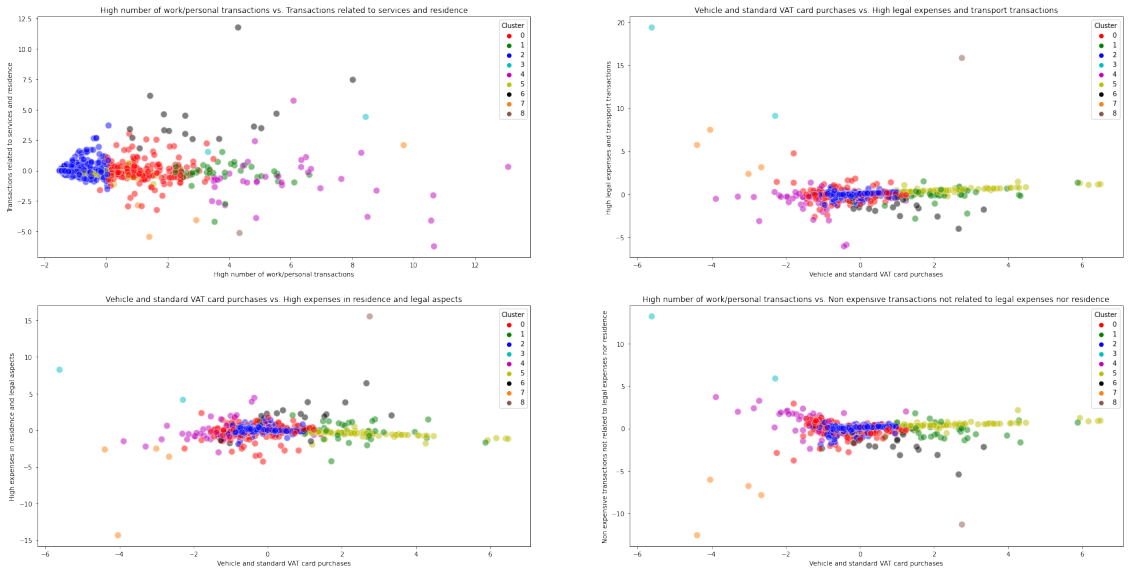
[100]:
```
fig = plt.figure(figsize=(30,15))
colorPalette = ["r", "g", "b", "c", "m", "y", "k", "tab:orange","tab:brown"]
ax = fig.add_subplot(2, 2, 1)
sns.scatterplot(x='High number of work/personal transactions', y='Vehicle and␣
↪standard VAT card purchases', hue="Cluster", data=data2014_kmeans, ax=ax,␣
↪palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. Vehicle and standard␣
↪VAT card purchases")
ax = fig.add_subplot(2, 2, 2)
```

```
sns.scatterplot(x="High number of work/personal transactions", y="High legal␣
 ↪expenses and transport transactions", hue="Cluster", data=data2014_kmeans,␣
 ↪ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. High legal expenses␣
 ↪and transport transactions")
ax = fig.add_subplot(2, 2, 3)
sns.scatterplot(x="High number of work/personal transactions", y="High expenses␣
 ↪in residence and legal aspects", hue="Cluster", data=data2014_kmeans, ax=ax,␣
 ↪palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. High expenses in␣
 ↪residence and legal aspects")
ax = fig.add_subplot(2, 2, 4)
sns.scatterplot(x='High number of work/personal transactions', y='Non expensive␣
 ↪transactions not related to legal expenses nor residence', hue="Cluster",␣
 ↪data=data2014_kmeans, ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. Non expensive␣
 ↪transactions not related to legal expenses nor residence")
plt.show()
```



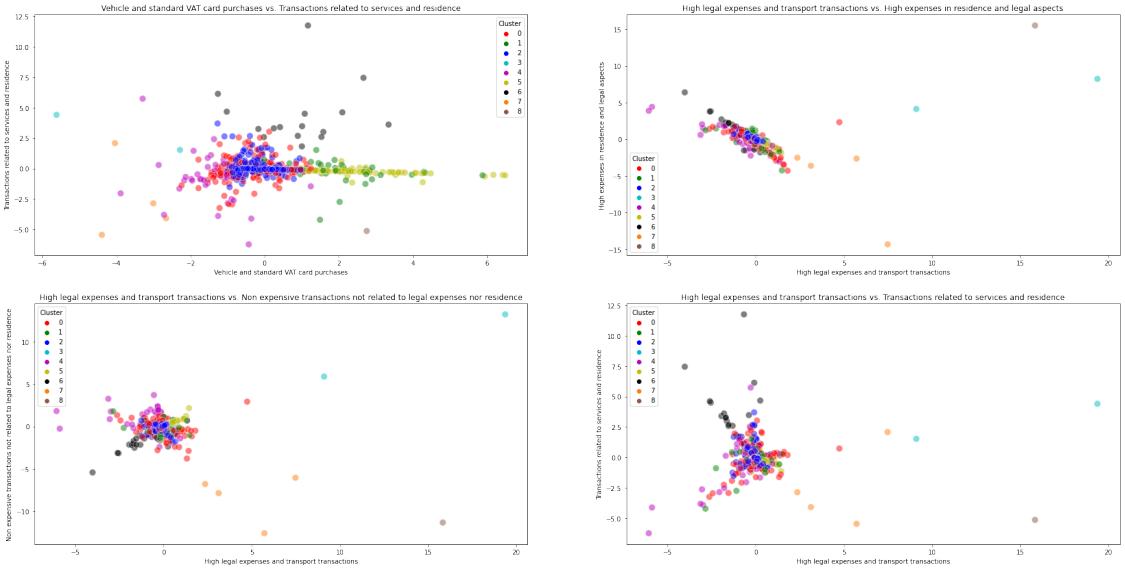
[101]:
```
fig = plt.figure(figsize=(30,15))
colorPalette = ["r", "g", "b", "c", "m", "y", "k", "tab:orange","tab:brown"]
ax = fig.add_subplot(2, 2, 1)
sns.scatterplot(x="High number of work/personal transactions", y="Transactions␣
 ↪related to services and residence", hue="Cluster", data=data2014_kmeans,␣
 ↪ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. Transactions related␣
 ↪to services and residence")
```

```
ax = fig.add_subplot(2, 2, 2)
sns.scatterplot(x="Vehicle and standard VAT card purchases", y="High legal␣
 ↪expenses and transport transactions", hue="Cluster", data=data2014_kmeans,␣
 ↪ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("Vehicle and standard VAT card purchases vs. High legal expenses and␣
 ↪transport transactions")
ax = fig.add_subplot(2, 2, 3)
sns.scatterplot(x='Vehicle and standard VAT card purchases', y='High expenses␣
 ↪in residence and legal aspects', hue="Cluster", data=data2014_kmeans, ax=ax,␣
 ↪palette=colorPalette, s=100, alpha=0.5)
plt.title("Vehicle and standard VAT card purchases vs. High expenses in␣
 ↪residence and legal aspects")
ax = fig.add_subplot(2, 2, 4)
sns.scatterplot(x="Vehicle and standard VAT card purchases", y="Non expensive␣
 ↪transactions not related to legal expenses nor residence", hue="Cluster",␣
 ↪data=data2014_kmeans, ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. Non expensive␣
 ↪transactions not related to legal expenses nor residence")
plt.show()
```



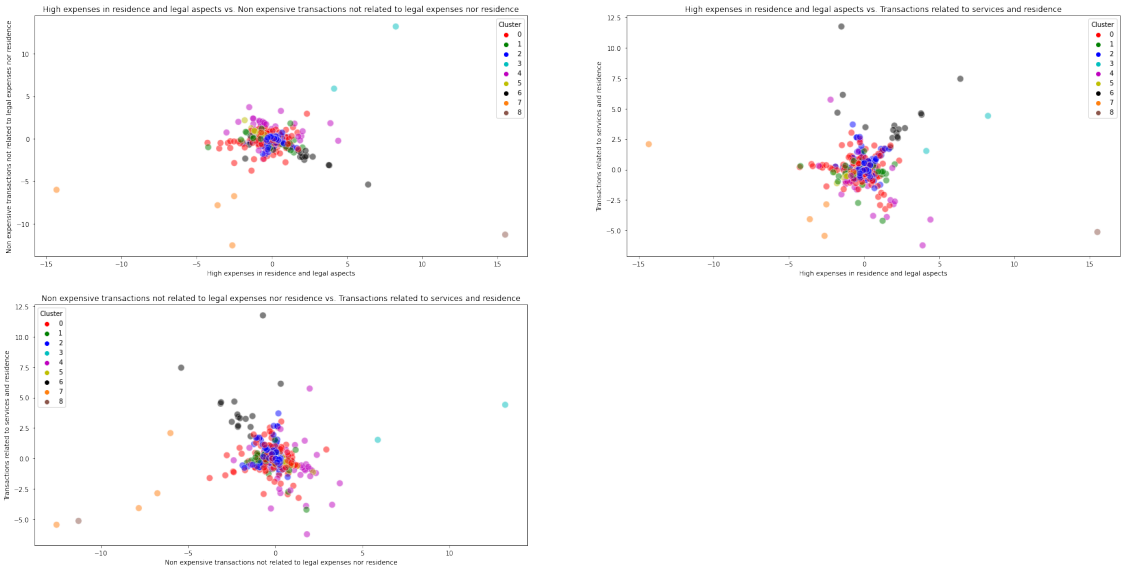
[102]:
```
fig = plt.figure(figsize=(30,15))
colorPalette = ["r", "g", "b", "c", "m", "y", "k", "tab:orange","tab:brown"]
ax = fig.add_subplot(2, 2, 1)
sns.scatterplot(x="Vehicle and standard VAT card purchases", y="Transactions␣
 ↪related to services and residence", hue="Cluster", data=data2014_kmeans,␣
 ↪ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("Vehicle and standard VAT card purchases vs. Transactions related to␣
 ↪services and residence")
```

```
ax = fig.add_subplot(2, 2, 2)
sns.scatterplot(x='High legal expenses and transport transactions', y='High␣
 ↪expenses in residence and legal aspects', hue="Cluster",␣
 ↪data=data2014_kmeans, ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High legal expenses and transport transactions vs. High expenses in␣
 ↪residence and legal aspects")
ax = fig.add_subplot(2, 2, 3)
sns.scatterplot(x="High legal expenses and transport transactions", y="Non␣
 ↪expensive transactions not related to legal expenses nor residence",␣
 ↪hue="Cluster", data=data2014_kmeans, ax=ax, palette=colorPalette, s=100,␣
 ↪alpha=0.5)
plt.title("High legal expenses and transport transactions vs. Non expensive␣
 ↪transactions not related to legal expenses nor residence")
ax = fig.add_subplot(2, 2, 4)
sns.scatterplot(x="High legal expenses and transport transactions",␣
 ↪y="Transactions related to services and residence", hue="Cluster",␣
 ↪data=data2014_kmeans, ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High legal expenses and transport transactions vs. Transactions␣
 ↪related to services and residence")
plt.show()
```



[103]:
```
fig = plt.figure(figsize=(30,15))
colorPalette = ["r", "g", "b", "c", "m", "y", "k", "tab:orange","tab:brown"]
ax = fig.add_subplot(2, 2, 1)
sns.scatterplot(x="High expenses in residence and legal aspects", y="Non␣
 ↪expensive transactions not related to legal expenses nor residence",␣
 ↪hue="Cluster", data=data2014_kmeans, ax=ax, palette=colorPalette, s=100,␣
 ↪alpha=0.5)
```

```
plt.title("High expenses in residence and legal aspects vs. Non expensive␣
 ↪transactions not related to legal expenses nor residence")
ax = fig.add_subplot(2, 2, 2)
sns.scatterplot(x="High expenses in residence and legal aspects",␣
 ↪y="Transactions related to services and residence", hue="Cluster",␣
 ↪data=data2014_kmeans, ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High expenses in residence and legal aspects vs. Transactions␣
 ↪related to services and residence")
ax = fig.add_subplot(2, 2, 3)
sns.scatterplot(x="Non expensive transactions not related to legal expenses nor␣
 ↪residence", y="Transactions related to services and residence",␣
 ↪hue="Cluster", data=data2014_kmeans, ax=ax, palette=colorPalette, s=100,␣
 ↪alpha=0.5)
plt.title("Non expensive transactions not related to legal expenses nor␣
 ↪residence vs. Transactions related to services and residence")
plt.show()
```



Profiling clients with k-means using k=9 By observing the density plots and the scatter plots we will try to figure out the profile of clients, although some of the plots are not that clear.

Cluster 0: clients with less than average of High number of work/personal transactions, high Vehicle and standard VAT card purchases, low High legal expenses and transport transactions, more than average in High expenses in residence and legal aspects, average in Non expensive transactions not related to legal expenses nor residence and average in Transactions related to services and residence.

Cluster 1: clients with average High number of work/personal transactions, high Vehicle and standard VAT card purchases, low High legal expenses and transport transactions, more than average in High expenses in residence and legal aspects, average in Non expensive transactions not related

to legal expenses nor residence and average in Transactions related to services and residence.

Cluster 2: clients with the lowest High number of work/personal transactions, more than average of Vehicle and standard VAT card purchases, low High legal expenses and transport transactions, more than average in High expenses in residence and legal aspects, average in Non expensive transactions not related to legal expenses nor residence and average in Transactions related to services and residence.

Cluster 3: clients with high High number of work/personal transactions, the lowest Vehicle and standard VAT card purchases, the highest High legal expenses and transport transactions, the highest in High expenses in residence and legal aspects, the highest in Non expensive transactions not related to legal expenses nor residence and the highest in Transactions related to services and residence.

Cluster 4: clients with more than average of High number of work/personal transactions, less that average of Vehicle and standard VAT card purchases, low High legal expenses and transport transactions, more than average in High expenses in residence and legal aspects, average in Non expensive transactions not related to legal expenses nor residence and average in Transactions related to services and residence.

Cluster 5: clients with low High number of work/personal transactions, the highest Vehicle and standard VAT card purchases, low High legal expenses and transport transactions, more than average in High expenses in residence and legal aspects, average in Non expensive transactions not related to legal expenses nor residence and average in Transactions related to services and residence.

Cluster 6: clients with average High number of work/personal transactions, more than average Vehicle and standard VAT card purchases, the lowest High legal expenses and transport transactions, high in High expenses in residence and legal aspects, less than average in Non expensive transactions not related to legal expenses nor residence and high in Transactions related to services and residence.

Cluster 7: clients with less average of High number of work/personal transactions, low Vehicle and standard VAT card purchases, less than average in High legal expenses and transport transactions, the highest in High expenses in residence and legal aspects, the highest in Non expensive transactions not related to legal expenses nor residence and the lowest on Transactions related to services and residence.

Cluster 8: just one data point. Not much to say.

1.4.4 K-means with k = 2

[104]:
```
kmeans2014_k2 = KMeans(n_clusters=2, random_state=0, n_init=10)
kmeans2014_k2.fit(data2014_kmeans)
```

[104]:
```
KMeans(n_clusters=2, random_state=0)
```

[105]:
```
counter = Counter(kmeans2014_k2.labels_)
counter
```

[105]:
```
Counter({1: 797, 0: 196})
```

[106]:
```
data2014_kmeans['Cluster'] = kmeans2014_k2.labels_
data2014_kmeans
```

[106]:

CARD NUMBER	High number of work/personal transactions	Vehicle and standard VAT card purchases	High legal expenses and transport transactions	High expenses in residence and legal aspects
************0007	-1.102613	-0.609923	-0.078921	
************0015	-1.510223	-0.474573	-0.049181	
************0040	-1.279980	0.171561	0.070341	
************0047	3.035147	-0.145967	-0.586113	
************0057	-1.244730	-0.104075	-0.108621	
…	…	…	…	
************9957	1.126939	1.249133	-0.095804	
************9963	-1.360561	-0.450382	-0.067909	
************9968	-1.224845	0.199069	0.106738	
************9971	0.046710	-0.869348	0.040100	
************9989	3.667547	-0.836873	-3.051555	

```
************0007                                    -0.002634
************0015                                     0.109719
************0040                                    -0.017045
************0047                                     0.532373
************0057                                     0.089445
…                                                        …
************9957                                     0.050369
************9963                                     0.153903
************9968                                    -0.070366
************9971                                    -0.278259
************9989                                     2.046489

                Non expensive transactions not related to legal expenses nor
residence  \
CARD NUMBER
************0007                                       0.090542
************0015                                       0.067047
************0040                                       0.157104
************0047                                      -0.254677
************0057                                       0.235960
…                                                          …
************9957                                      -0.173236
************9963                                       0.024214
************9968                                       0.219315
************9971                                       0.110072
************9989                                       0.870602

                Transactions related to services and residence  Cluster
CARD NUMBER
************0007                                        0.506741        1
************0015                                       -0.008816        1
************0040                                       -0.063487        1
************0047                                       -0.018432        1
************0057                                       -0.274399        1
…                                                          …       …
************9957                                        0.005066        1
************9963                                       -0.012115        1
************9968                                       -0.085066        1
************9971                                       -0.316886        1
************9989                                       -2.621232        0

[993 rows x 7 columns]
```

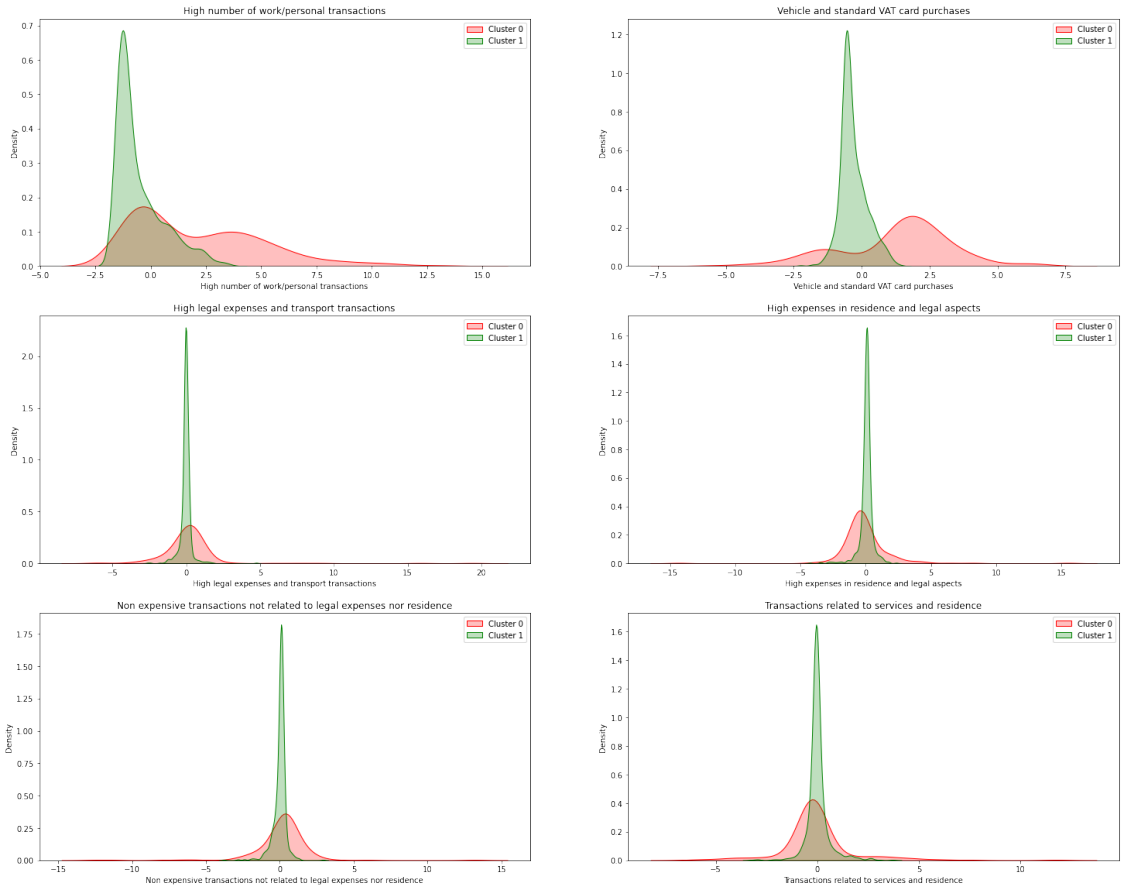
We save the dataframe with k-means where k = 2.

[107]:
```
df2014_kmeans2_pkl_filename = '/Users/andresaristi/Documents/
↪BirminghamCardTransactions/pickle_files/df2014_kmeans2.pkl'
```

```
df2014_kmeans2_pkl = open(df2014_kmeans2_pkl_filename, 'wb')
pickle.dump(data2014_kmeans, df2014_kmeans2_pkl)
df2014_kmeans2_pkl.close()
```

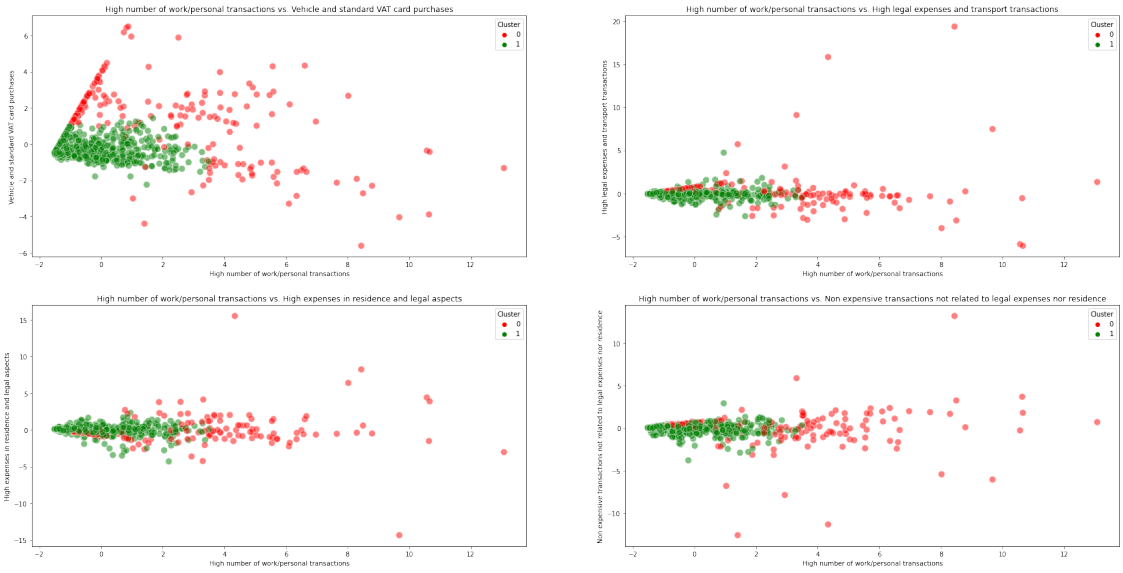
Densitity plots to help understand the clusters.

[108]:
```
var_num = data2014_kmeans.columns[0:6]
fig = plt.figure(figsize=(25,20))
i=1
for var in var_num:
    ax = fig.add_subplot(math.ceil(len(var_num)/2), 2, i)
    sns.kdeplot(data2014_kmeans.loc[data2014_kmeans.Cluster==0][var],␣
 ↪shade=True, color='r', ax=ax);
    sns.kdeplot(data2014_kmeans.loc[data2014_kmeans.Cluster==1][var],␣
 ↪shade=True, color='g', ax=ax);
    plt.title(var)
    plt.legend(['Cluster 0', 'Cluster 1' ])
    i+=1
```

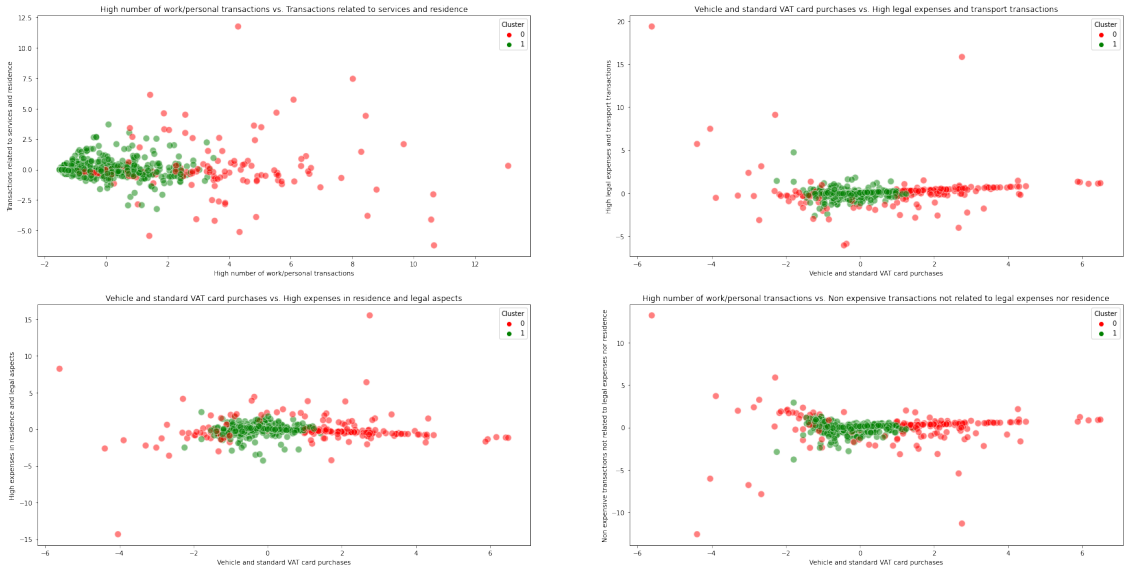


Scatter plots to help understand the clusters.

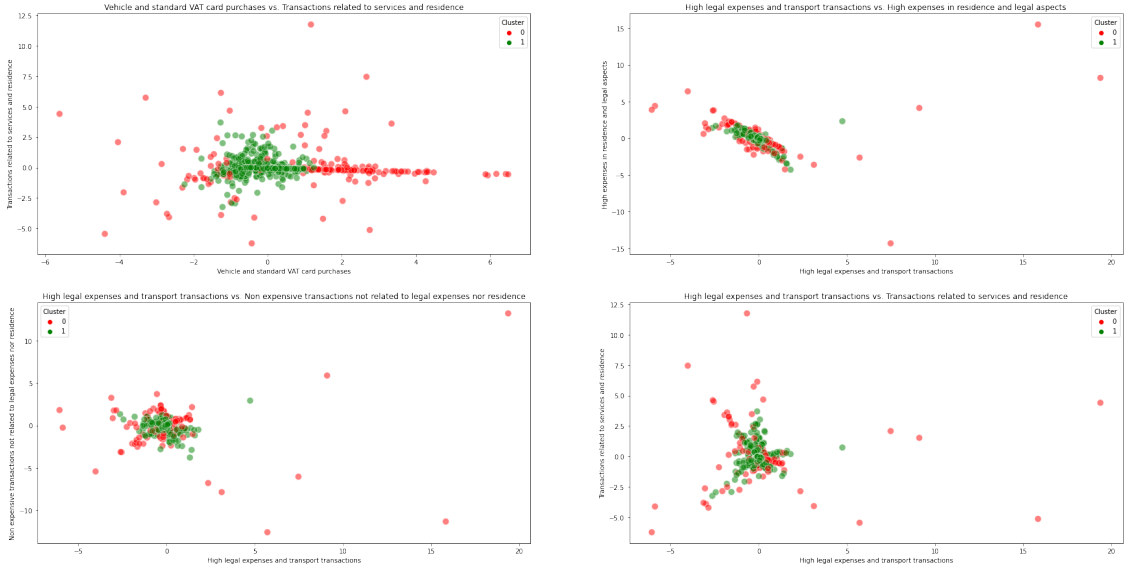
[109]:
```
fig = plt.figure(figsize=(30,15))
colorPalette = ["r", "g"]
ax = fig.add_subplot(2, 2, 1)
sns.scatterplot(x='High number of work/personal transactions', y='Vehicle and␣
 ↪standard VAT card purchases', hue="Cluster", data=data2014_kmeans, ax=ax,␣
 ↪palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. Vehicle and standard␣
 ↪VAT card purchases")
ax = fig.add_subplot(2, 2, 2)
sns.scatterplot(x="High number of work/personal transactions", y="High legal␣
 ↪expenses and transport transactions", hue="Cluster", data=data2014_kmeans,␣
 ↪ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. High legal expenses␣
 ↪and transport transactions")
ax = fig.add_subplot(2, 2, 3)
sns.scatterplot(x="High number of work/personal transactions", y="High expenses␣
 ↪in residence and legal aspects", hue="Cluster", data=data2014_kmeans, ax=ax,␣
 ↪palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. High expenses in␣
 ↪residence and legal aspects")
ax = fig.add_subplot(2, 2, 4)
sns.scatterplot(x='High number of work/personal transactions', y='Non expensive␣
 ↪transactions not related to legal expenses nor residence', hue="Cluster",␣
 ↪data=data2014_kmeans, ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. Non expensive␣
 ↪transactions not related to legal expenses nor residence")
plt.show()
```



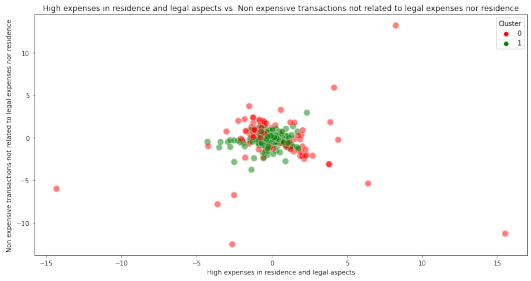
[110]:
```
fig = plt.figure(figsize=(30,15))
colorPalette = ["r", "g",]
ax = fig.add_subplot(2, 2, 1)
sns.scatterplot(x="High number of work/personal transactions", y="Transactions␣
 ↪related to services and residence", hue="Cluster", data=data2014_kmeans,␣
 ↪ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. Transactions related␣
 ↪to services and residence")
ax = fig.add_subplot(2, 2, 2)
sns.scatterplot(x="Vehicle and standard VAT card purchases", y="High legal␣
 ↪expenses and transport transactions", hue="Cluster", data=data2014_kmeans,␣
 ↪ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("Vehicle and standard VAT card purchases vs. High legal expenses and␣
 ↪transport transactions")
ax = fig.add_subplot(2, 2, 3)
sns.scatterplot(x='Vehicle and standard VAT card purchases', y='High expenses␣
 ↪in residence and legal aspects', hue="Cluster", data=data2014_kmeans, ax=ax,␣
 ↪palette=colorPalette, s=100, alpha=0.5)
plt.title("Vehicle and standard VAT card purchases vs. High expenses in␣
 ↪residence and legal aspects")
ax = fig.add_subplot(2, 2, 4)
sns.scatterplot(x="Vehicle and standard VAT card purchases", y="Non expensive␣
 ↪transactions not related to legal expenses nor residence", hue="Cluster",␣
 ↪data=data2014_kmeans, ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High number of work/personal transactions vs. Non expensive␣
 ↪transactions not related to legal expenses nor residence")
plt.show()
```

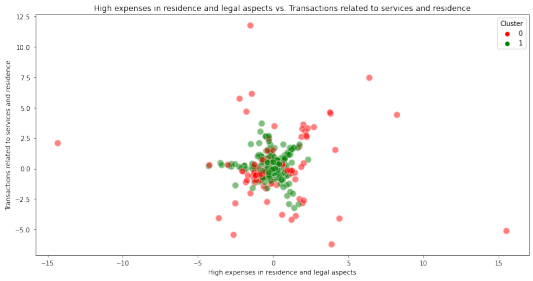


[111]:
```
fig = plt.figure(figsize=(30,15))
colorPalette = ["r", "g",]
ax = fig.add_subplot(2, 2, 1)
sns.scatterplot(x="Vehicle and standard VAT card purchases", y="Transactions␣
 ↪related to services and residence", hue="Cluster", data=data2014_kmeans,␣
 ↪ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("Vehicle and standard VAT card purchases vs. Transactions related to␣
 ↪services and residence")
ax = fig.add_subplot(2, 2, 2)
sns.scatterplot(x='High legal expenses and transport transactions', y='High␣
 ↪expenses in residence and legal aspects', hue="Cluster",␣
 ↪data=data2014_kmeans, ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High legal expenses and transport transactions vs. High expenses in␣
 ↪residence and legal aspects")
ax = fig.add_subplot(2, 2, 3)
sns.scatterplot(x="High legal expenses and transport transactions", y="Non␣
 ↪expensive transactions not related to legal expenses nor residence",␣
 ↪hue="Cluster", data=data2014_kmeans, ax=ax, palette=colorPalette, s=100,␣
 ↪alpha=0.5)
plt.title("High legal expenses and transport transactions vs. Non expensive␣
 ↪transactions not related to legal expenses nor residence")
ax = fig.add_subplot(2, 2, 4)
sns.scatterplot(x="High legal expenses and transport transactions",␣
 ↪y="Transactions related to services and residence", hue="Cluster",␣
 ↪data=data2014_kmeans, ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High legal expenses and transport transactions vs. Transactions␣
 ↪related to services and residence")
plt.show()
```



[112]:
```
fig = plt.figure(figsize=(30,15))
colorPalette = ["r", "g"]
ax = fig.add_subplot(2, 2, 1)
sns.scatterplot(x="High expenses in residence and legal aspects", y="Non␣
 ↪expensive transactions not related to legal expenses nor residence",␣
 ↪hue="Cluster", data=data2014_kmeans, ax=ax, palette=colorPalette, s=100,␣
 ↪alpha=0.5)
plt.title("High expenses in residence and legal aspects vs. Non expensive␣
 ↪transactions not related to legal expenses nor residence")
ax = fig.add_subplot(2, 2, 2)
sns.scatterplot(x="High expenses in residence and legal aspects",␣
 ↪y="Transactions related to services and residence", hue="Cluster",␣
 ↪data=data2014_kmeans, ax=ax, palette=colorPalette, s=100, alpha=0.5)
plt.title("High expenses in residence and legal aspects vs. Transactions␣
 ↪related to services and residence")
ax = fig.add_subplot(2, 2, 3)
sns.scatterplot(x="Non expensive transactions not related to legal expenses nor␣
 ↪residence", y="Transactions related to services and residence",␣
 ↪hue="Cluster", data=data2014_kmeans, ax=ax, palette=colorPalette, s=100,␣
 ↪alpha=0.5)
plt.title("Non expensive transactions not related to legal expenses nor␣
 ↪residence vs. Transactions related to services and residence")
plt.show()
```





Profiling clients with k-means using k=2 Even though k=2 seems like not such a good idea from the business point view, the data is telling us something. Therefore, by observing the density and scatter plots we will try to figure out the profile of clients.

Cluster 0: scattered group. Clients with really high High number of work/personal transactions (data spread up), very low to very high Vehicle and standard VAT card purchases, mainly the lowest High legal expenses and transport transactions (with a few high values), average in High expenses in residence and legal aspects, average in Non expensive transactions not related to legal expenses nor residence (although a few quite highs and quite lows) and a spread up data from really low to quite high in Transactions related to services and residence.

Cluster 1: dense group. Clients with average High number of work/personal transactions, the lowest Vehicle and standard VAT card purchases, the lowest High legal expenses and transport transactions, average in High expenses in residence and legal aspects, average in Non expensive transactions not related to legal expenses nor residence and less than average in Transactions related to services and residence.

1.4.5 Hierarchichal clustering

We will summarize the different linkages by means of Scipy.

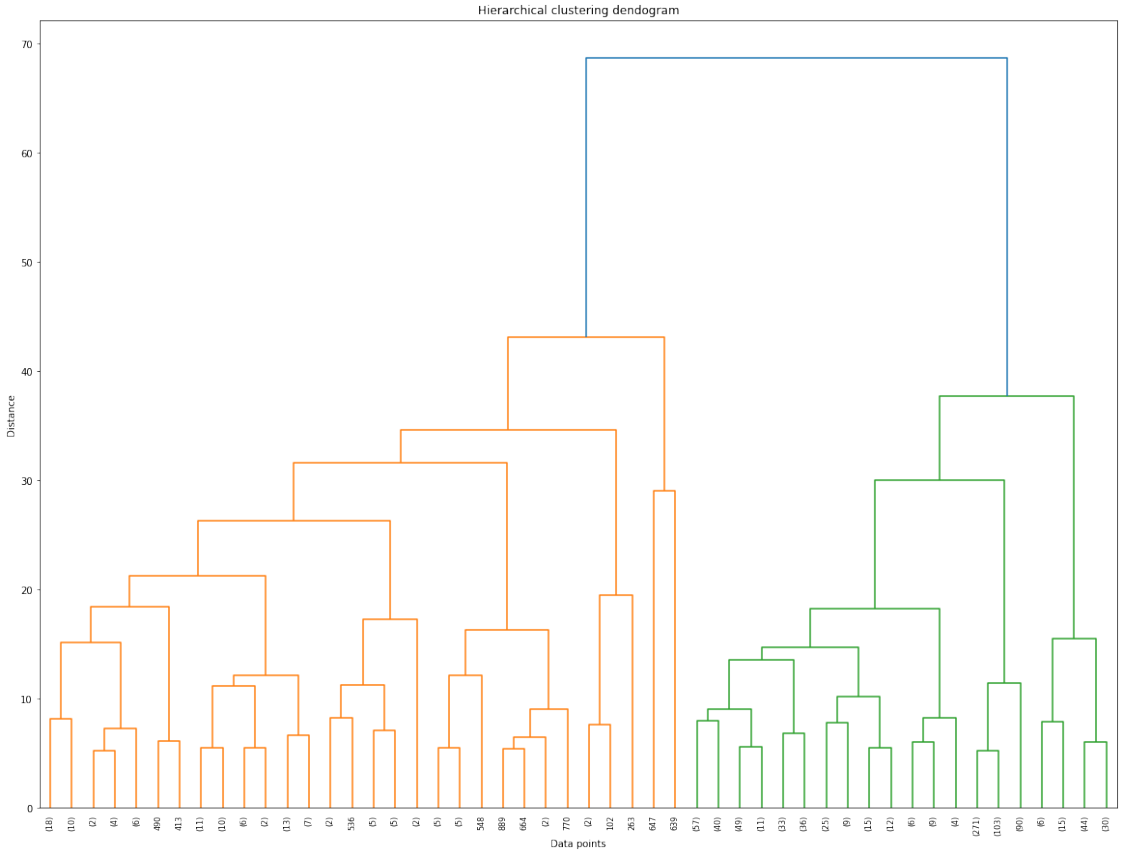
[113]:
```
data2014_hc = data2014_kmeans.copy()
X = data2014_hc.iloc[:,[0,1,2,3,4,5]].values
```

Dendogram We borrow a function from https://joernhees.de/blog/2015/08/26/scipy-hierarchical-clustering-and-dendrogram-tutorial/ that allows the returned data from the dendogram function to be improved.

[114]:
```
def fancy_dendrogram(*args, **kwargs):
    max_d = kwargs.pop('max_d', None)
    if max_d and 'color_threshold' not in kwargs:
        kwargs['color_threshold'] = max_d
    annotate_above = kwargs.pop('annotate_above', 0)
    ddata = dendrogram(*args, **kwargs)

    if not kwargs.get('no_plot', False):
        plt.title('Hierarchical Clustering Dendrogram (truncated)')
        plt.xlabel('sample index or (cluster size)')
        plt.ylabel('distance')
        for i, d, c in zip(ddata['icoord'], ddata['dcoord'],␣
↪ddata['color_list']):
            x = 0.5 * sum(i[1:3])
            y = d[1]
            if y > annotate_above:
                plt.plot(x, y, 'o', c=c)
                plt.annotate("%.3g" % y, (x, y), xytext=(0, -5),
                             textcoords='offset points',
                             va='top', ha='center')
        if max_d:
            plt.axhline(y=max_d, c='k')
```

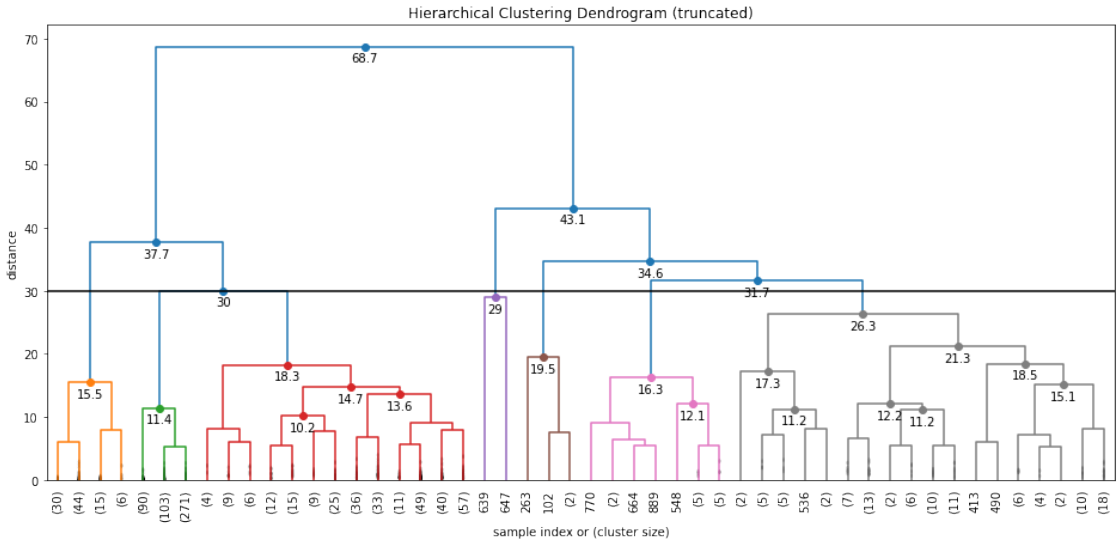
Ward linkage We plot the dendogram from Scipy using ward linkage.

[115]:
```
fusions = linkage(X, 'ward')
plt.figure(figsize=(20, 15))
plt.title('Hierarchical clustering dendogram')
plt.xlabel('Data points')
plt.ylabel('Distance')
dendrogram(fusions,
           orientation='top',
           distance_sort='descending',
           show_leaf_counts=True,
            p=50,
          truncate_mode = 'lastp')
plt.show()
```



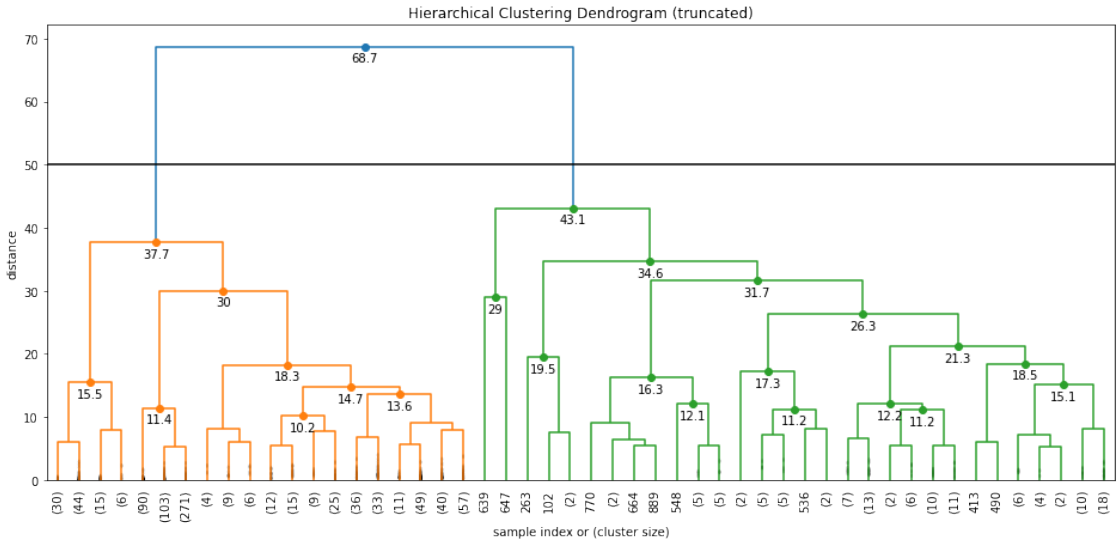
[116]:
```
plt.figure(figsize=(16, 7))
fancy_dendrogram(
    fusions,
    truncate_mode='lastp',
    p=50,
    leaf_rotation=90.,
```

```
    leaf_font_size=10.,
    show_contracted=True,
    annotate_above=10,
    max_d=30,  # plot a horizontal cut-off line
)
```



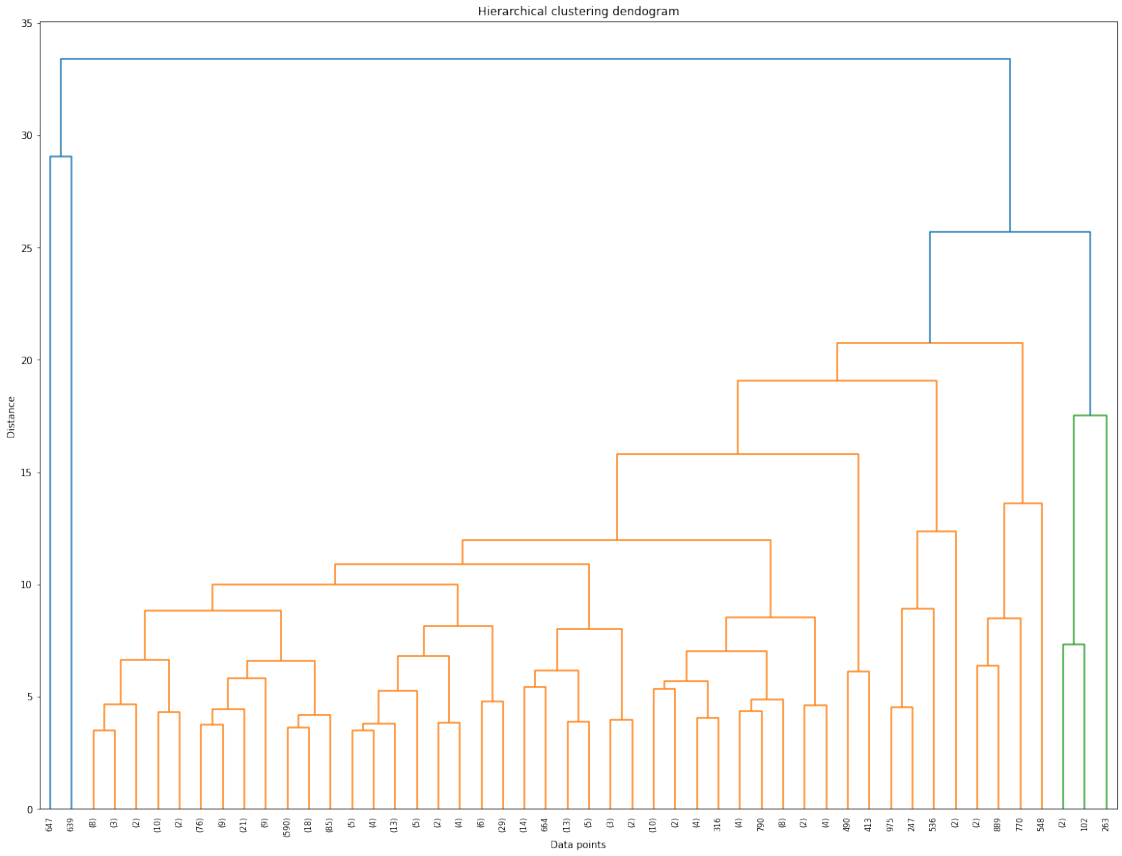
It seems that, using ward, k = 7 seems good

[117]:
```
plt.figure(figsize=(16, 7))
fancy_dendrogram(
    fusions,
    truncate_mode='lastp',
    p=50,
    leaf_rotation=90.,
    leaf_font_size=10.,
    show_contracted=True,
    annotate_above=10,
    max_d=50,  # plot a horizontal cut-off line
)
```

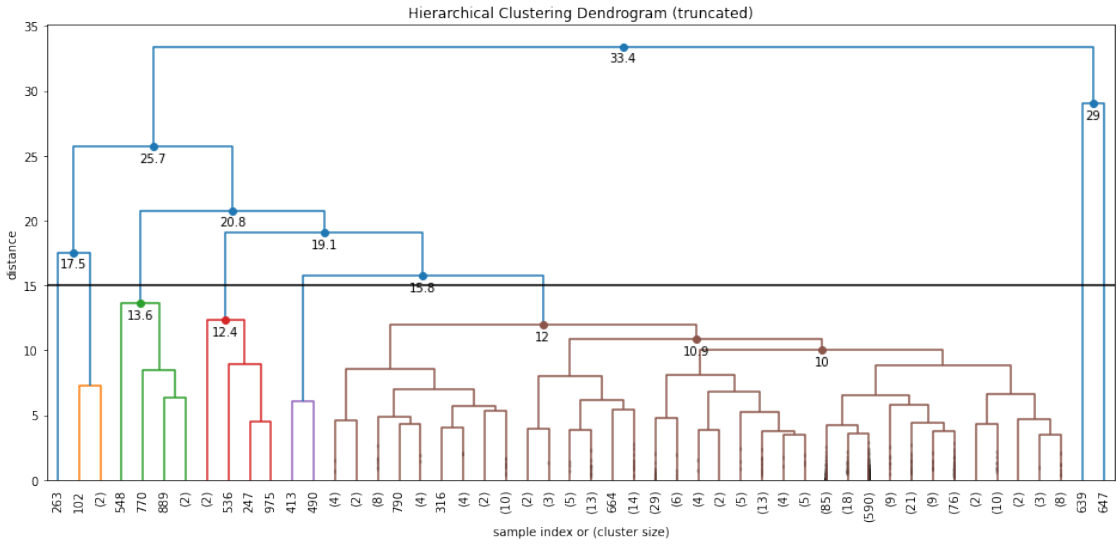


Complete linkage We plot the dendogram from Scipy using complete linkage

[118]:
```
fusions = linkage(X, 'complete')
plt.figure(figsize=(20, 15))
plt.title('Hierarchical clustering dendogram')
plt.xlabel('Data points')
plt.ylabel('Distance')
dendrogram(fusions,
           orientation='top',
           distance_sort='descending',
           show_leaf_counts=True,
            p=50,
          truncate_mode = 'lastp')
plt.show()
```



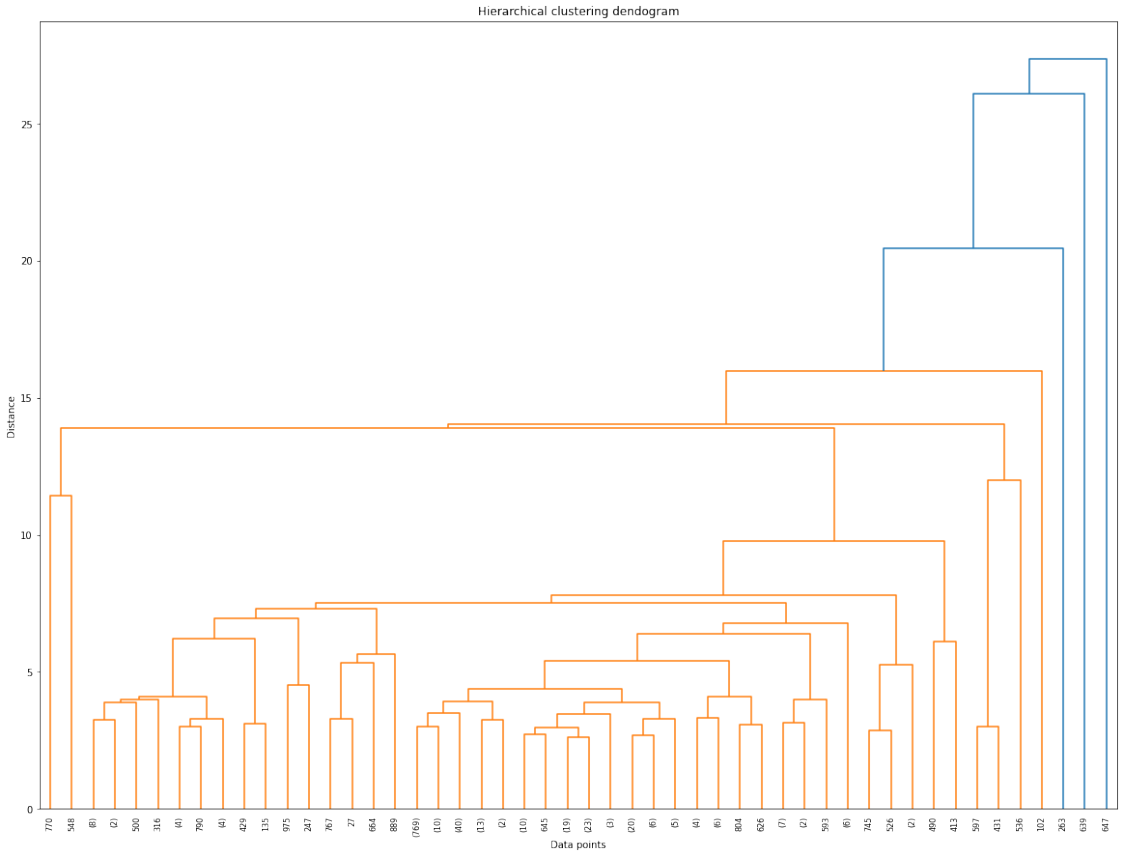
[119]:
```
plt.figure(figsize=(16, 7))
fancy_dendrogram(
    fusions,
    truncate_mode='lastp',
    p=50,
    leaf_rotation=90.,
    leaf_font_size=10.,
    show_contracted=True,
    annotate_above=10,
    max_d=15,  # plot a horizontal cut-off line
)
```



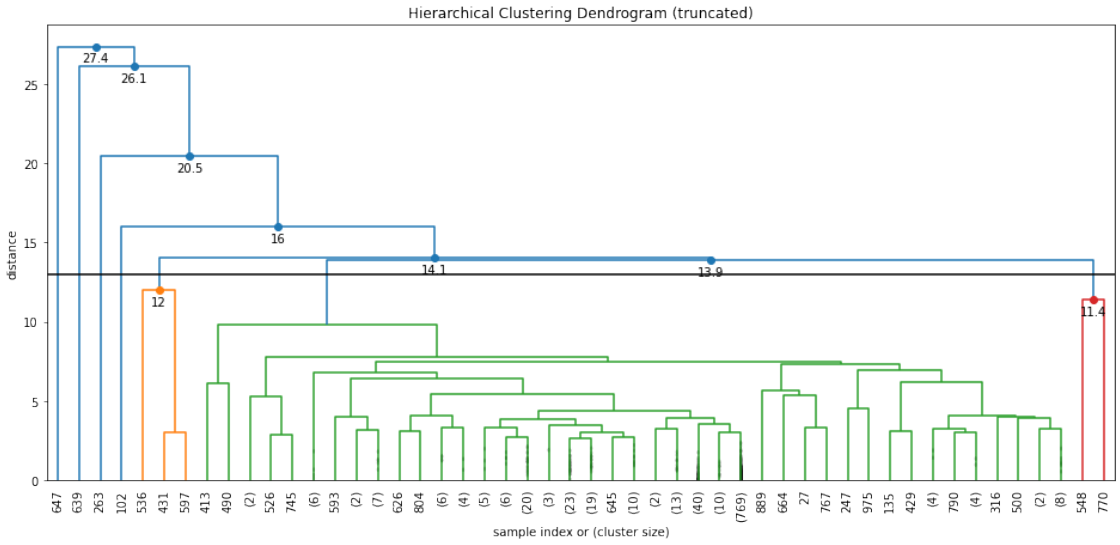
It seems that, using complete, the ideal k = 8, but does not look that good.

Average linkage We plot the dendogram from Scipy using average linkage.

[120]:
```
fusions = linkage(X, 'average')
plt.figure(figsize=(20, 15))
plt.title('Hierarchical clustering dendogram')
plt.xlabel('Data points')
plt.ylabel('Distance')
dendrogram(fusions,
           orientation='top',
           distance_sort='descending',
           show_leaf_counts=True,
            p=50,
          truncate_mode = 'lastp')
plt.show()
```



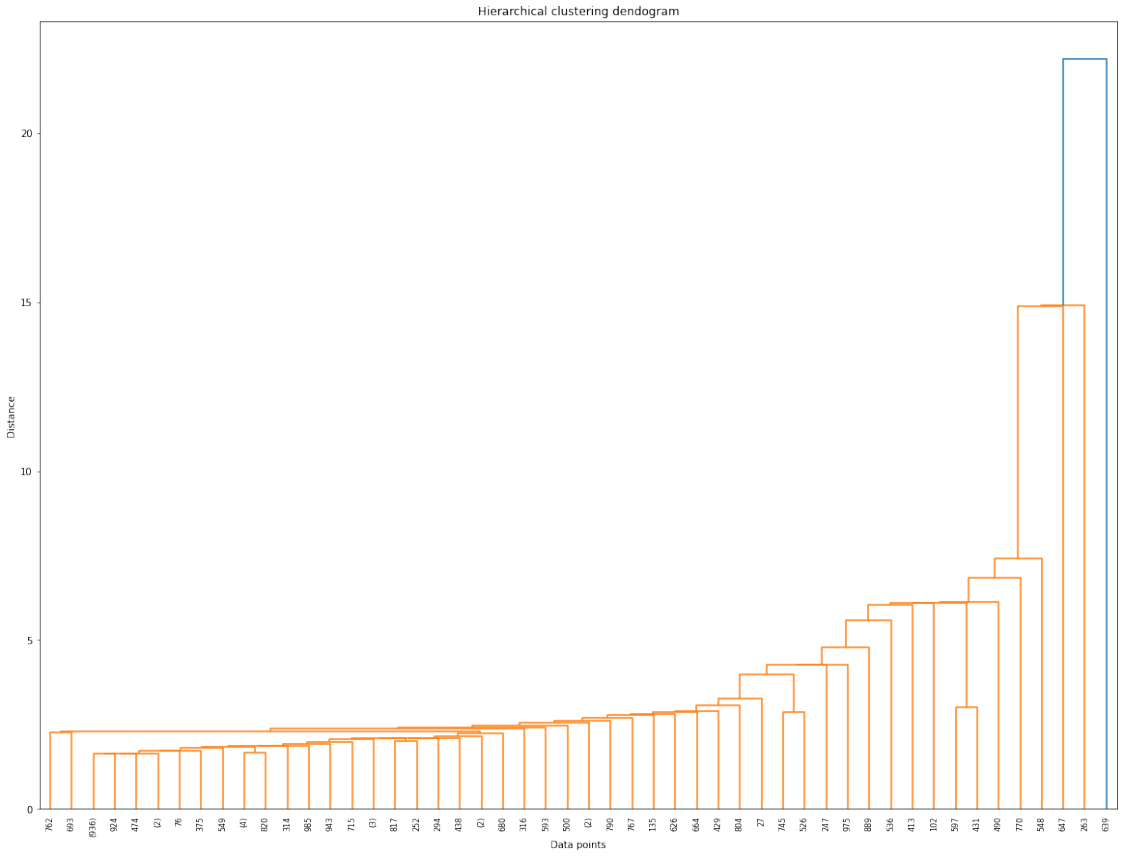
[121]:
```
plt.figure(figsize=(16, 7))
fancy_dendrogram(
    fusions,
    truncate_mode='lastp',
    p=50,
    leaf_rotation=90.,
    leaf_font_size=10.,
    show_contracted=True,
    annotate_above=10,
    max_d=13,  # plot a horizontal cut-off line
)
```



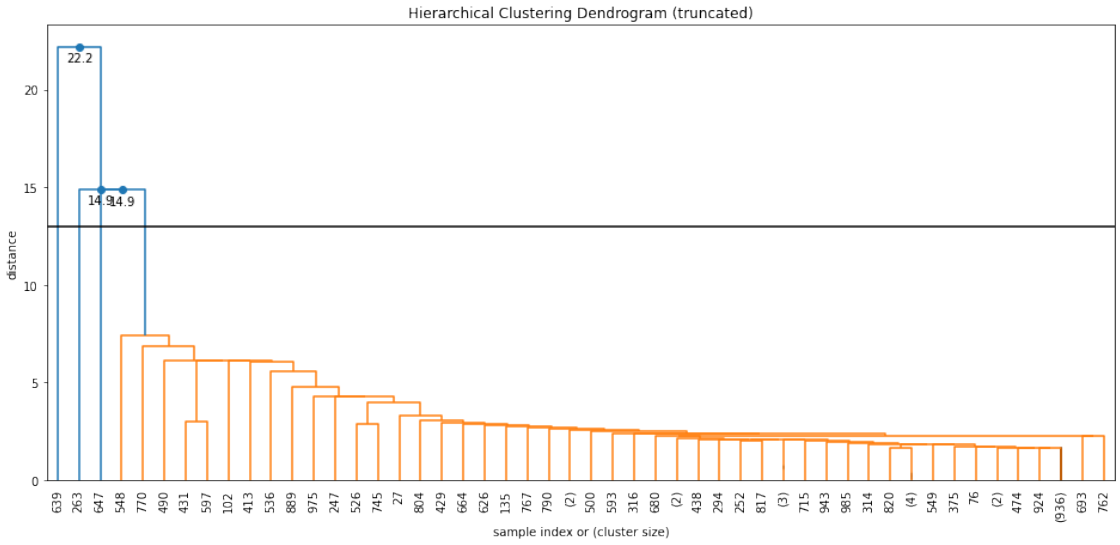
It seems that, using average, the ideal k = 7, but does not look that good.

Single linkage We plot the dendogram from Scipy using single linkage.

[122]:
```
fusions = linkage(X, 'single')
plt.figure(figsize=(20, 15))
plt.title('Hierarchical clustering dendogram')
plt.xlabel('Data points')
plt.ylabel('Distance')
dendrogram(fusions,
           orientation='top',
           distance_sort='descending',
           show_leaf_counts=True,
            p=50,
          truncate_mode = 'lastp')
plt.show()
```



[123]:
```
plt.figure(figsize=(16, 7))
fancy_dendrogram(
    fusions,
    truncate_mode='lastp',
    p=50,
    leaf_rotation=90.,
    leaf_font_size=10.,
    show_contracted=True,
    annotate_above=10,
    max_d=13,  # plot a horizontal cut-off line
)
```



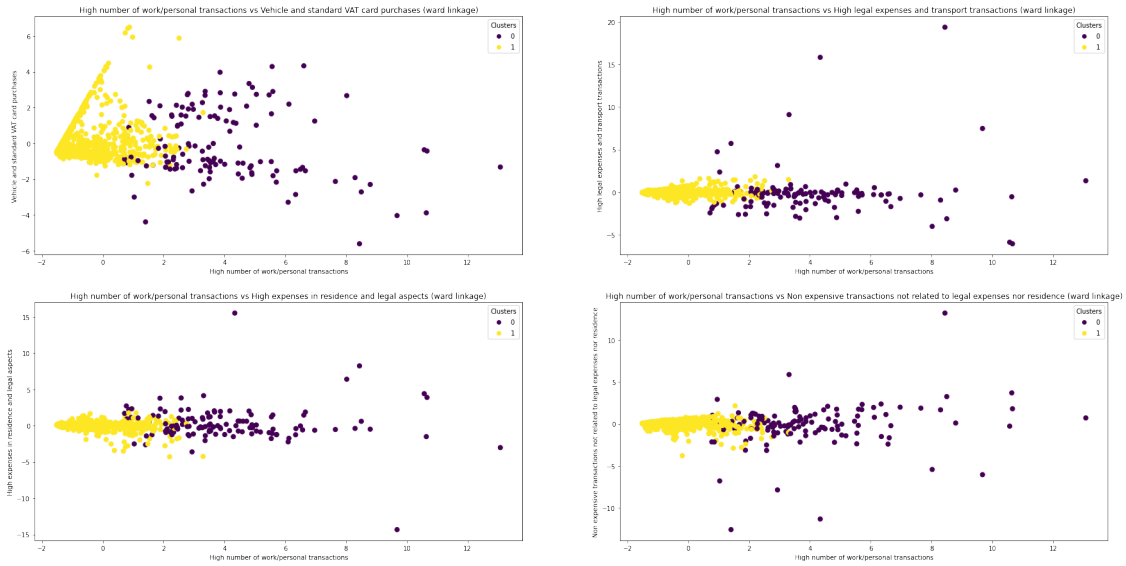
The results do not seem that good even with k=4

Clustering using ward linkage with 2 clusters Since the resulting tree seems balanced, we chose a cutting point and we got 2 clusters.

[124]:
```
fig, axes = plt.subplots(2,2,figsize=(30,15))

link = 'ward'
clustering = AgglomerativeClustering(linkage=link, n_clusters=2)
clustering.fit(X)
points=axes[0][0].scatter(X[:, 0], X[:, 1], c=clustering.labels_, s=50,␣
↪cmap='viridis')
axes[0][0].set_title("High number of work/personal transactions vs Vehicle and␣
↪standard VAT card purchases (%s linkage)" % link)
axes[0][0].set_xlabel('High number of work/personal transactions')
axes[0][0].set_ylabel('Vehicle and standard VAT card purchases')
axes[0][0].legend(*points.legend_elements(), title="Clusters")

points=axes[0][1].scatter(X[:, 0], X[:, 2], c=clustering.labels_, s=50,␣
↪cmap='viridis')
axes[0][1].set_title("High number of work/personal transactions vs High legal␣
↪expenses and transport transactions (%s linkage)" % link)
axes[0][1].set_xlabel('High number of work/personal transactions')
axes[0][1].set_ylabel('High legal expenses and transport transactions')
axes[0][1].legend(*points.legend_elements(), title="Clusters")

points=axes[1][0].scatter(X[:, 0], X[:, 3], c=clustering.labels_, s=50,␣
↪cmap='viridis')
```

```
axes[1][0].set_title("High number of work/personal transactions vs High␣
 ↪expenses in residence and legal aspects (%s linkage)" % link)
axes[1][0].set_xlabel('High number of work/personal transactions')
axes[1][0].set_ylabel('High expenses in residence and legal aspects')
axes[1][0].legend(*points.legend_elements(), title="Clusters")

points=axes[1][1].scatter(X[:, 0], X[:, 4], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[1][1].set_title("High number of work/personal transactions vs Non␣
 ↪expensive transactions not related to legal expenses nor residence (%s␣
 ↪linkage)" % link)
axes[1][1].set_xlabel('High number of work/personal transactions')
axes[1][1].set_ylabel('Non expensive transactions not related to legal expenses␣
 ↪nor residence')
axes[1][1].legend(*points.legend_elements(), title="Clusters")
```

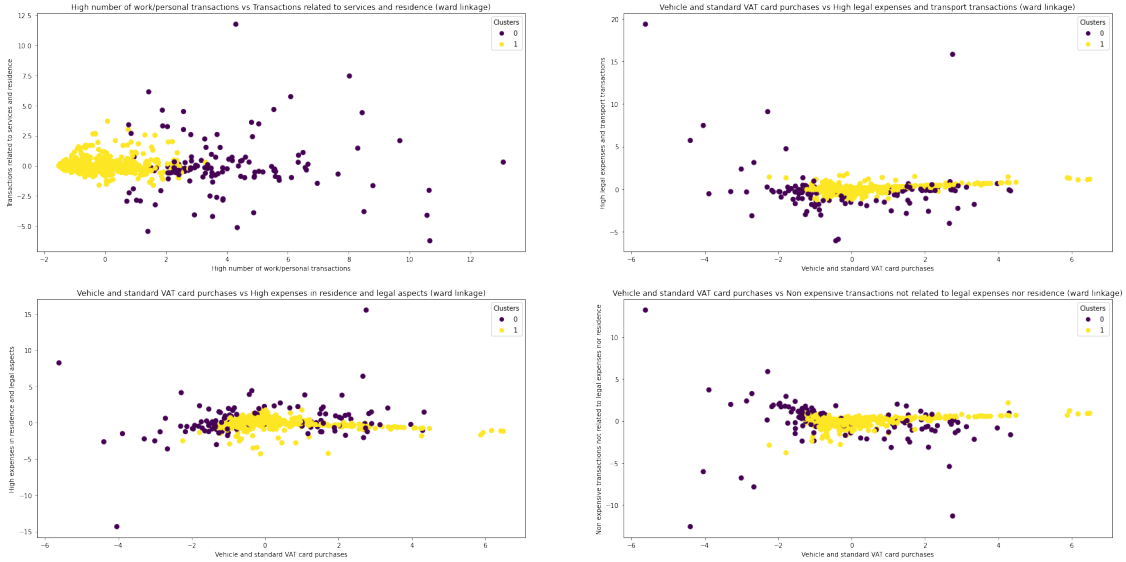
[124]: <matplotlib.legend.Legend at 0x7f9ce4345220>



[125]:
```
fig, axes = plt.subplots(2,2,figsize=(30,15))

points=axes[0][0].scatter(X[:, 0], X[:, 5], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[0][0].set_title("High number of work/personal transactions vs Transactions␣
 ↪related to services and residence (%s linkage)" % link)
axes[0][0].set_xlabel('High number of work/personal transactions')
axes[0][0].set_ylabel('Transactions related to services and residence')
axes[0][0].legend(*points.legend_elements(), title="Clusters")
```

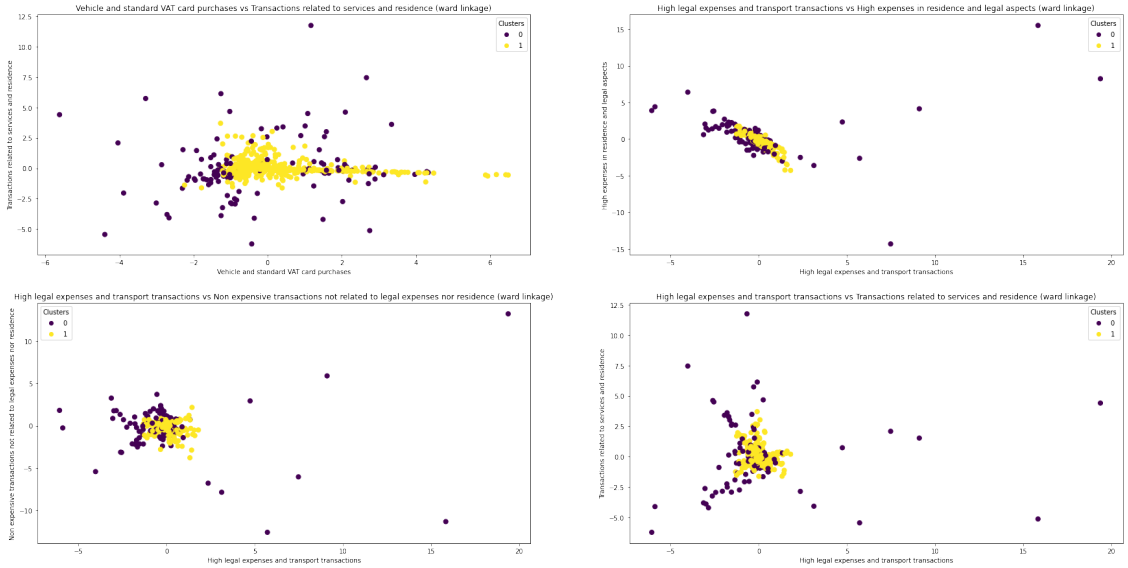
```
points=axes[0][1].scatter(X[:, 1], X[:, 2], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[0][1].set_title("Vehicle and standard VAT card purchases vs High legal␣
 ↪expenses and transport transactions (%s linkage)" % link)
axes[0][1].set_xlabel('Vehicle and standard VAT card purchases')
axes[0][1].set_ylabel('High legal expenses and transport transactions')
axes[0][1].legend(*points.legend_elements(), title="Clusters")

points=axes[1][0].scatter(X[:, 1], X[:, 3], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[1][0].set_title("Vehicle and standard VAT card purchases vs High expenses␣
 ↪in residence and legal aspects (%s linkage)" % link)
axes[1][0].set_xlabel('Vehicle and standard VAT card purchases')
axes[1][0].set_ylabel('High expenses in residence and legal aspects')
axes[1][0].legend(*points.legend_elements(), title="Clusters")

points=axes[1][1].scatter(X[:, 1], X[:, 4], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[1][1].set_title("Vehicle and standard VAT card purchases vs Non expensive␣
 ↪transactions not related to legal expenses nor residence (%s linkage)" %␣
 ↪link)
axes[1][1].set_xlabel('Vehicle and standard VAT card purchases')
axes[1][1].set_ylabel('Non expensive transactions not related to legal expenses␣
 ↪nor residence')
axes[1][1].legend(*points.legend_elements(), title="Clusters")
```

[125]: <matplotlib.legend.Legend at 0x7f9ce53a6fa0>



[126]:
```
fig, axes = plt.subplots(2,2,figsize=(30,15))

points=axes[0][0].scatter(X[:, 1], X[:, 5], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[0][0].set_title("Vehicle and standard VAT card purchases vs Transactions␣
 ↪related to services and residence (%s linkage)" % link)
axes[0][0].set_xlabel('Vehicle and standard VAT card purchases')
axes[0][0].set_ylabel('Transactions related to services and residence')
axes[0][0].legend(*points.legend_elements(), title="Clusters")

points=axes[0][1].scatter(X[:, 2], X[:, 3], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[0][1].set_title("High legal expenses and transport transactions vs High␣
 ↪expenses in residence and legal aspects (%s linkage)" % link)
axes[0][1].set_xlabel('High legal expenses and transport transactions')
axes[0][1].set_ylabel('High expenses in residence and legal aspects')
axes[0][1].legend(*points.legend_elements(), title="Clusters")

points=axes[1][0].scatter(X[:, 2], X[:, 4], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[1][0].set_title("High legal expenses and transport transactions vs Non␣
 ↪expensive transactions not related to legal expenses nor residence (%s␣
 ↪linkage)" % link)
axes[1][0].set_xlabel('High legal expenses and transport transactions')
axes[1][0].set_ylabel('Non expensive transactions not related to legal expenses␣
 ↪nor residence')
axes[1][0].legend(*points.legend_elements(), title="Clusters")

points=axes[1][1].scatter(X[:, 2], X[:, 5], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[1][1].set_title("High legal expenses and transport transactions vs␣
 ↪Transactions related to services and residence (%s linkage)" % link)
axes[1][1].set_xlabel('High legal expenses and transport transactions')
axes[1][1].set_ylabel('Transactions related to services and residence')
axes[1][1].legend(*points.legend_elements(), title="Clusters")
```

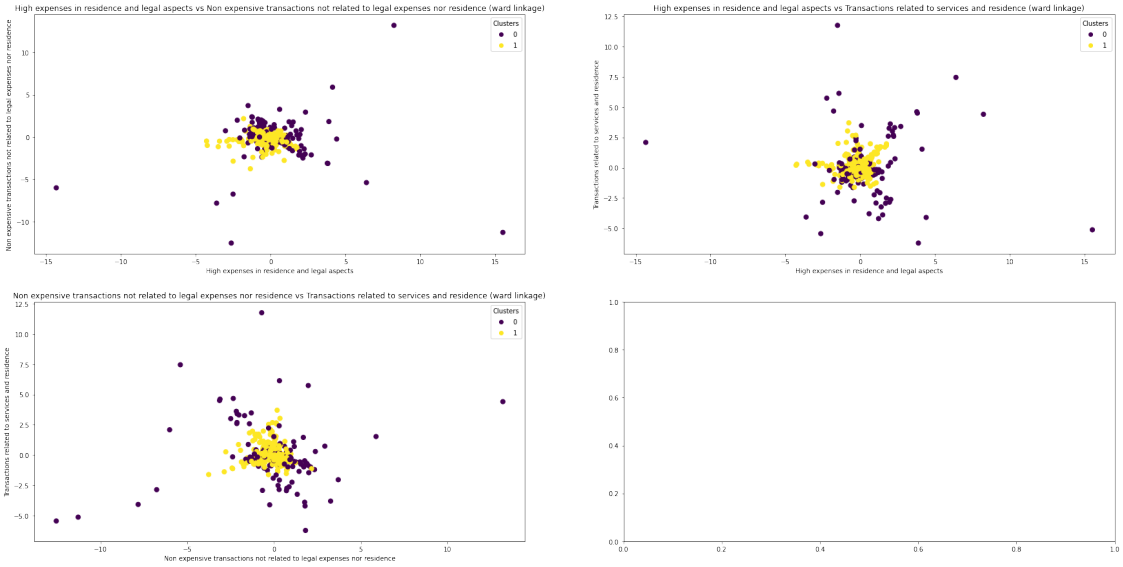
[126]:
```
<matplotlib.legend.Legend at 0x7f9ce2f4e790>
```



[127]:
```
fig, axes = plt.subplots(2,2,figsize=(30,15))

points=axes[0][0].scatter(X[:, 3], X[:, 4], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[0][0].set_title("High expenses in residence and legal aspects vs Non␣
 ↪expensive transactions not related to legal expenses nor residence (%s␣
 ↪linkage)" % link)
axes[0][0].set_xlabel('High expenses in residence and legal aspects')
axes[0][0].set_ylabel('Non expensive transactions not related to legal expenses␣
 ↪nor residence')
axes[0][0].legend(*points.legend_elements(), title="Clusters")

points=axes[0][1].scatter(X[:, 3], X[:, 5], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[0][1].set_title("High expenses in residence and legal aspects vs␣
 ↪Transactions related to services and residence (%s linkage)" % link)
axes[0][1].set_xlabel('High expenses in residence and legal aspects')
axes[0][1].set_ylabel('Transactions related to services and residence')
axes[0][1].legend(*points.legend_elements(), title="Clusters")

points=axes[1][0].scatter(X[:, 4], X[:, 5], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[1][0].set_title("Non expensive transactions not related to legal expenses␣
 ↪nor residence vs Transactions related to services and residence (%s␣
 ↪linkage)" % link)
axes[1][0].set_xlabel('Non expensive transactions not related to legal expenses␣
 ↪nor residence')
axes[1][0].set_ylabel('Transactions related to services and residence')
```

```
axes[1][0].legend(*points.legend_elements(), title="Clusters")
```

[127]: <matplotlib.legend.Legend at 0x7f9ce44c3f10>



We save the clustering object.

[128]:
```
df2014_hc2_pkl_filename = '/Users/andresaristi/Documents/
 ↪BirminghamCardTransactions/pickle_files/df2014_hc2.pkl'
df2014_hc2_pkl = open(df2014_hc2_pkl_filename, 'wb')
pickle.dump(clustering, df2014_hc2_pkl)
df2014_hc2_pkl.close()
```

Population per cluster:

[129]:
```
counter = Counter(clustering.labels_)
counter
```

[129]: Counter({1: 865, 0: 128})

Profiling clients with hierarchical clustering with 2 clusters By observing the scatter plots we will try to profile the clients, although 2 clusters seem too little.

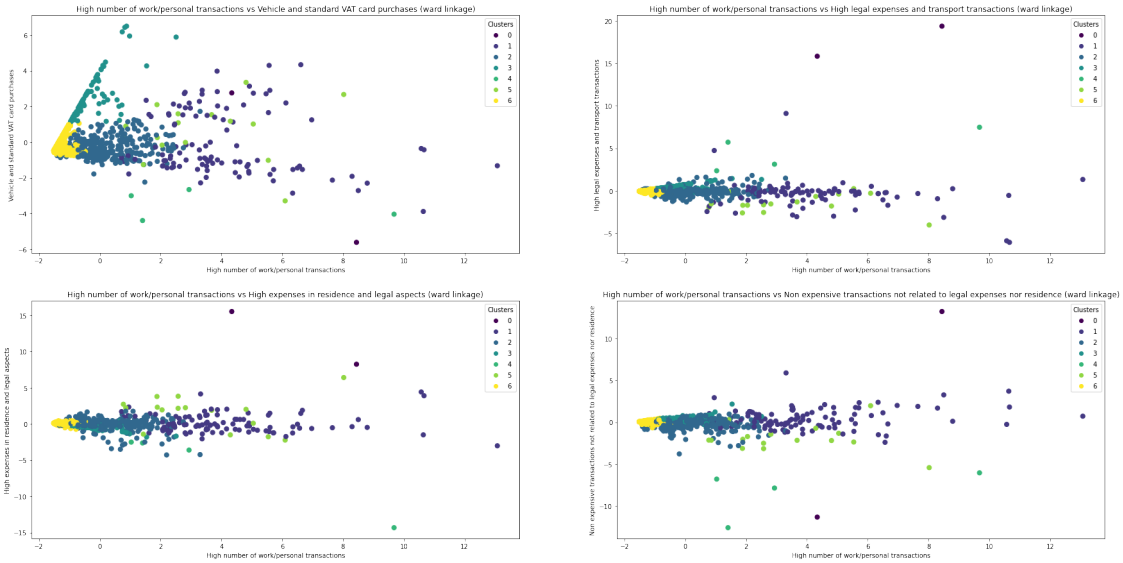
Cluster 0: scattered or spread out cluster. Clients from low up to high High number of work/personal transactions (scattered area), low to more than average Vehicle and standard VAT card purchases (scattered area), low High legal expenses and transport transactions (few of the clients have high values), average in High expenses in residence and legal aspects, average in Non expensive transactions not related to legal expenses nor residence and low to high in Transactions related to services and residence (scattered area).

Cluster 1: sense cluster. Clients with the lowest High number of work/personal transactions, average to high Vehicle and standard VAT card purchases, low High legal expenses and transport transactions, average in High expenses in residence and legal aspects, average in Non expensive transactions not related to legal expenses nor residence and less than average in Transactions related to services and residence.

Clustering using ward linkage with k = 7

[130]:
```
fig, axes = plt.subplots(2,2,figsize=(30,15))

link = 'ward'
clustering = AgglomerativeClustering(linkage=link, n_clusters=7)
clustering.fit(X)
points=axes[0][0].scatter(X[:, 0], X[:, 1], c=clustering.labels_, s=50,␣
↪cmap='viridis')
axes[0][0].set_title("High number of work/personal transactions vs Vehicle and␣
↪standard VAT card purchases (%s linkage)" % link)
axes[0][0].set_xlabel('High number of work/personal transactions')
axes[0][0].set_ylabel('Vehicle and standard VAT card purchases')
axes[0][0].legend(*points.legend_elements(), title="Clusters")

points=axes[0][1].scatter(X[:, 0], X[:, 2], c=clustering.labels_, s=50,␣
↪cmap='viridis')
axes[0][1].set_title("High number of work/personal transactions vs High legal␣
↪expenses and transport transactions (%s linkage)" % link)
axes[0][1].set_xlabel('High number of work/personal transactions')
axes[0][1].set_ylabel('High legal expenses and transport transactions')
axes[0][1].legend(*points.legend_elements(), title="Clusters")

points=axes[1][0].scatter(X[:, 0], X[:, 3], c=clustering.labels_, s=50,␣
↪cmap='viridis')
axes[1][0].set_title("High number of work/personal transactions vs High␣
↪expenses in residence and legal aspects (%s linkage)" % link)
axes[1][0].set_xlabel('High number of work/personal transactions')
axes[1][0].set_ylabel('High expenses in residence and legal aspects')
axes[1][0].legend(*points.legend_elements(), title="Clusters")

points=axes[1][1].scatter(X[:, 0], X[:, 4], c=clustering.labels_, s=50,␣
↪cmap='viridis')
axes[1][1].set_title("High number of work/personal transactions vs Non␣
↪expensive transactions not related to legal expenses nor residence (%s␣
↪linkage)" % link)
axes[1][1].set_xlabel('High number of work/personal transactions')
axes[1][1].set_ylabel('Non expensive transactions not related to legal expenses␣
↪nor residence')
axes[1][1].legend(*points.legend_elements(), title="Clusters")
```

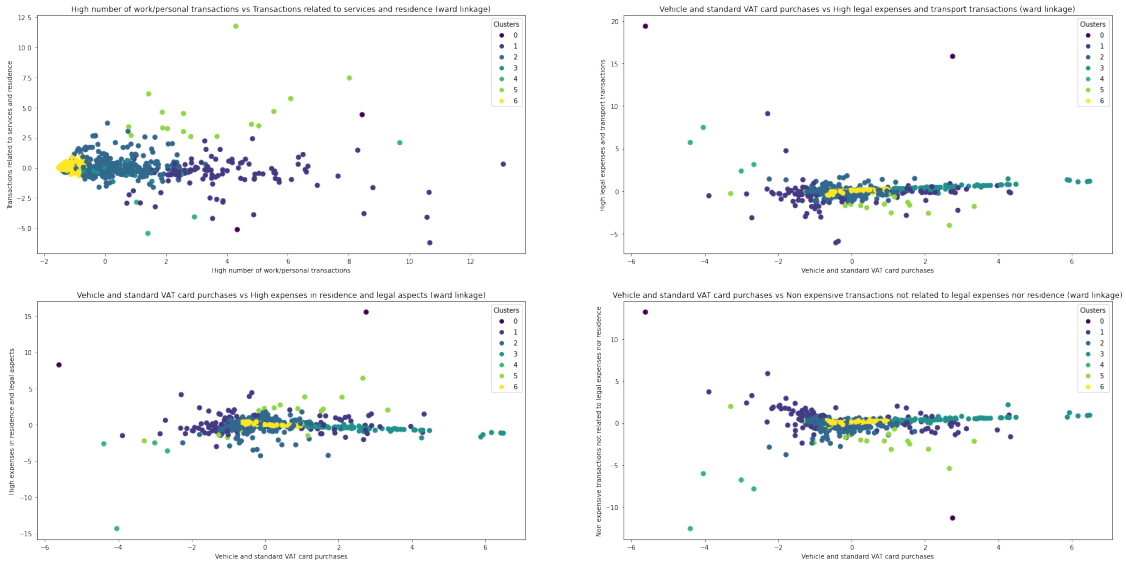
[130]: <matplotlib.legend.Legend at 0x7f9ce5b99790>



[131]:
```
fig, axes = plt.subplots(2,2,figsize=(30,15))

points=axes[0][0].scatter(X[:, 0], X[:, 5], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[0][0].set_title("High number of work/personal transactions vs Transactions␣
 ↪related to services and residence (%s linkage)" % link)
axes[0][0].set_xlabel('High number of work/personal transactions')
axes[0][0].set_ylabel('Transactions related to services and residence')
axes[0][0].legend(*points.legend_elements(), title="Clusters")

points=axes[0][1].scatter(X[:, 1], X[:, 2], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[0][1].set_title("Vehicle and standard VAT card purchases vs High legal␣
 ↪expenses and transport transactions (%s linkage)" % link)
axes[0][1].set_xlabel('Vehicle and standard VAT card purchases')
axes[0][1].set_ylabel('High legal expenses and transport transactions')
axes[0][1].legend(*points.legend_elements(), title="Clusters")

points=axes[1][0].scatter(X[:, 1], X[:, 3], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[1][0].set_title("Vehicle and standard VAT card purchases vs High expenses␣
 ↪in residence and legal aspects (%s linkage)" % link)
axes[1][0].set_xlabel('Vehicle and standard VAT card purchases')
axes[1][0].set_ylabel('High expenses in residence and legal aspects')
axes[1][0].legend(*points.legend_elements(), title="Clusters")
```

```
points=axes[1][1].scatter(X[:, 1], X[:, 4], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[1][1].set_title("Vehicle and standard VAT card purchases vs Non expensive␣
 ↪transactions not related to legal expenses nor residence (%s linkage)" %␣
 ↪link)
axes[1][1].set_xlabel('Vehicle and standard VAT card purchases')
axes[1][1].set_ylabel('Non expensive transactions not related to legal expenses␣
 ↪nor residence')
axes[1][1].legend(*points.legend_elements(), title="Clusters")
```

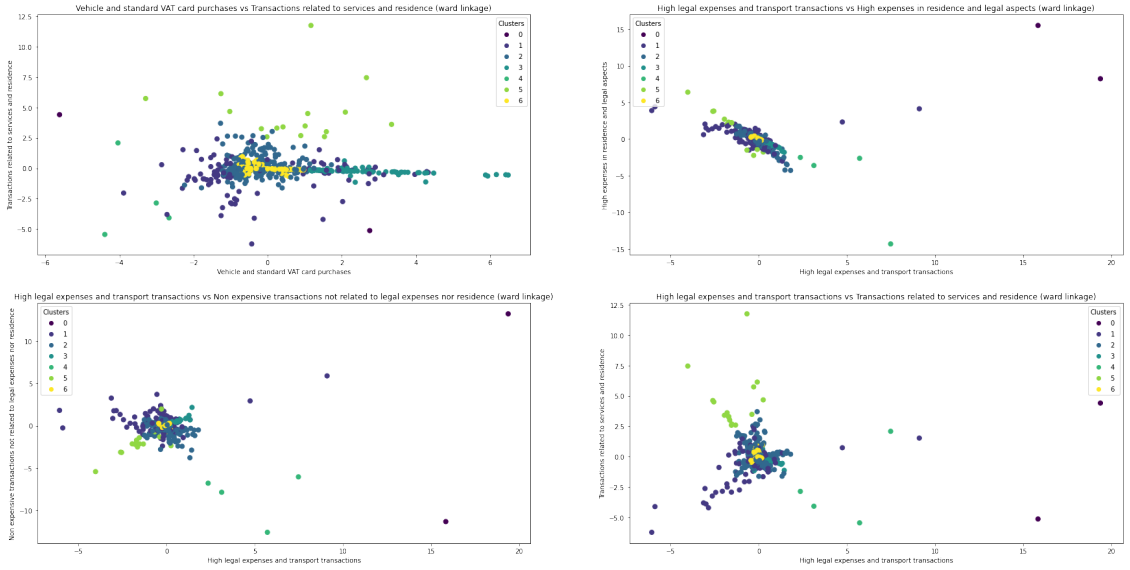
[131]: <matplotlib.legend.Legend at 0x7f9ce55f1f10>



[132]:
```
fig, axes = plt.subplots(2,2,figsize=(30,15))

points=axes[0][0].scatter(X[:, 1], X[:, 5], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[0][0].set_title("Vehicle and standard VAT card purchases vs Transactions␣
 ↪related to services and residence (%s linkage)" % link)
axes[0][0].set_xlabel('Vehicle and standard VAT card purchases')
axes[0][0].set_ylabel('Transactions related to services and residence')
axes[0][0].legend(*points.legend_elements(), title="Clusters")

points=axes[0][1].scatter(X[:, 2], X[:, 3], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[0][1].set_title("High legal expenses and transport transactions vs High␣
 ↪expenses in residence and legal aspects (%s linkage)" % link)
axes[0][1].set_xlabel('High legal expenses and transport transactions')
axes[0][1].set_ylabel('High expenses in residence and legal aspects')
```

```
axes[0][1].legend(*points.legend_elements(), title="Clusters")

points=axes[1][0].scatter(X[:, 2], X[:, 4], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[1][0].set_title("High legal expenses and transport transactions vs Non␣
 ↪expensive transactions not related to legal expenses nor residence (%s␣
 ↪linkage)" % link)
axes[1][0].set_xlabel('High legal expenses and transport transactions')
axes[1][0].set_ylabel('Non expensive transactions not related to legal expenses␣
 ↪nor residence')
axes[1][0].legend(*points.legend_elements(), title="Clusters")

points=axes[1][1].scatter(X[:, 2], X[:, 5], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[1][1].set_title("High legal expenses and transport transactions vs␣
 ↪Transactions related to services and residence (%s linkage)" % link)
axes[1][1].set_xlabel('High legal expenses and transport transactions')
axes[1][1].set_ylabel('Transactions related to services and residence')
axes[1][1].legend(*points.legend_elements(), title="Clusters")
```

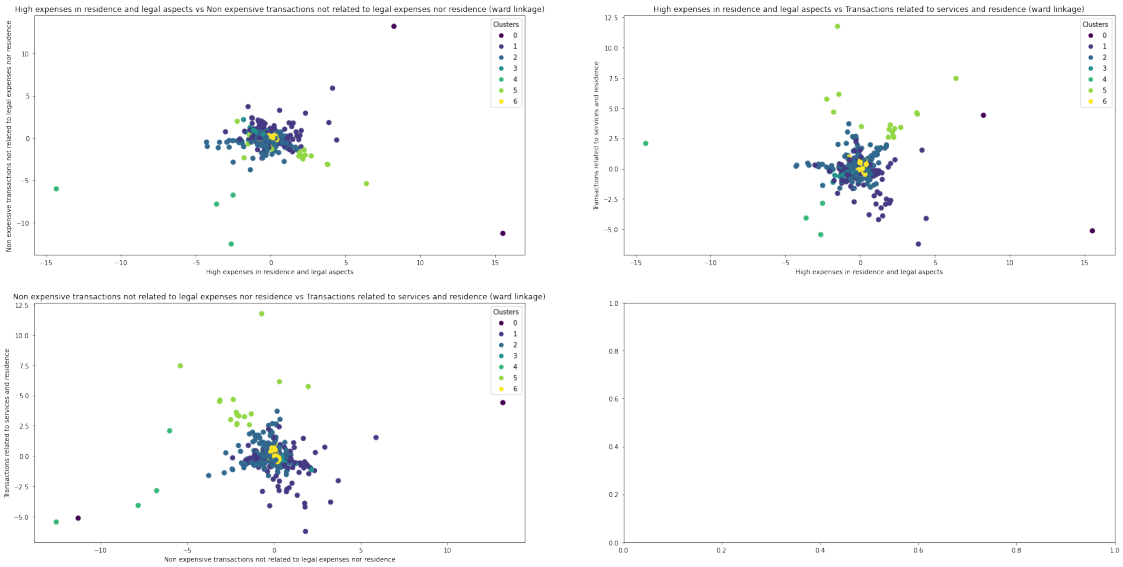
[132]: <matplotlib.legend.Legend at 0x7f9ce509b340>



[133]:
```
fig, axes = plt.subplots(2,2,figsize=(30,15))

points=axes[0][0].scatter(X[:, 3], X[:, 4], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
```

```
axes[0][0].set_title("High expenses in residence and legal aspects vs Non␣
 ↪expensive transactions not related to legal expenses nor residence (%s␣
 ↪linkage)" % link)
axes[0][0].set_xlabel('High expenses in residence and legal aspects')
axes[0][0].set_ylabel('Non expensive transactions not related to legal expenses␣
 ↪nor residence')
axes[0][0].legend(*points.legend_elements(), title="Clusters")

points=axes[0][1].scatter(X[:, 3], X[:, 5], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[0][1].set_title("High expenses in residence and legal aspects vs␣
 ↪Transactions related to services and residence (%s linkage)" % link)
axes[0][1].set_xlabel('High expenses in residence and legal aspects')
axes[0][1].set_ylabel('Transactions related to services and residence')
axes[0][1].legend(*points.legend_elements(), title="Clusters")

points=axes[1][0].scatter(X[:, 4], X[:, 5], c=clustering.labels_, s=50,␣
 ↪cmap='viridis')
axes[1][0].set_title("Non expensive transactions not related to legal expenses␣
 ↪nor residence vs Transactions related to services and residence (%s␣
 ↪linkage)" % link)
axes[1][0].set_xlabel('Non expensive transactions not related to legal expenses␣
 ↪nor residence')
axes[1][0].set_ylabel('Transactions related to services and residence')
axes[1][0].legend(*points.legend_elements(), title="Clusters")
```

[133]: <matplotlib.legend.Legend at 0x7f9ce5253c10>



We save the clustering object.

[134]:
```
df2014_hc7_pkl_filename = '/Users/andresaristi/Documents/
 ↪BirminghamCardTransactions/pickle_files/df2014_hc7.pkl'
df2014_hc7_pkl = open(df2014_hc7_pkl_filename, 'wb')
pickle.dump(clustering, df2014_hc7_pkl)
df2014_hc7_pkl.close()
```

Population per cluster:

[135]:
```
counter = Counter(clustering.labels_)
counter
```

[135]:
```
Counter({6: 464, 1: 106, 2: 306, 3: 95, 5: 16, 4: 4, 0: 2})
```

Profiling clients with hierarchical clustering with 7 clusters By observing the scatter plots we will try to figure out the profile of clients, although some of the plots are not really useful. Clusters 4 and 0 have to few observations to interpret.

Cluster 0: not enough data points to interpret.

Cluster 1: clients with average High number of work/personal transactions (with some clients having really high values), average Vehicle and standard VAT card purchases, low High legal expenses and transport transactions (few of the clients have high values), average in High expenses in residence and legal aspects, average in Non expensive transactions not related to legal expenses nor residence and less than average in Transactions related to services and residence.

Cluster 2: clients with the lowest High number of work/personal transactions, average in Vehicle and standard VAT card purchases, lower than average in High legal expenses and transport transactions, average in High expenses in residence and legal aspects, average in Non expensive transactions not related to legal expenses nor residence and less than average in Transactions related to services and residence.

Cluster 3: clients with the lowest High number of work/personal transactions, the highest Vehicle and standard VAT card purchases, the lowest High legal expenses and transport transactions, average in High expenses in residence and legal aspects, average in Non expensive transactions not related to legal expenses nor residence and lower than average in Transactions related to services and residence.

Cluster 4: not enough data points to interpret.

Cluster 5: clients with less than average in High number of work/personal transactions, an average Vehicle and standard VAT card purchases, the lowest High legal expenses and transport transactions, average in High expenses in residence and legal aspects, average in Non expensive transactions not related to legal expenses nor residence and few Transactions related to services and residence.

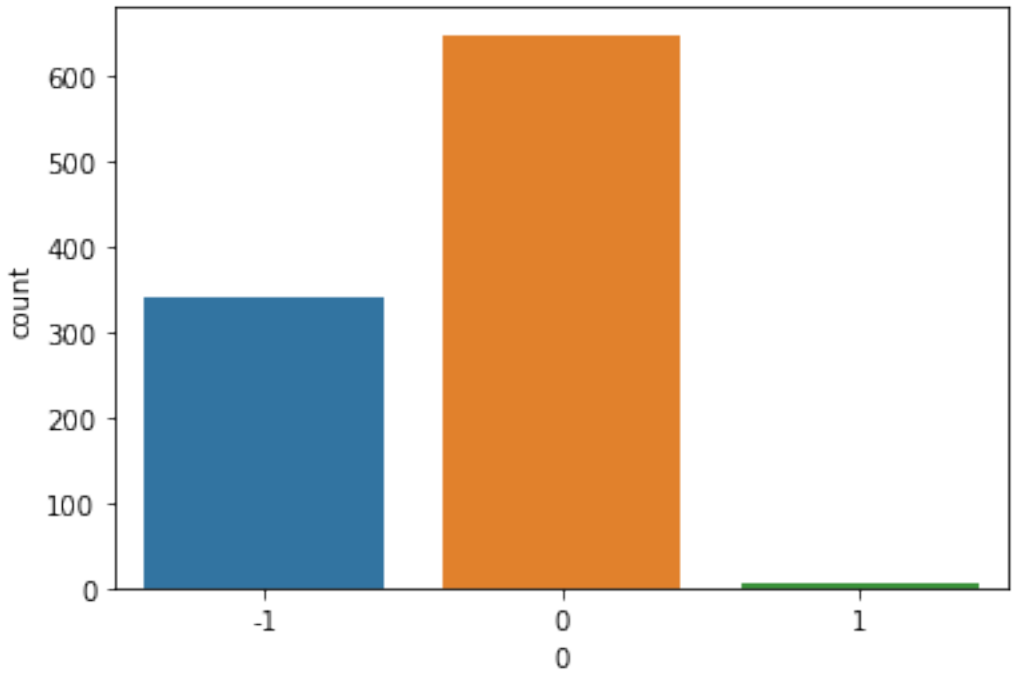
Cluster 6: clients with average High number of work/personal transactions, more than average Vehicle and standard VAT card purchases, the lowest High legal expenses and transport transactions,

high in High expenses in residence and legal aspects, less than average in Non expensive transactions not related to legal expenses nor residence and high on Transactions related to services and residence.

DBSCAN We will be using DBSCAN as another clustering techinque to find the best groups of clients, in order for us to profile them.

[136]:
```
data2014_db = data2014_kmeans.copy()
min_points = 5
eps = 0.45
model = DBSCAN(min_samples = min_points, eps=eps)
model.fit(X)
clusters = pd.DataFrame(model.fit_predict(X))
data2014_db['Cluster'] = clusters.values
num_clusters = len(set(model.labels_))-1
num_clusters
clusters
sns.countplot(x=0,data=clusters)
```

[136]:
```
<AxesSubplot:xlabel='0', ylabel='count'>
```



We found out that there were two clusters plus the noise. We made a plot bar with those results.

[137]:
```
data2014_db.Cluster.value_counts()
data2014_db
```

[137]:

CARD NUMBER	High number of work/personal transactions \
************0007	-1.102613
************0015	-1.510223
************0040	-1.279980
************0047	3.035147
************0057	-1.244730
…	…
************9957	1.126939
************9963	-1.360561
************9968	-1.224845
************9971	0.046710
************9989	3.667547

CARD NUMBER	Vehicle and standard VAT card purchases \
************0007	-0.609923
************0015	-0.474573
************0040	0.171561
************0047	-0.145967
************0057	-0.104075
…	…
************9957	1.249133
************9963	-0.450382
************9968	0.199069
************9971	-0.869348
************9989	-0.836873

CARD NUMBER	High legal expenses and transport transactions \
************0007	-0.078921
************0015	-0.049181
************0040	0.070341
************0047	-0.586113
************0057	-0.108621
…	…
************9957	-0.095804
************9963	-0.067909
************9968	0.106738
************9971	0.040100
************9989	-3.051555

CARD NUMBER	High expenses in residence and legal aspects \
************0007	-0.002634
************0015	0.109719
************0040	-0.017045

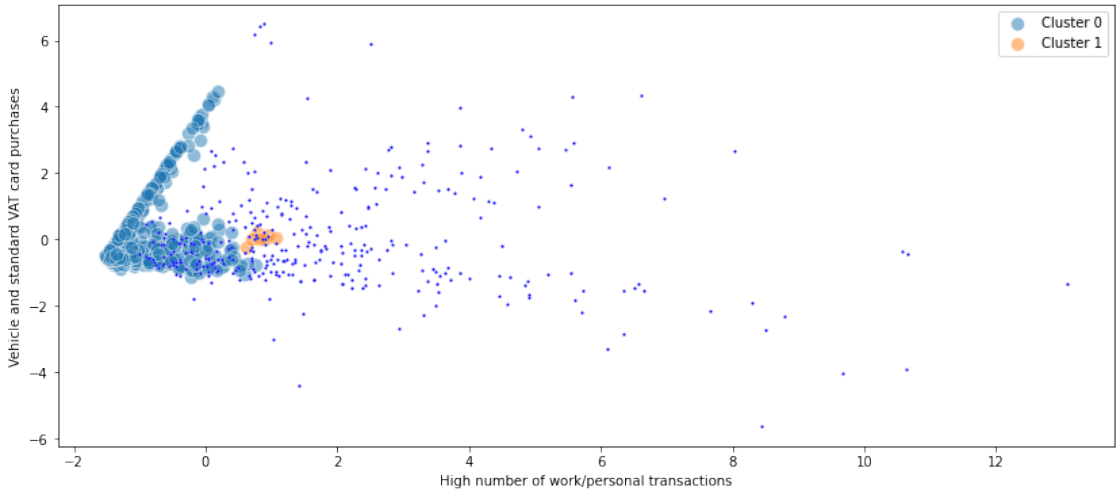
```
************0047                                          0.532373
************0057                                          0.089445
…                                                            …
************9957                                          0.050369
************9963                                          0.153903
************9968                                         -0.070366
************9971                                         -0.278259
************9989                                          2.046489

                 Non expensive transactions not related to legal expenses nor
residence  \
CARD NUMBER
************0007                                            0.090542
************0015                                            0.067047
************0040                                            0.157104
************0047                                           -0.254677
************0057                                            0.235960
…                                                              …
************9957                                           -0.173236
************9963                                            0.024214
************9968                                            0.219315
************9971                                            0.110072
************9989                                            0.870602

                 Transactions related to services and residence  Cluster
CARD NUMBER
************0007                                          0.506741        0
************0015                                         -0.008816        0
************0040                                         -0.063487        0
************0047                                         -0.018432       -1
************0057                                         -0.274399        0
…                                                            …       …
************9957                                          0.005066       -1
************9963                                         -0.012115        0
************9968                                         -0.085066        0
************9971                                         -0.316886        0
************9989                                         -2.621232       -1

[993 rows x 7 columns]
```

We save the dataframe resulting from the DBSCAN

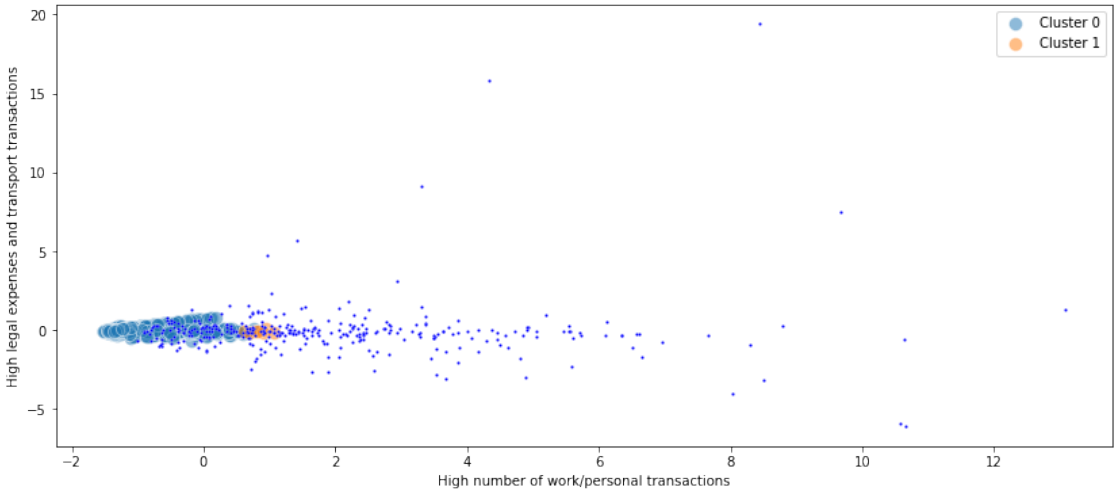
[138]:
```
df2014_db_pkl_filename = '/Users/andresaristi/Documents/
 ↪BirminghamCardTransactions/pickle_files/df2014_db.pkl'
df2014_db_pkl = open(df2014_db_pkl_filename, 'wb')
pickle.dump(data2014_db, df2014_db_pkl)
df2014_db_pkl.close()
```

We start using scatter plots taking into account the PCs in order to characterize our segments.

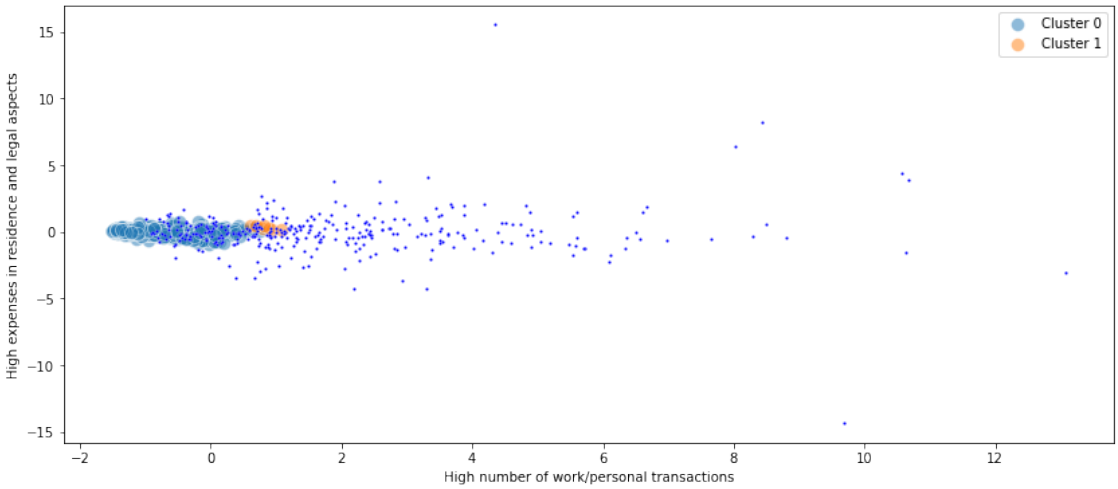
[139]:
```
fig = plt.figure(figsize=(14,6))

for cluster in range(0,num_clusters):
    sns.scatterplot(x="High number of work/personal transactions", y="Vehicle␣
 ↪and standard VAT card purchases", s=100, alpha=0.5,
                    data=data2014_db.loc[data2014_db.Cluster==cluster,:])
sns.scatterplot(x="High number of work/personal transactions", y="Vehicle and␣
 ↪standard VAT card purchases", s=5, color = 'b',
                    data=data2014_db.loc[data2014_db.Cluster==-1,:])
plt.legend(['Cluster 0', 'Cluster 1'])
plt.show()
```



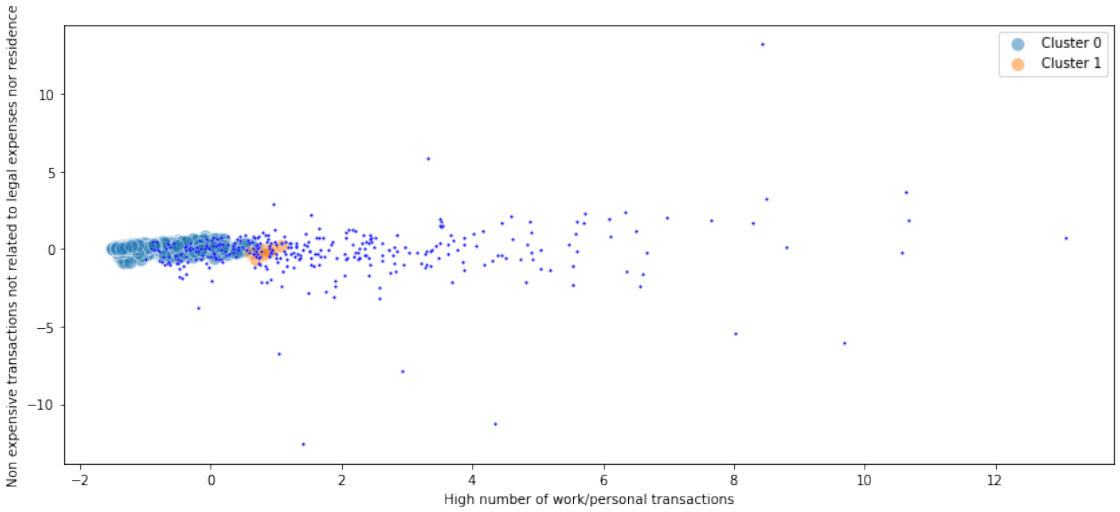
[140]:
```
fig = plt.figure(figsize=(14,6))

for cluster in range(0,num_clusters):
    sns.scatterplot(x="High number of work/personal transactions", y="High␣
 ↪legal expenses and transport transactions", s=100, alpha=0.5,
                    data=data2014_db.loc[data2014_db.Cluster==cluster,:])
sns.scatterplot(x="High number of work/personal transactions", y="High legal␣
 ↪expenses and transport transactions", s=5, color = 'b',
                    data=data2014_db.loc[data2014_db.Cluster==-1,:])
plt.legend(['Cluster 0', 'Cluster 1'])
plt.show()
```



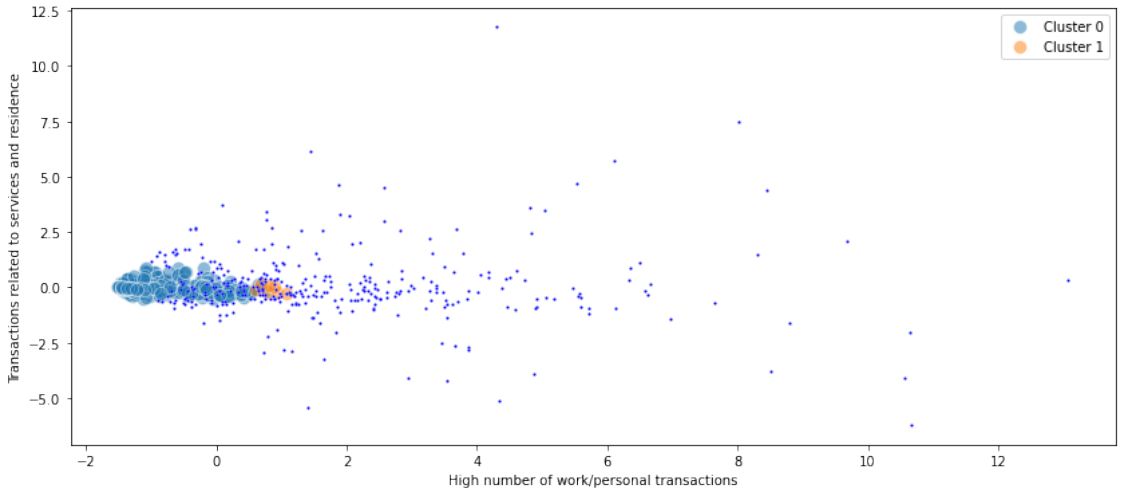
[141]:
```
fig = plt.figure(figsize=(14,6))

for cluster in range(0,num_clusters):
    sns.scatterplot(x="High number of work/personal transactions", y="High␣
 ↪expenses in residence and legal aspects", s=100, alpha=0.5,
                    data=data2014_db.loc[data2014_db.Cluster==cluster,:])
sns.scatterplot(x="High number of work/personal transactions", y="High expenses␣
 ↪in residence and legal aspects", s=5, color = 'b',
                    data=data2014_db.loc[data2014_db.Cluster==-1,:])
plt.legend(['Cluster 0', 'Cluster 1'])
plt.show()
```



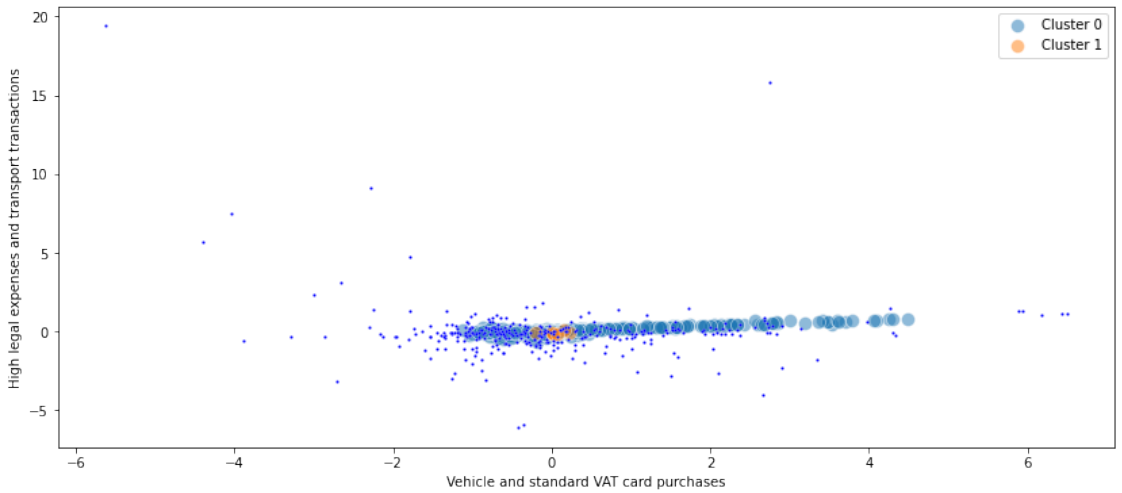
[142]:
```
fig = plt.figure(figsize=(14,6))

for cluster in range(0,num_clusters):
    sns.scatterplot(x="High number of work/personal transactions", y="Non␣
 ↪expensive transactions not related to legal expenses nor residence", s=100,␣
 ↪alpha=0.5,
                    data=data2014_db.loc[data2014_db.Cluster==cluster,:])
sns.scatterplot(x="High number of work/personal transactions", y="Non expensive␣
 ↪transactions not related to legal expenses nor residence", s=5, color = 'b',
                    data=data2014_db.loc[data2014_db.Cluster==-1,:])
plt.legend(['Cluster 0', 'Cluster 1'])
plt.show()
```



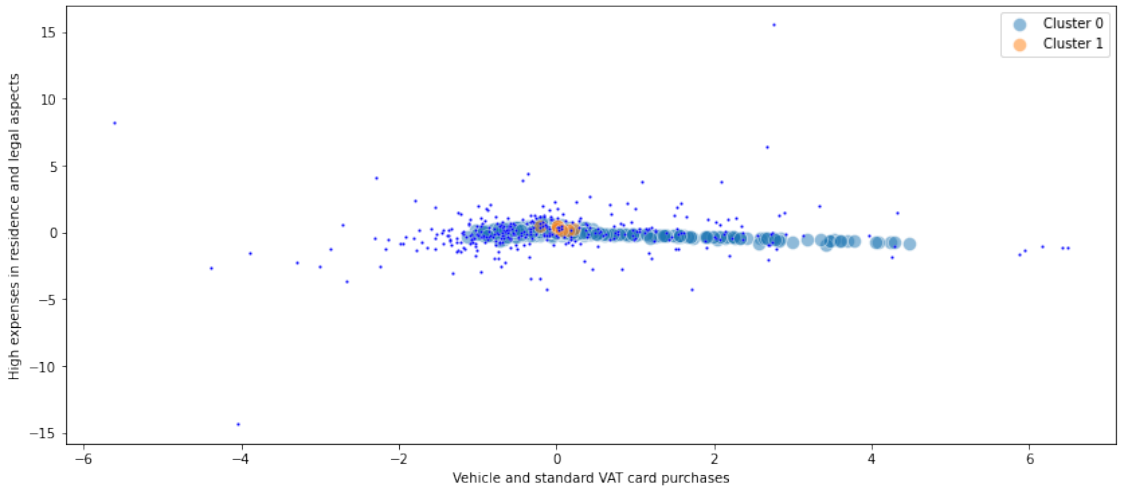
[143]:
```
fig = plt.figure(figsize=(14,6))

for cluster in range(0,num_clusters):
    sns.scatterplot(x="High number of work/personal transactions",␣
 ↪y="Transactions related to services and residence", s=100, alpha=0.5,
                    data=data2014_db.loc[data2014_db.Cluster==cluster,:])
sns.scatterplot(x="High number of work/personal transactions", y="Transactions␣
 ↪related to services and residence", s=5, color = 'b',
                    data=data2014_db.loc[data2014_db.Cluster==-1,:])
plt.legend(['Cluster 0', 'Cluster 1'])
plt.show()
```



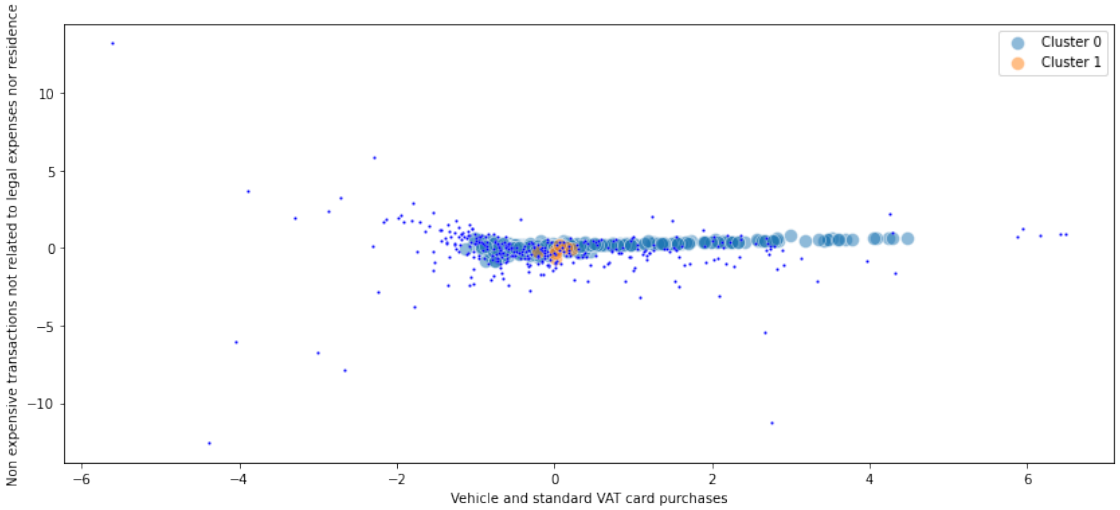
[144]:
```
fig = plt.figure(figsize=(14,6))

for cluster in range(0,num_clusters):
    sns.scatterplot(x="Vehicle and standard VAT card purchases", y="High legal␣
 ↪expenses and transport transactions", s=100, alpha=0.5,
                    data=data2014_db.loc[data2014_db.Cluster==cluster,:])
sns.scatterplot(x="Vehicle and standard VAT card purchases", y="High legal␣
 ↪expenses and transport transactions", s=5, color = 'b',
                    data=data2014_db.loc[data2014_db.Cluster==-1,:])
plt.legend(['Cluster 0', 'Cluster 1'])
plt.show()
```



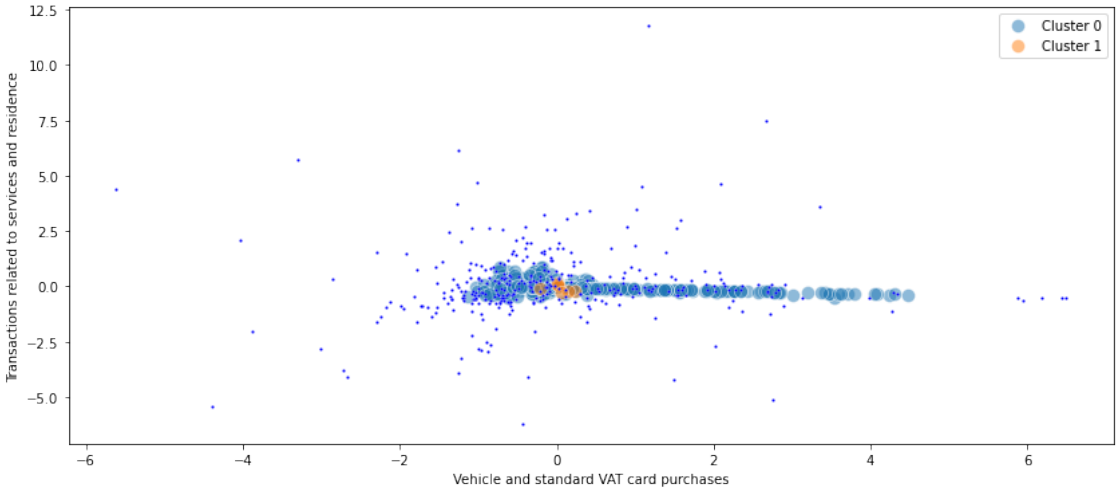
[145]:
```
fig = plt.figure(figsize=(14,6))

for cluster in range(0,num_clusters):
    sns.scatterplot(x="Vehicle and standard VAT card purchases", y="High␣
 ↪expenses in residence and legal aspects", s=100, alpha=0.5,
                    data=data2014_db.loc[data2014_db.Cluster==cluster,:])
sns.scatterplot(x="Vehicle and standard VAT card purchases", y="High expenses␣
 ↪in residence and legal aspects", s=5, color = 'b',
                    data=data2014_db.loc[data2014_db.Cluster==-1,:])
plt.legend(['Cluster 0', 'Cluster 1'])
plt.show()
```



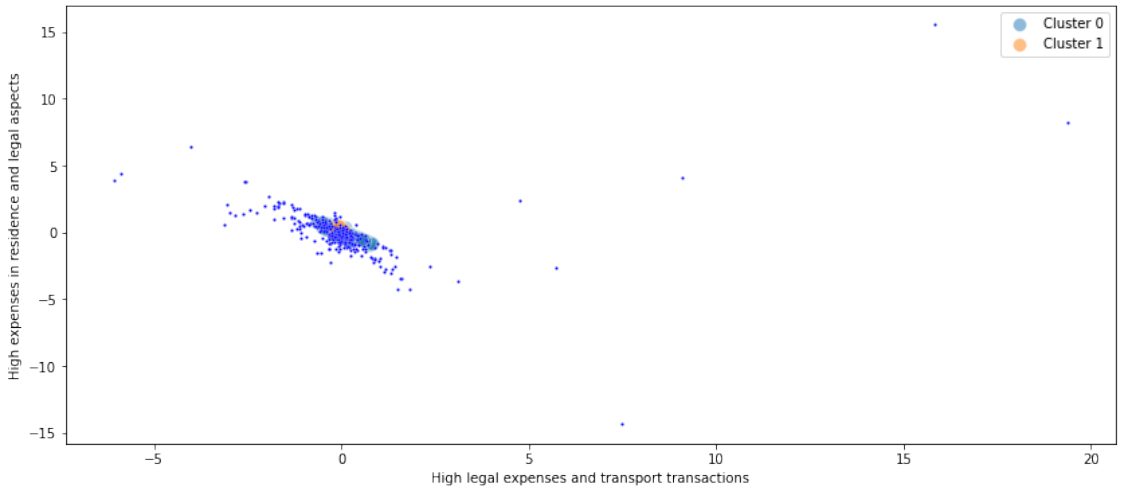
[146]:
```
fig = plt.figure(figsize=(14,6))

for cluster in range(0,num_clusters):
    sns.scatterplot(x="Vehicle and standard VAT card purchases", y="Non␣
 ↪expensive transactions not related to legal expenses nor residence", s=100,␣
 ↪alpha=0.5,
                    data=data2014_db.loc[data2014_db.Cluster==cluster,:])
sns.scatterplot(x="Vehicle and standard VAT card purchases", y="Non expensive␣
 ↪transactions not related to legal expenses nor residence", s=5, color = 'b',
                    data=data2014_db.loc[data2014_db.Cluster==-1,:])
plt.legend(['Cluster 0', 'Cluster 1'])
plt.show()
```



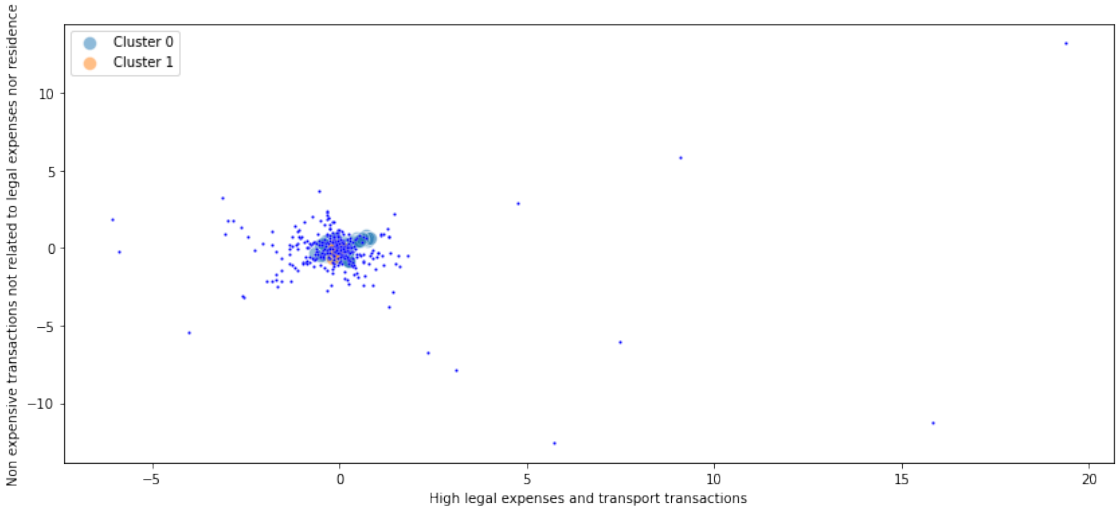
[147]:
```
fig = plt.figure(figsize=(14,6))

for cluster in range(0,num_clusters):
    sns.scatterplot(x="Vehicle and standard VAT card purchases",␣
 ↪y="Transactions related to services and residence", s=100, alpha=0.5,
                    data=data2014_db.loc[data2014_db.Cluster==cluster,:])
sns.scatterplot(x="Vehicle and standard VAT card purchases", y="Transactions␣
 ↪related to services and residence", s=5, color = 'b',
                    data=data2014_db.loc[data2014_db.Cluster==-1,:])
plt.legend(['Cluster 0', 'Cluster 1'])
plt.show()
```



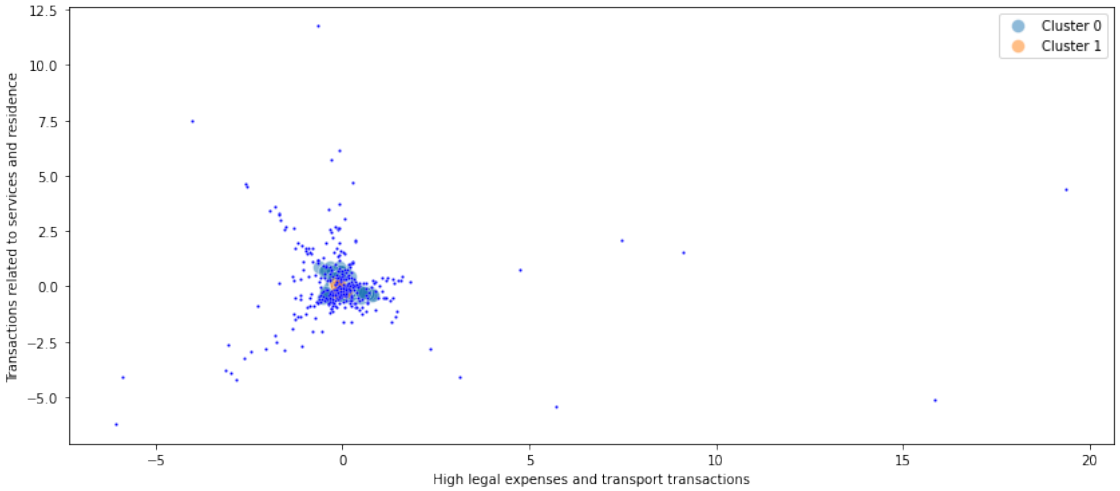
[148]:
```
fig = plt.figure(figsize=(14,6))

for cluster in range(0,num_clusters):
    sns.scatterplot(x="High legal expenses and transport transactions", y="High␣
 ↪expenses in residence and legal aspects", s=100, alpha=0.5,
                    data=data2014_db.loc[data2014_db.Cluster==cluster,:])
sns.scatterplot(x="High legal expenses and transport transactions", y="High␣
 ↪expenses in residence and legal aspects", s=5, color = 'b',
                    data=data2014_db.loc[data2014_db.Cluster==-1,:])
plt.legend(['Cluster 0', 'Cluster 1'])
plt.show()
```



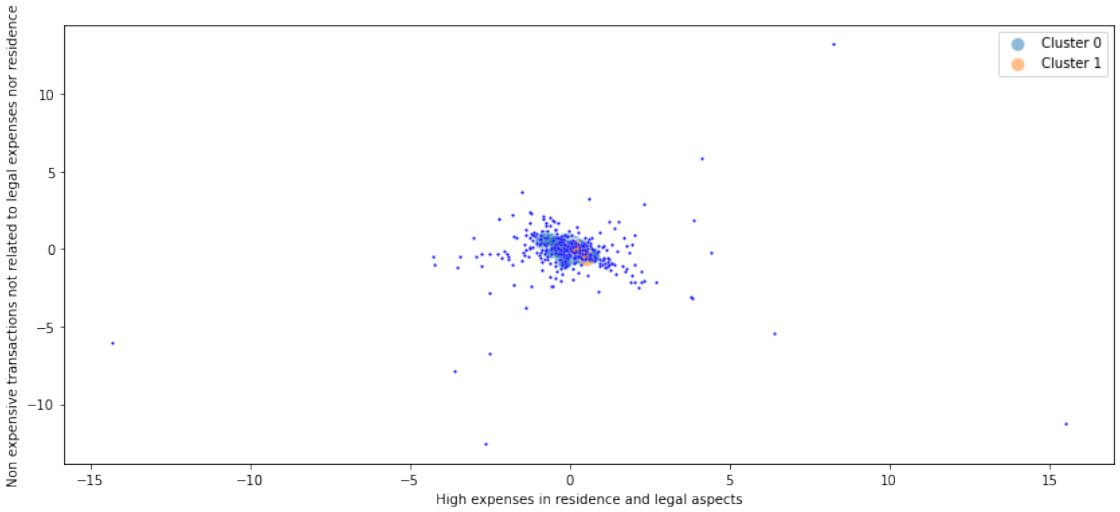
[149]:
```
fig = plt.figure(figsize=(14,6))

for cluster in range(0,num_clusters):
    sns.scatterplot(x="High legal expenses and transport transactions", y="Non␣
 ↪expensive transactions not related to legal expenses nor residence", s=100,␣
 ↪alpha=0.5,
                    data=data2014_db.loc[data2014_db.Cluster==cluster,:])
sns.scatterplot(x="High legal expenses and transport transactions", y="Non␣
 ↪expensive transactions not related to legal expenses nor residence", s=5,␣
 ↪color = 'b',
                    data=data2014_db.loc[data2014_db.Cluster==-1,:])
plt.legend(['Cluster 0', 'Cluster 1'])
plt.show()
```



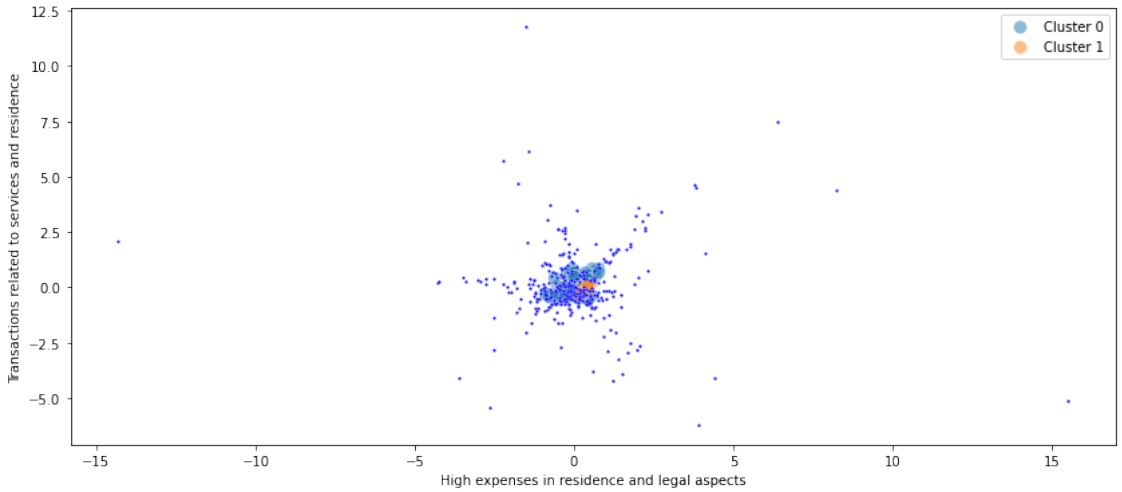
[150]:
```
fig = plt.figure(figsize=(14,6))

for cluster in range(0,num_clusters):
    sns.scatterplot(x="High legal expenses and transport transactions",␣
 ↪y="Transactions related to services and residence", s=100, alpha=0.5,
                    data=data2014_db.loc[data2014_db.Cluster==cluster,:])
sns.scatterplot(x="High legal expenses and transport transactions",␣
 ↪y="Transactions related to services and residence", s=5, color = 'b',
                    data=data2014_db.loc[data2014_db.Cluster==-1,:])
plt.legend(['Cluster 0', 'Cluster 1'])
plt.show()
```



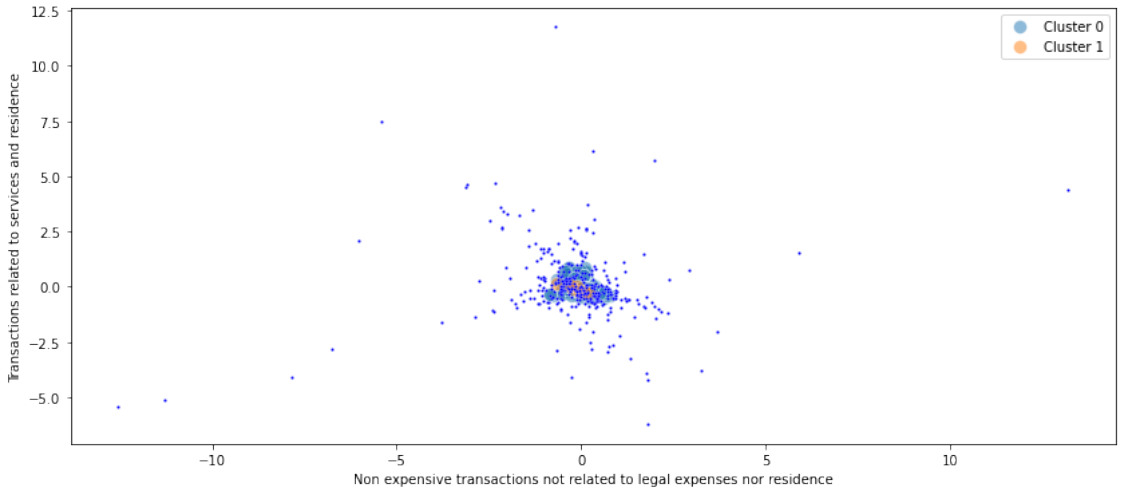
[151]:
```
fig = plt.figure(figsize=(14,6))

for cluster in range(0,num_clusters):
    sns.scatterplot(x="High expenses in residence and legal aspects", y="Non␣
 ↪expensive transactions not related to legal expenses nor residence", s=100,␣
 ↪alpha=0.5,
                    data=data2014_db.loc[data2014_db.Cluster==cluster,:])
sns.scatterplot(x="High expenses in residence and legal aspects", y="Non␣
 ↪expensive transactions not related to legal expenses nor residence", s=5,␣
 ↪color = 'b',
                    data=data2014_db.loc[data2014_db.Cluster==-1,:])
plt.legend(['Cluster 0', 'Cluster 1'])
plt.show()
```



[152]:
```
fig = plt.figure(figsize=(14,6))

for cluster in range(0,num_clusters):
    sns.scatterplot(x="High expenses in residence and legal aspects",␣
 ↪y="Transactions related to services and residence", s=100, alpha=0.5,
                    data=data2014_db.loc[data2014_db.Cluster==cluster,:])
sns.scatterplot(x="High expenses in residence and legal aspects",␣
 ↪y="Transactions related to services and residence", s=5, color = 'b',
                    data=data2014_db.loc[data2014_db.Cluster==-1,:])
plt.legend(['Cluster 0', 'Cluster 1'])
plt.show()
```



[153]:
```
fig = plt.figure(figsize=(14,6))

for cluster in range(0,num_clusters):
    sns.scatterplot(x="Non expensive transactions not related to legal expenses␣
 ↪nor residence", y="Transactions related to services and residence", s=100,␣
 ↪alpha=0.5,
                    data=data2014_db.loc[data2014_db.Cluster==cluster,:])
sns.scatterplot(x="Non expensive transactions not related to legal expenses nor␣
 ↪residence", y="Transactions related to services and residence", s=5, color =␣
 ↪'b',
                    data=data2014_db.loc[data2014_db.Cluster==-1,:])
plt.legend(['Cluster 0', 'Cluster 1'])
plt.show()
```



Profiling clients with DBSCAN with 2 clusters By observing the scatter plots we will try to profile the clients, although 2 clusters seem too little.

Cluster 0: more data points. Clients with the lowest High number of work/personal transactions (scattered area), average to high Vehicle and standard VAT card purchases, low High legal expenses and transport transactions, average in High expenses in residence and legal aspects, average in Non expensive transactions not related to legal expenses nor residence and less than average in Transactions related to services and residence.

Cluster 1: less data points. Clients with low High number of work/personal transactions, average Vehicle and standard VAT card purchases, low High legal expenses and transport transactions, average in High expenses in residence and legal aspects, average in Non expensive transactions not related to legal expenses nor residence and less than average in Transactions related to services and residence.

1.4.6 K-modes

We will try to use k-modes, an extension of k-means, which instead of using distances uses dissimilarities (that is, quantification of the total mismatches between two objects), so we can work with categorical values, most of the features available in this dataset.

We set the date of the transaction as the index because with this clustering technique we are going to try to answer the question which are the all around purchase card transactions' profiles for the year 2014?

[154]:
```
df2014_kmodes = df2014_pkl_loaded.copy()
df2014_kmodes.set_index('TRANS DATE',inplace=True)
```

[155]:
```
cost = []
K = range(1,9)
for num_clusters in list(K):
    kmode = KModes(n_clusters=num_clusters, init = "random", n_init = 5,␣
 ↪verbose=1)
    kmode.fit_predict(df2014_kmodes)
    cost.append(kmode.cost_)

plt.plot(K, cost, 'bx-')
plt.xlabel('No. of clusters')
plt.ylabel('Cost')
plt.title('Elbow Method For Optimal k')
plt.show()
```

```
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 0, cost: 255090.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
```

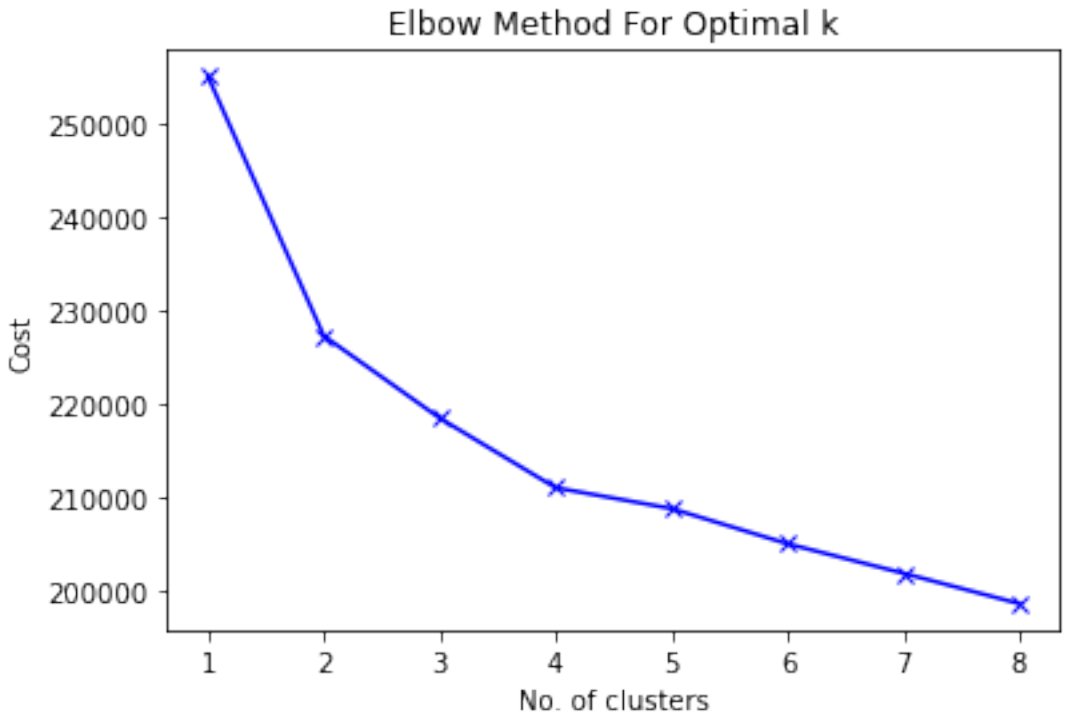
```
Run 2, iteration: 1/100, moves: 0, cost: 255090.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 3, iteration: 1/100, moves: 0, cost: 255090.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 4, iteration: 1/100, moves: 0, cost: 255090.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 5, iteration: 1/100, moves: 0, cost: 255090.0
Best run was number 1
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 8026, cost: 227306.0
Run 1, iteration: 2/100, moves: 173, cost: 227306.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 2, iteration: 1/100, moves: 4329, cost: 232472.0
Run 2, iteration: 2/100, moves: 1446, cost: 232459.0
Run 2, iteration: 3/100, moves: 5, cost: 232459.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 3, iteration: 1/100, moves: 4627, cost: 237217.0
Run 3, iteration: 2/100, moves: 1571, cost: 237023.0
Run 3, iteration: 3/100, moves: 16, cost: 237023.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 4, iteration: 1/100, moves: 1175, cost: 242676.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 5, iteration: 1/100, moves: 8064, cost: 227329.0
Run 5, iteration: 2/100, moves: 43, cost: 227329.0
Best run was number 1
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 9060, cost: 219106.0
Run 1, iteration: 2/100, moves: 2174, cost: 218502.0
Run 1, iteration: 3/100, moves: 111, cost: 218502.0
Init: initializing centroids
```

```
Init: initializing clusters
Starting iterations…
Run 2, iteration: 1/100, moves: 4629, cost: 219813.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 3, iteration: 1/100, moves: 1870, cost: 226656.0
Run 3, iteration: 2/100, moves: 22, cost: 226656.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 4, iteration: 1/100, moves: 8325, cost: 220565.0
Run 4, iteration: 2/100, moves: 4116, cost: 220561.0
Run 4, iteration: 3/100, moves: 82, cost: 220561.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 5, iteration: 1/100, moves: 9531, cost: 219846.0
Run 5, iteration: 2/100, moves: 525, cost: 219769.0
Run 5, iteration: 3/100, moves: 12, cost: 219769.0
Best run was number 1
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 12161, cost: 215461.0
Run 1, iteration: 2/100, moves: 131, cost: 215461.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 2, iteration: 1/100, moves: 6289, cost: 218110.0
Run 2, iteration: 2/100, moves: 1885, cost: 218110.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 3, iteration: 1/100, moves: 4338, cost: 211050.0
Run 3, iteration: 2/100, moves: 6, cost: 211050.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 4, iteration: 1/100, moves: 10544, cost: 215321.0
Run 4, iteration: 2/100, moves: 34, cost: 215321.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 5, iteration: 1/100, moves: 5047, cost: 216653.0
Run 5, iteration: 2/100, moves: 576, cost: 216099.0
Run 5, iteration: 3/100, moves: 38, cost: 216099.0
Best run was number 3
```

```
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 4884, cost: 211585.0
Run 1, iteration: 2/100, moves: 68, cost: 211585.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 2, iteration: 1/100, moves: 3730, cost: 216432.0
Run 2, iteration: 2/100, moves: 56, cost: 216432.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 3, iteration: 1/100, moves: 3980, cost: 217047.0
Run 3, iteration: 2/100, moves: 206, cost: 217047.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 4, iteration: 1/100, moves: 7415, cost: 215600.0
Run 4, iteration: 2/100, moves: 338, cost: 215600.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 5, iteration: 1/100, moves: 3611, cost: 214671.0
Run 5, iteration: 2/100, moves: 8466, cost: 208823.0
Run 5, iteration: 3/100, moves: 1860, cost: 208823.0
Best run was number 5
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 9774, cost: 209990.0
Run 1, iteration: 2/100, moves: 234, cost: 209990.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 2, iteration: 1/100, moves: 9058, cost: 208964.0
Run 2, iteration: 2/100, moves: 719, cost: 208964.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 3, iteration: 1/100, moves: 3989, cost: 207690.0
Run 3, iteration: 2/100, moves: 228, cost: 207683.0
Run 3, iteration: 3/100, moves: 0, cost: 207683.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 4, iteration: 1/100, moves: 7880, cost: 205082.0
Run 4, iteration: 2/100, moves: 2762, cost: 205055.0
```

```
Run 4, iteration: 3/100, moves: 14, cost: 205055.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 5, iteration: 1/100, moves: 1180, cost: 206741.0
Run 5, iteration: 2/100, moves: 15, cost: 206741.0
Best run was number 4
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 3403, cost: 205732.0
Run 1, iteration: 2/100, moves: 534, cost: 205732.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 2, iteration: 1/100, moves: 7311, cost: 205349.0
Run 2, iteration: 2/100, moves: 123, cost: 205349.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 3, iteration: 1/100, moves: 5266, cost: 206312.0
Run 3, iteration: 2/100, moves: 88, cost: 206312.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 4, iteration: 1/100, moves: 5646, cost: 206035.0
Run 4, iteration: 2/100, moves: 136, cost: 206035.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 5, iteration: 1/100, moves: 4454, cost: 201901.0
Run 5, iteration: 2/100, moves: 8, cost: 201901.0
Best run was number 5
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 7964, cost: 198676.0
Run 1, iteration: 2/100, moves: 993, cost: 198649.0
Run 1, iteration: 3/100, moves: 5, cost: 198649.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 2, iteration: 1/100, moves: 8008, cost: 203423.0
Run 2, iteration: 2/100, moves: 399, cost: 203284.0
Run 2, iteration: 3/100, moves: 6, cost: 203284.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
```

```
Run 3, iteration: 1/100, moves: 5761, cost: 202339.0
Run 3, iteration: 2/100, moves: 120, cost: 202339.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 4, iteration: 1/100, moves: 7517, cost: 200576.0
Run 4, iteration: 2/100, moves: 63, cost: 200576.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 5, iteration: 1/100, moves: 5516, cost: 204918.0
Run 5, iteration: 2/100, moves: 177, cost: 204780.0
Run 5, iteration: 3/100, moves: 56, cost: 204780.0
Best run was number 1
```



Using the elbow method we found out that the amount of clusters may be 2 or 8. We will analyze both of them.

[156]:
```
df2014_kmodes2 = df2014_kmodes.copy()
kmodes = KModes(n_clusters=2)
df2014_kmodes2['kmodes_cluster'] = kmodes.fit_predict(df2014_kmodes2)

df2014_kmodes8 = df2014_kmodes.copy()
kmodes = KModes(n_clusters=8)
```

```
df2014_kmodes8['kmodes_cluster'] = kmodes.fit_predict(df2014_kmodes8)
```

We found out the names of the features in the dataframe

[157]:
```
df2014_kmodes.columns
```

[157]:
```
Index(['TRANS VAT DESC', 'ORIGINAL GROSS AMT', 'MERCHANT NAME', 'CARD NUMBER',
       'BILLING CUR CODE', 'TRANS CAC CODE 1', 'TRANS CAC DESC 1',
       'TRANS CAC CODE 2', 'TRANS CAC DESC 2', 'TRANS CAC CODE 3',
       'Directorate'],
      dtype='object')
```

We define a function to make a count plot for each of the categories of the features for each cluster. That in order to profile the cluster of transactions.

[158]:
```
def analyze_cluster(features,data,cluster_number):
    for feature in features:
        plt.figure(figsize=(15,15))
        sns.countplot(x=feature,␣
↪data=data[data['kmodes_cluster']==cluster_number])
        plt.show()
```

[159]:
```
type(df2014_kmodes2)
```

[159]:
```
pandas.core.frame.DataFrame
```

We reduce the categories in 'TRANS CAC DESC 1' by grouping them in similar groups.

[160]:
```
feature = 'TRANS CAC DESC 1'
old_cats = ['Vehicle Fuel','Vehicle Excise Lics','Vehicle R&M','Vehicle␣
↪OthrunCosts','Car Allowances etc','Car Parking','Vehicle Hire␣
↪Charge','Vehicle Tyres']
new_cat = 'Vehicles'
for c in old_cats:
    df2014_kmodes2.loc[df2014_kmodes2[feature] == c, feature] = new_cat
    df2014_kmodes8.loc[df2014_kmodes8[feature] == c, feature] = new_cat
```

[161]:
```
feature = 'TRANS CAC DESC 1'
old_cats = ['Books','Purchases Food','Hospitality','Postage','Personal␣
↪Needs','Clothing&Uniforms',
            'Subscriptions','Laundry','Entertainers/Artists','Mobiles/Radios/
↪Pagrs','Floral Decorations',
           'Aftercare Assistance',"Mat'l Raw/Drct",'In Year Credits','CSDP Act␣
↪Telephones']
new_cat = 'Personal needs'
for c in old_cats:
    df2014_kmodes2.loc[df2014_kmodes2[feature] == c, feature] = new_cat
    df2014_kmodes8.loc[df2014_kmodes8[feature] == c, feature] = new_cat
```

[162]:
```
feature = 'TRANS CAC DESC 1'
old_cats = ['Photocopying','Gas','Electricity',"N'Papers&Periodicals",'Other␣
↪Services','Disinfestation',
          'Consultancy Fees','Security Contracts','Phon NonCentrx Lines','Bank␣
↪& Goro ChgsS','Refuse Collection']
new_cat = 'Services'
for c in old_cats:
    df2014_kmodes2.loc[df2014_kmodes2[feature] == c, feature] = new_cat
    df2014_kmodes8.loc[df2014_kmodes8[feature] == c, feature] = new_cat
```

[163]:
```
feature = 'TRANS CAC DESC 1'
old_cats = ['Travel Bus/Rail',"Ttavel Other (UK)",'Travel Taxis','Travel␣
↪Foreign','Transport Misc',
           'Transport Insurance']
new_cat = 'Transport'
for c in old_cats:
    df2014_kmodes2.loc[df2014_kmodes2[feature] == c, feature] = new_cat
    df2014_kmodes8.loc[df2014_kmodes8[feature] == c, feature] = new_cat
```

[164]:
```
feature = 'TRANS CAC DESC 1'
old_cats = ['Other Fix&Fittings','Bldg RM Fair Fund NS','Bldg RM Routine␣
↪UDD','Bldg RM Emergy UDD',
          'Bldg RM Fair Fund S','Rents incl Svce Chgs','Bldg RM Departmental']
new_cat = 'Residence'
for c in old_cats:
    df2014_kmodes2.loc[df2014_kmodes2[feature] == c, feature] = new_cat
    df2014_kmodes8.loc[df2014_kmodes8[feature] == c, feature] = new_cat
```

[165]:
```
feature = 'TRANS CAC DESC 1'
old_cats = ['Training Other','Computing Other','Equip Other','Staff Advert␣
↪Exp','Equip Operational',
           'Supplies & Sev Mic','Conference Fees Subs Foreign','Conference␣
↪Fees Subs UK','Equip Office',
           'Equip Maintenance','Licences & Permits','Stock Misc','Cleaning␣
↪Materials','Grounds Maintenance',
           "Signs & N'boards",'Premises Provisions','Oth Indirect␣
↪EmpExps','Advertising NonStaff',
           'GoodsPurchforResale','Promotions/Marketing','Other␣
↪Agencies','Accomodation Hire',
           "Purchases Othermat'l","Training EquipMat'ls",'Catering␣
↪Disposables',"Fire/Sec'yAlarm/Eq't",
           "Equip Hire/Op Lease",'Contract␣
↪Meals','NonEmpAllow-General','Insurance NonPremise',
           'Other Third Parties','Training Travel&Subs','IT Leasing␣
↪Charges','NonEmpAllow-Training',
```

```
            'Visits Expenditure','Premises Misc',"Not'l fin Cha␣
 ↪IT",'Stationery','HR&M Door Entry',
            'Recordings (S&V)']
new_cat = 'Work related'
for c in old_cats:
    df2014_kmodes2.loc[df2014_kmodes2[feature] == c, feature] = new_cat
    df2014_kmodes8.loc[df2014_kmodes8[feature] == c, feature] = new_cat
```

[166]:
```
feature = 'TRANS CAC DESC 1'
old_cats = ['Prof Fees other','Family Support S17','Other Grants','Training␣
 ↪Tutor Fees','SchGovBrds Clerks',
            'Sec. 24 CH Act 1989']
new_cat = "Family&School"
for c in old_cats:
    df2014_kmodes2.loc[df2014_kmodes2[feature] == c, feature] = new_cat
    df2014_kmodes8.loc[df2014_kmodes8[feature] == c, feature] = new_cat
```

[167]:
```
feature = 'TRANS CAC DESC 1'
old_cats = ['Legal Fee Other','Witness Expenses']
new_cat = "Legal"
for c in old_cats:
    df2014_kmodes2.loc[df2014_kmodes2[feature] == c, feature] = new_cat
    df2014_kmodes8.loc[df2014_kmodes8[feature] == c, feature] = new_cat
```

1.4.7 We profile the clusters that came out from k-modes where k=2.

[168]:
```
df2014_kmodes2.columns
```

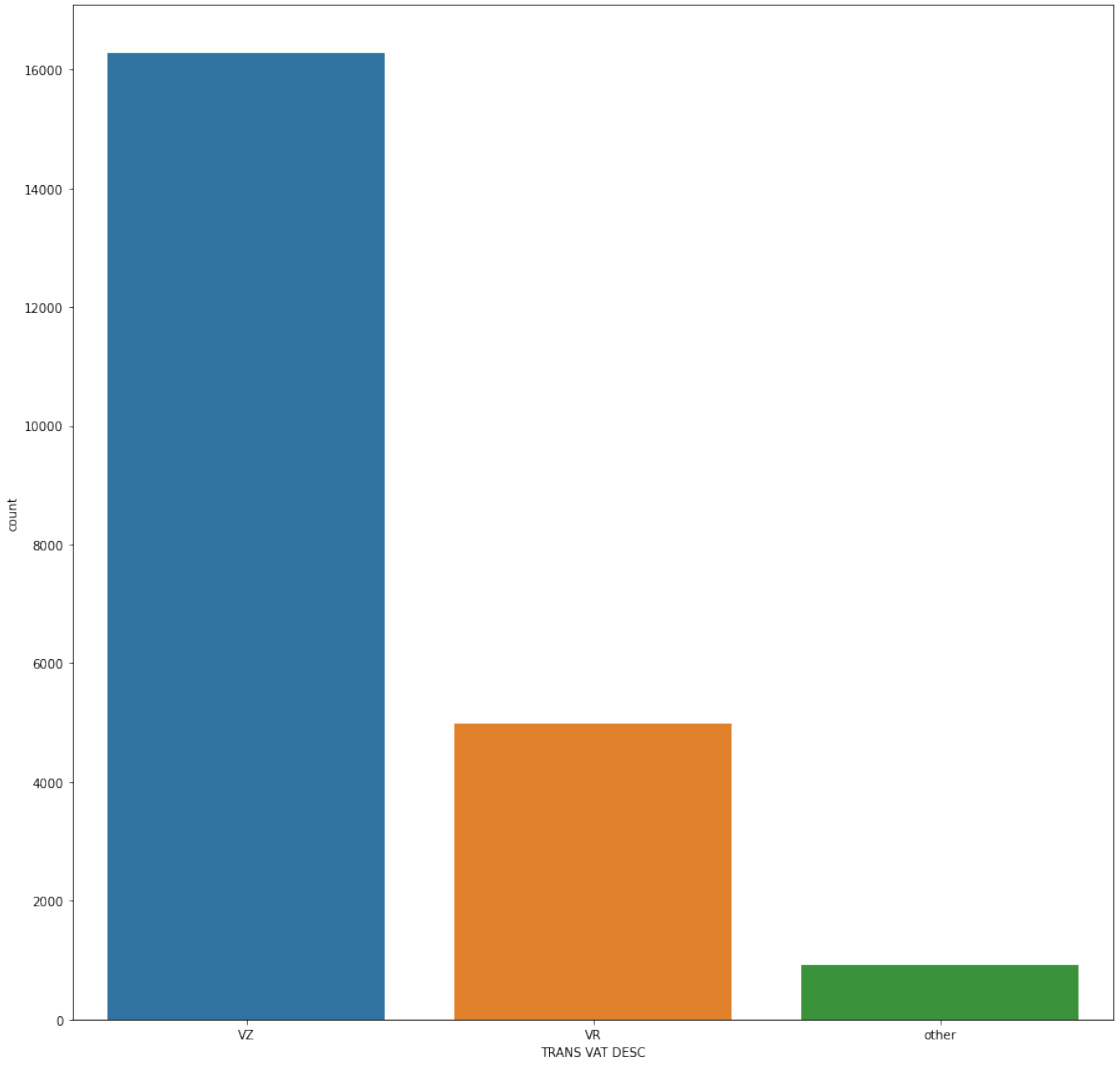
[168]:
```
Index(['TRANS VAT DESC', 'ORIGINAL GROSS AMT', 'MERCHANT NAME', 'CARD NUMBER',
       'BILLING CUR CODE', 'TRANS CAC CODE 1', 'TRANS CAC DESC 1',
       'TRANS CAC CODE 2', 'TRANS CAC DESC 2', 'TRANS CAC CODE 3',
       'Directorate', 'kmodes_cluster'],
      dtype='object')
```

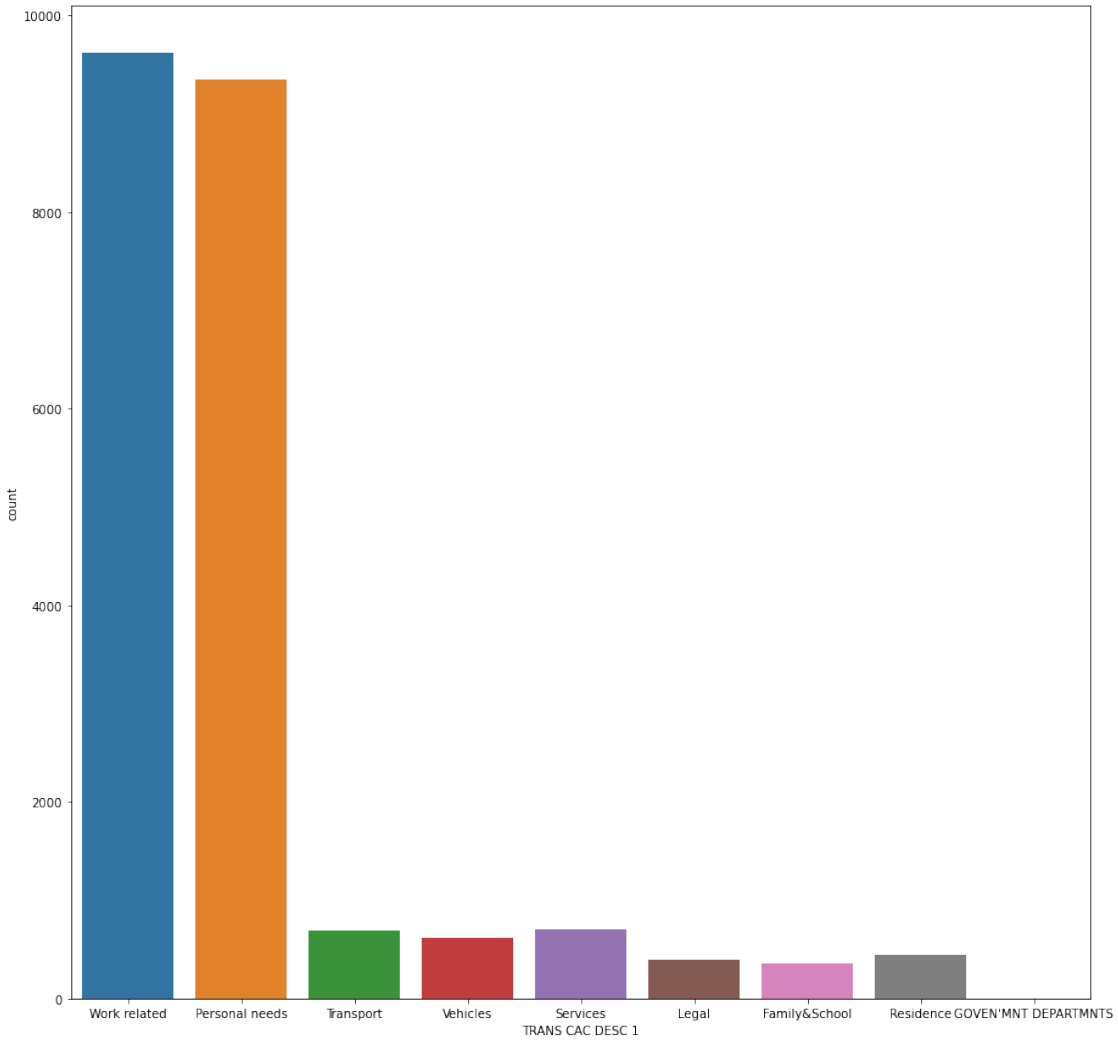
We define the features we will need to analyze each cluster.

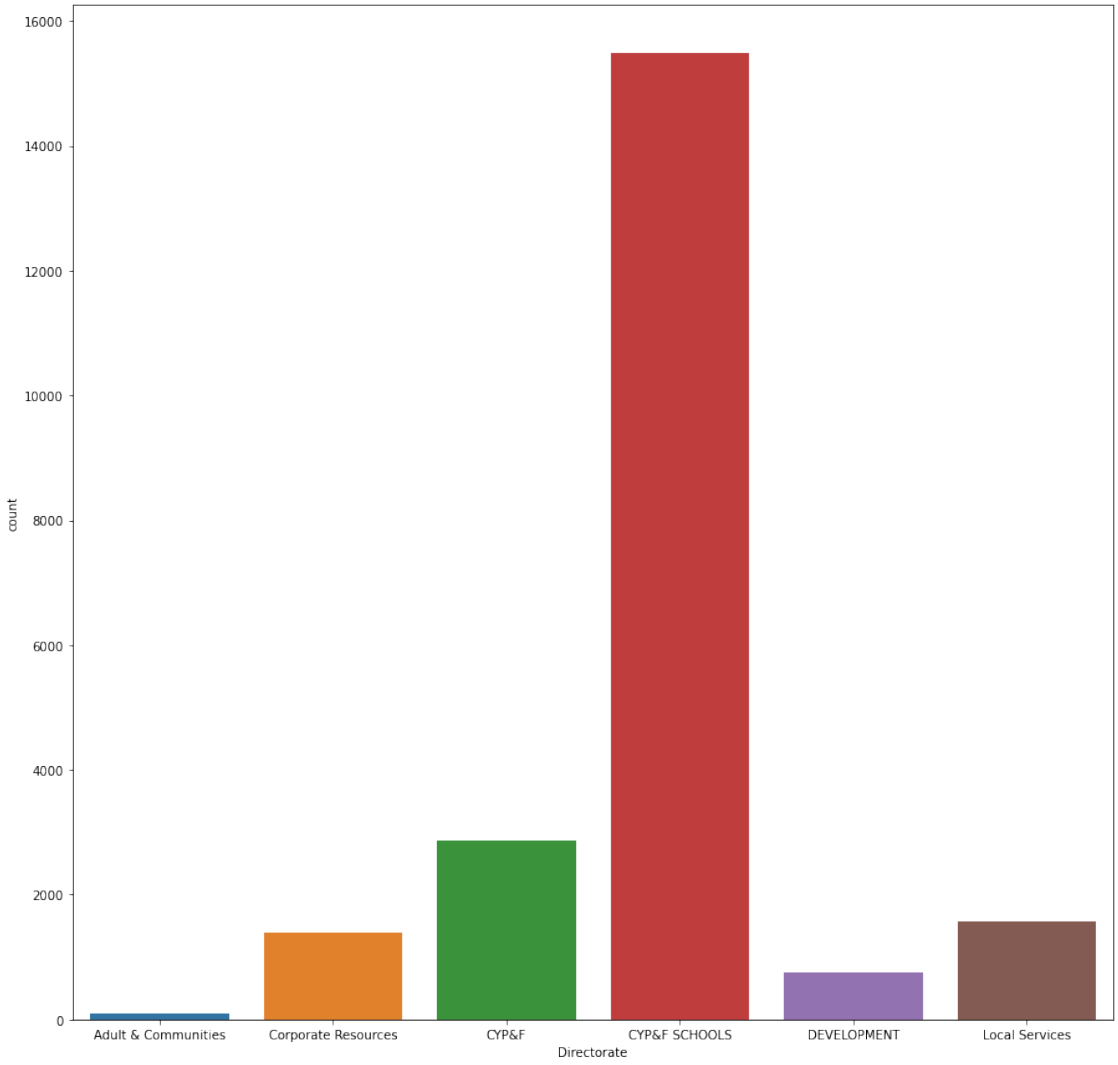
[169]:
```
features = df2014_kmodes.columns[[0,6,10]]
```

We analyze cluster 0:

[170]:
```
analyze_cluster(features,df2014_kmodes2,0)
```


16000
14000
12000
10000
8000
6000
4000
2000
0
count
VZ
VR
other
TRANS VAT DESC


10000
8000
6000
4000
2000
0
count
Work related
Personal needs
Transport
Vehicles
Services
Legal
Family&School
Residence
GOVEN'MNT DEPARTMNTS
TRANS CAC DESC 1

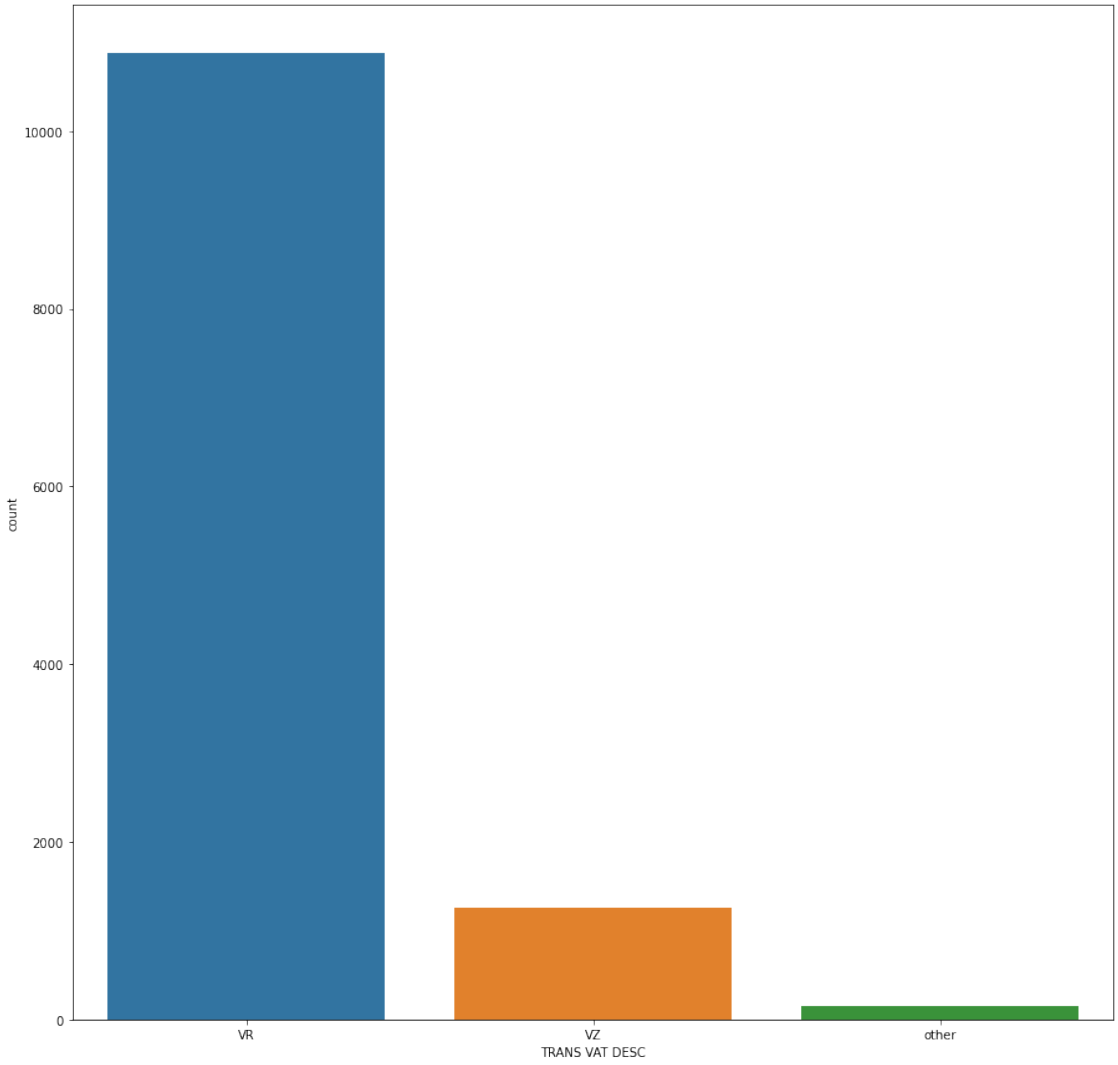


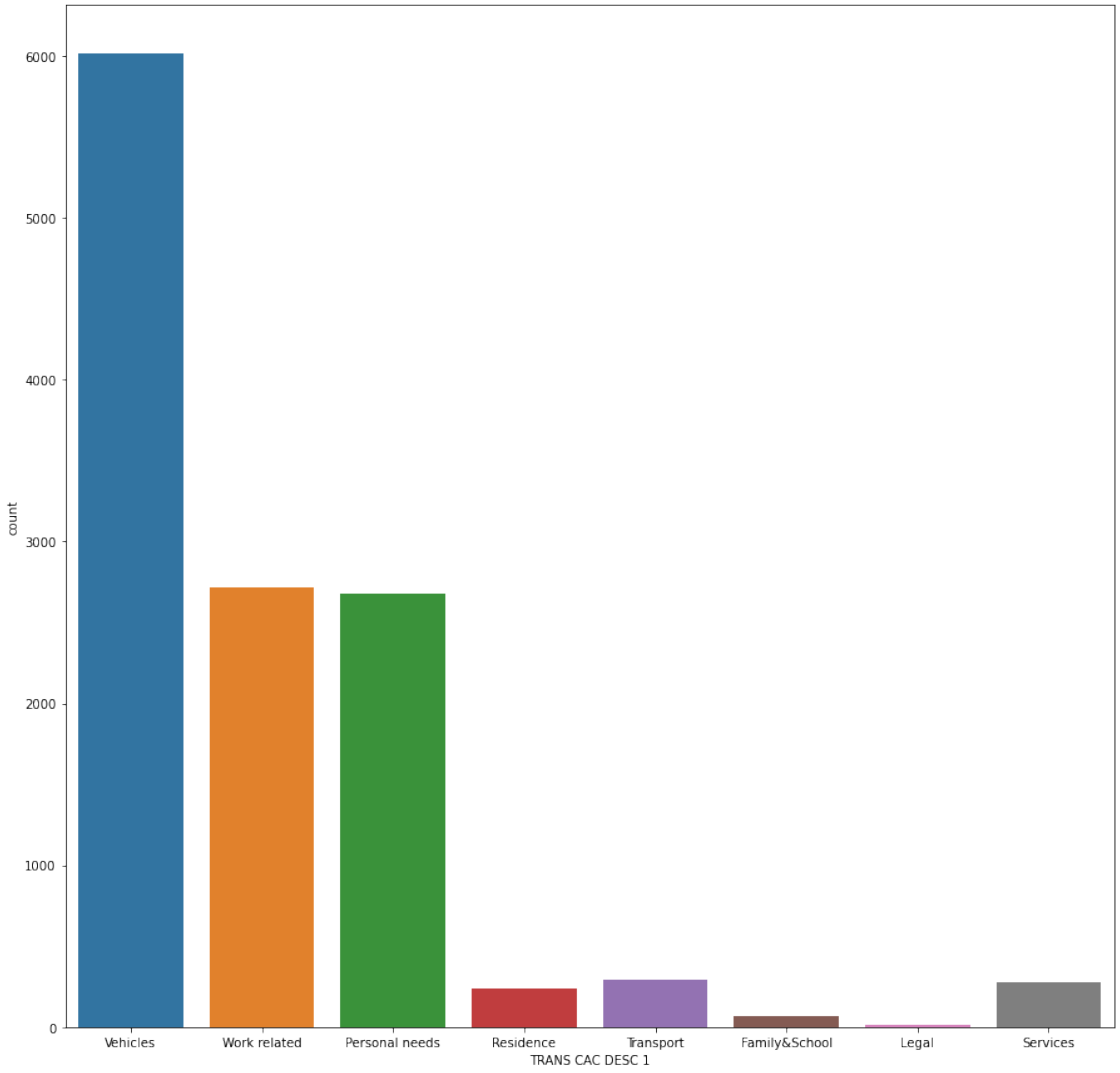
Profiling transactions

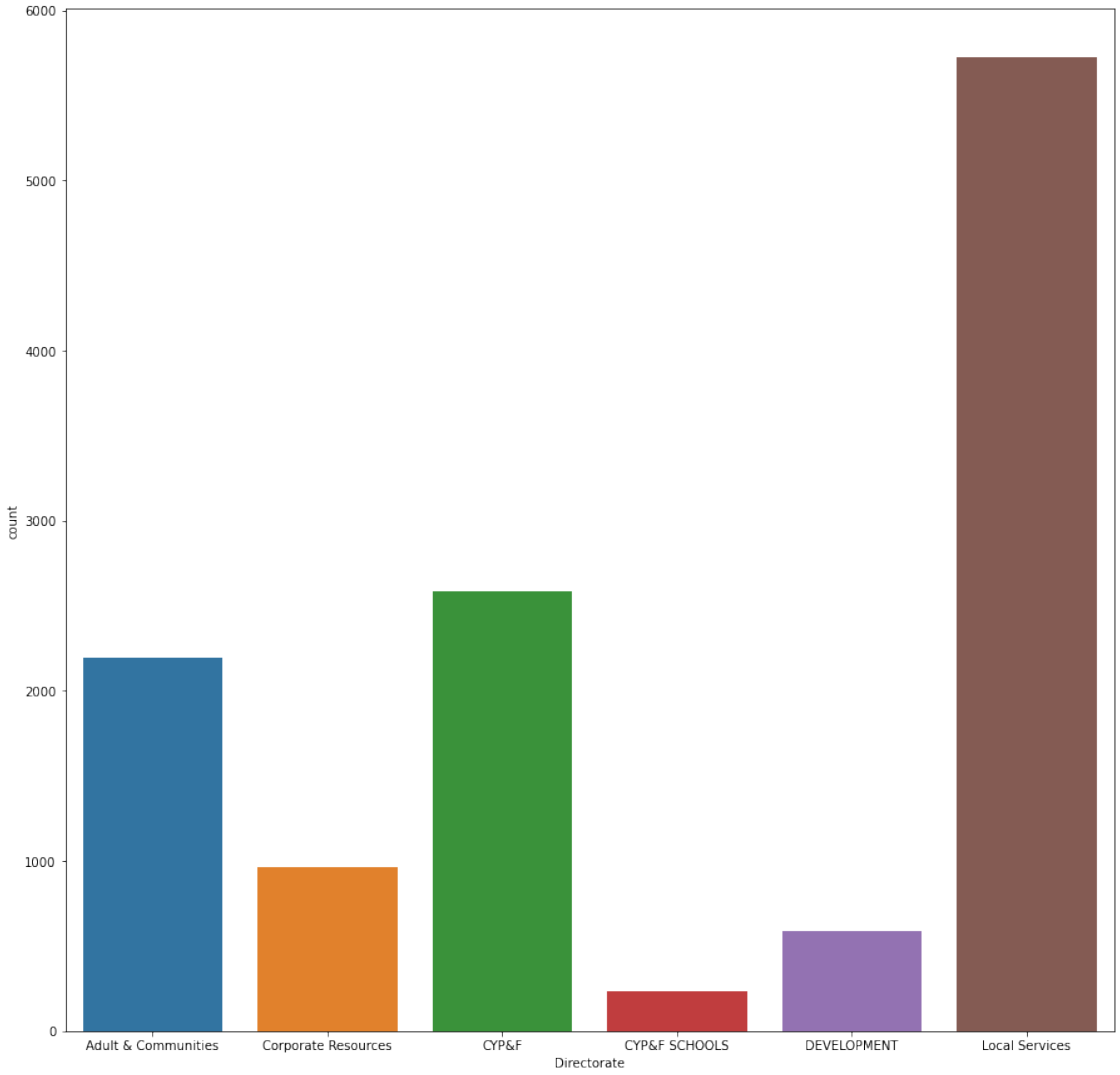
Cluster 0: these are transactions mainly with zero Value added tax, related to work and personal needs. Are involved with the CY&PF directorate.

We analyze cluster 1:

[171]:
```
analyze_cluster(features,df2014_kmodes2,1)
```


10000
8000
6000
4000
2000
0
count
VR
VZ
other
TRANS VAT DESC


6000
5000
4000
3000
2000
1000
0
count
Vehicles
Work related
Personal needs
Residence
Transport
Family&School
Legal
Services
TRANS CAC DESC 1



Profiling transactions

Cluster 1: these are transactions mainly with standard Value added tax, related in a high percentage to vehicles. Much less about work and personal needs. Are involved mainly with local services.

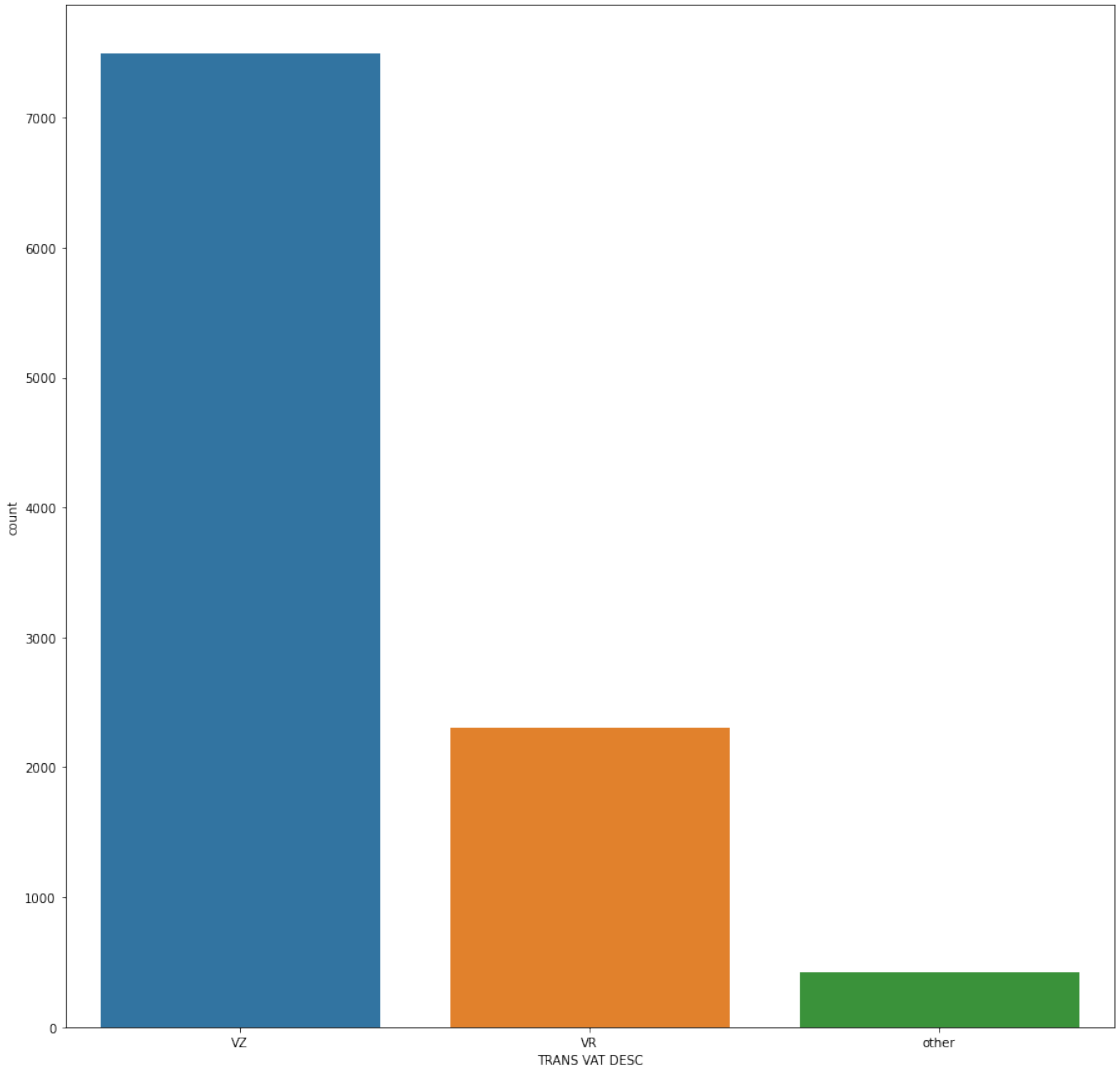
1.4.8 Now we profile the clusters that came out from k-modes where k=8.

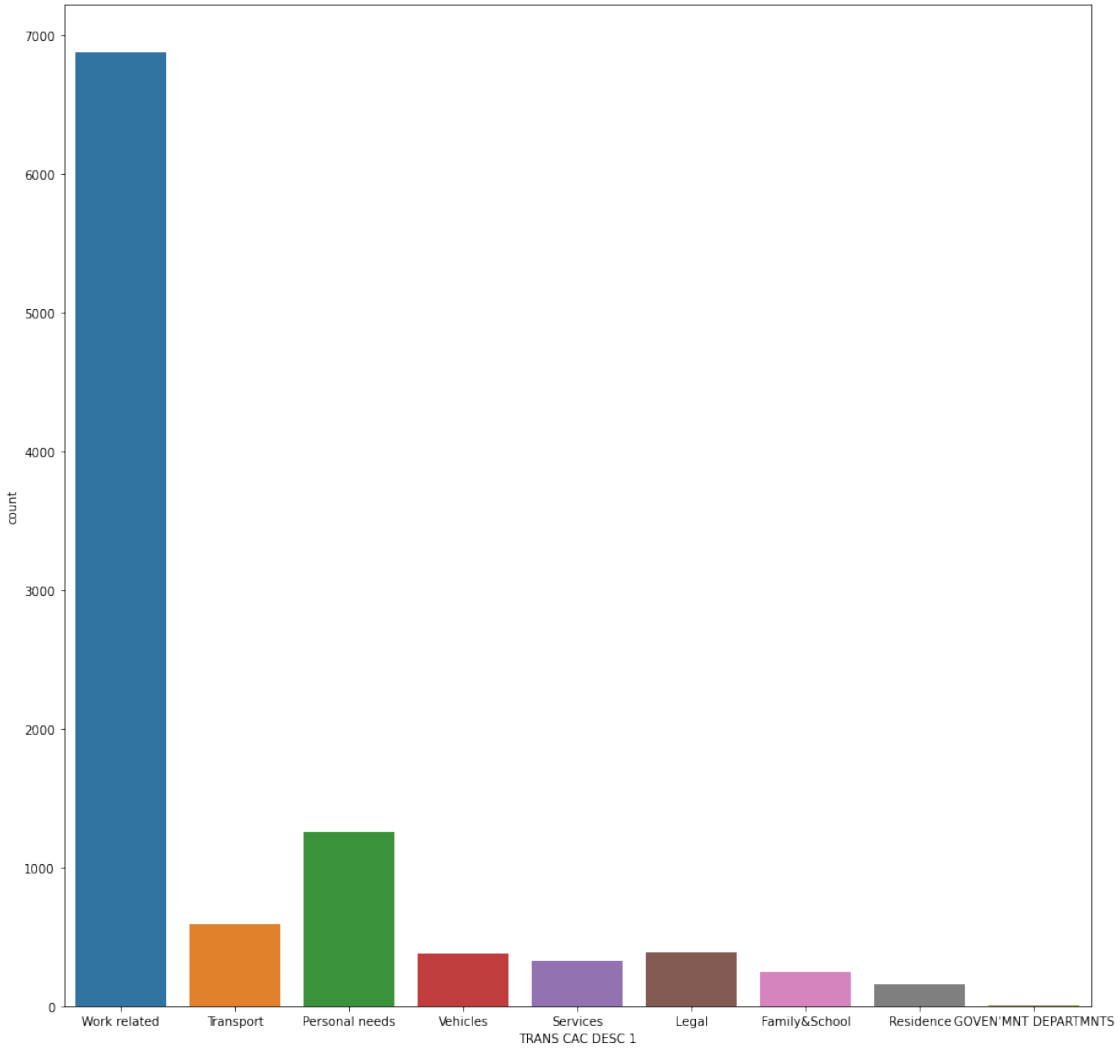
[172]:
```
df2014_kmodes8.columns
```

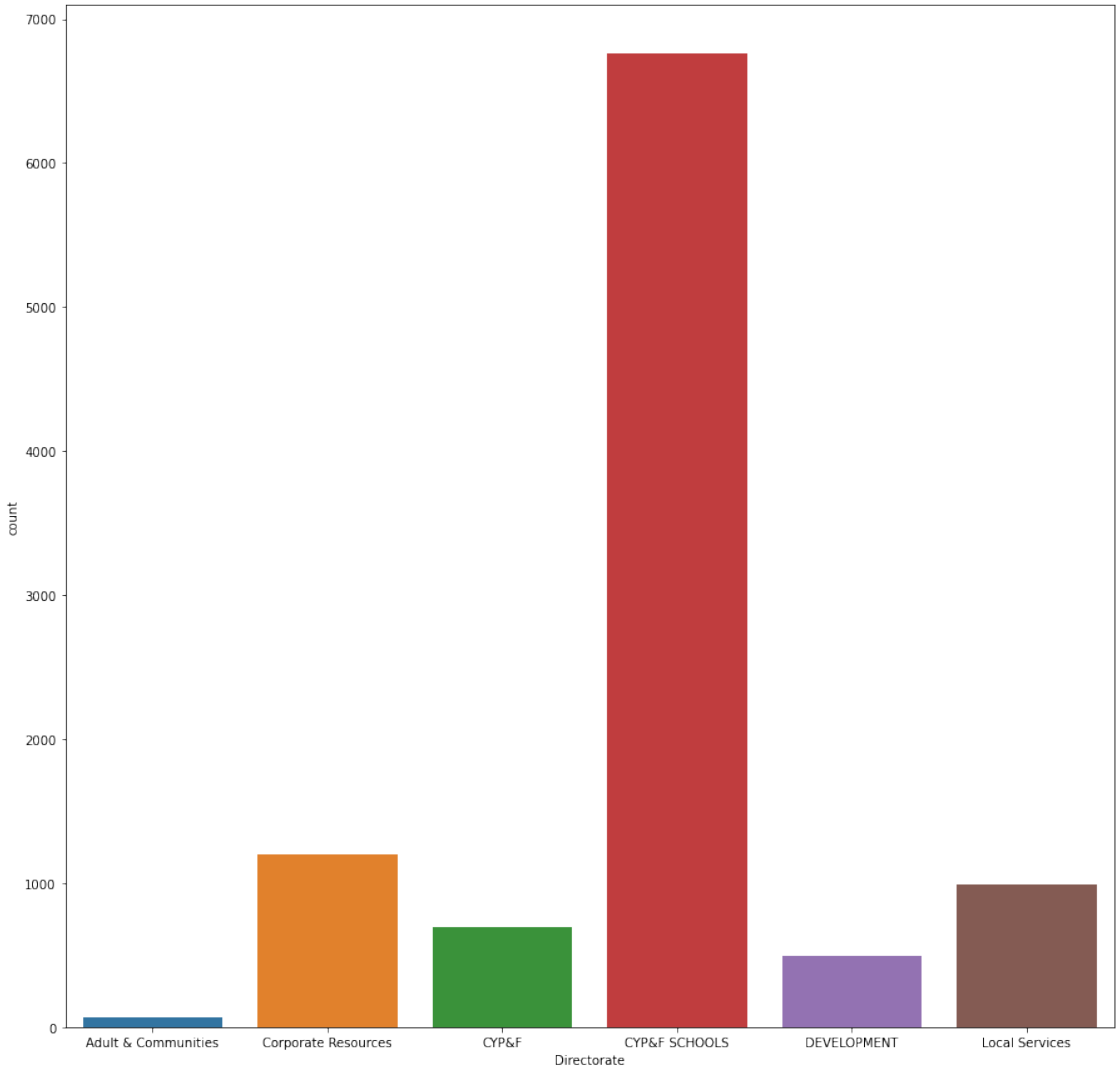
[172]:
```
Index(['TRANS VAT DESC', 'ORIGINAL GROSS AMT', 'MERCHANT NAME', 'CARD NUMBER',
       'BILLING CUR CODE', 'TRANS CAC CODE 1', 'TRANS CAC DESC 1',
       'TRANS CAC CODE 2', 'TRANS CAC DESC 2', 'TRANS CAC CODE 3',
       'Directorate', 'kmodes_cluster'],
      dtype='object')
```

We analyze cluster 0:

[173]:
```
analyze_cluster(features,df2014_kmodes8,0)
```




7000
6000
5000
4000
3000
2000
1000
0
count
Work related
Transport
Personal needs
Vehicles
Services
Legal
Family&School
Residence
GOVEN'MNT DEPARTMNTS
TRANS CAC DESC 1

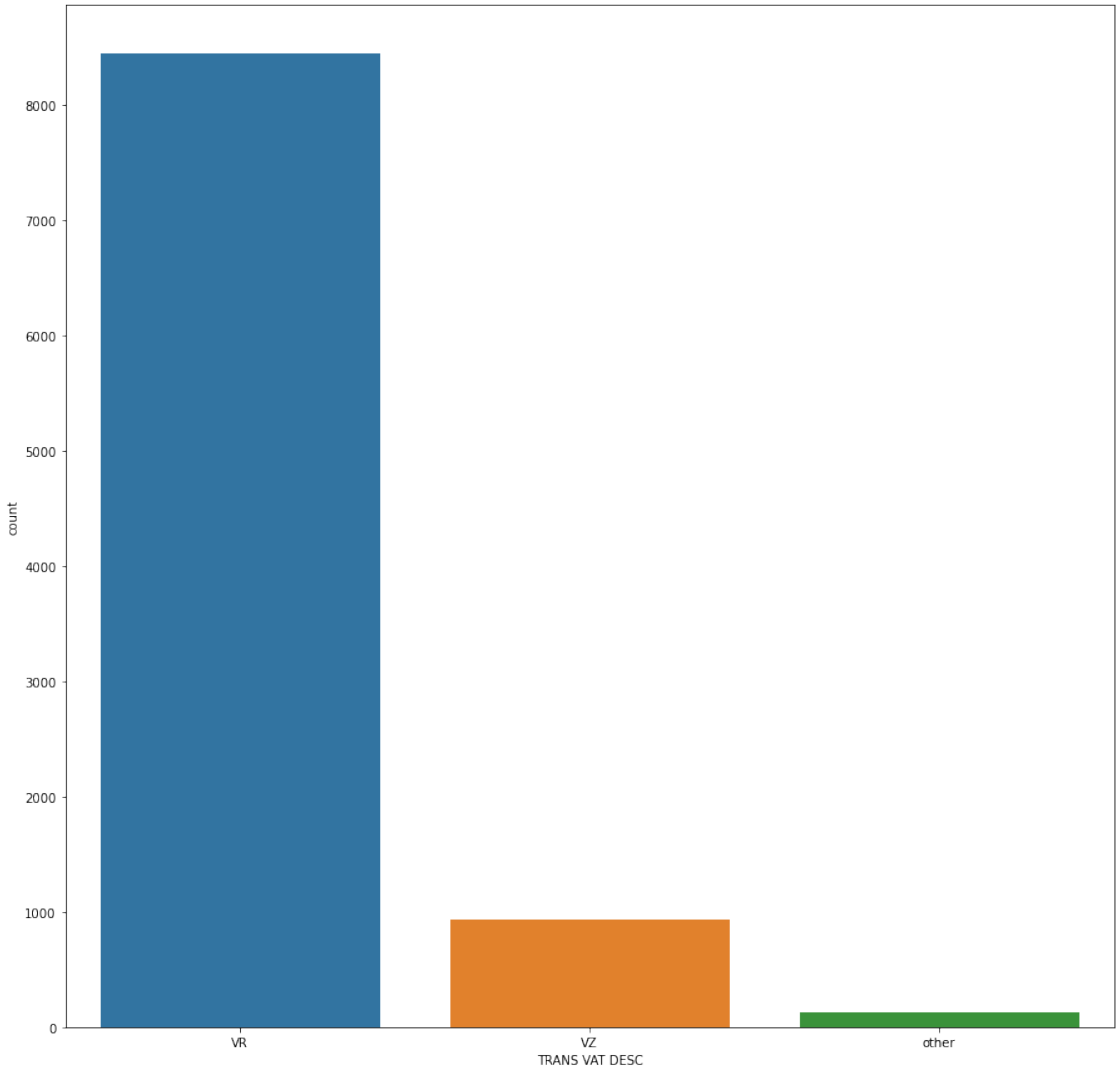


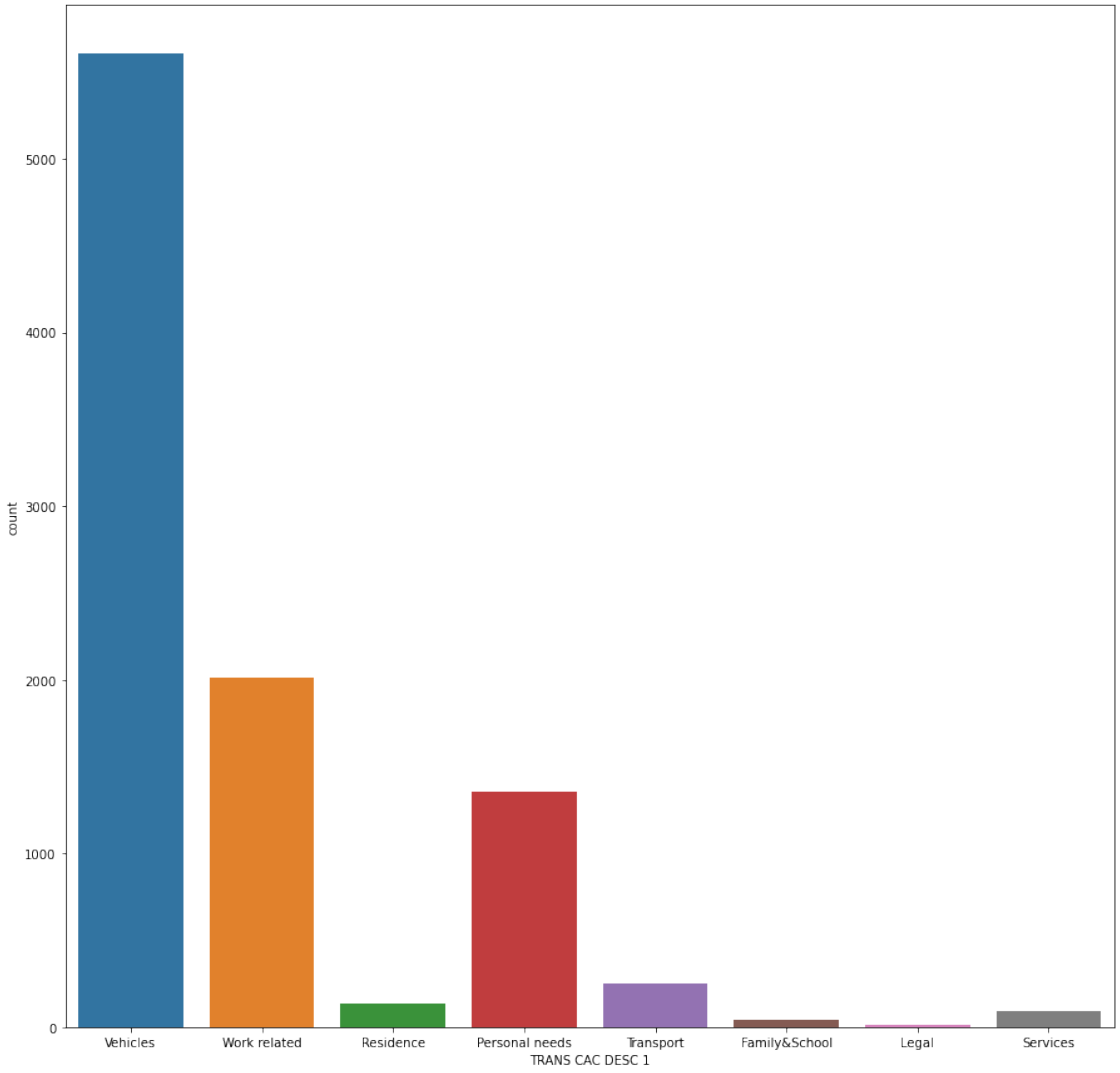
Profiling transactions

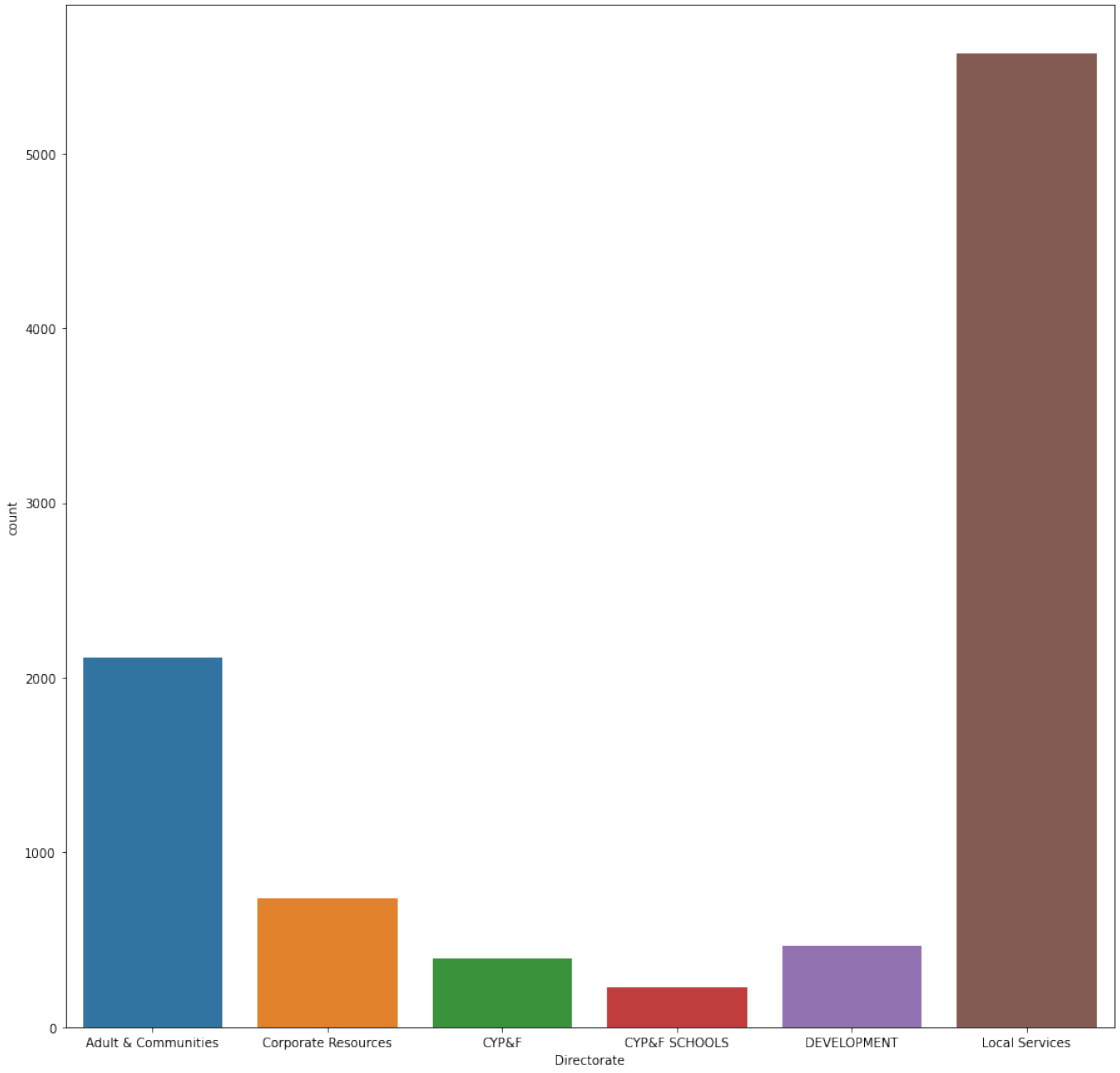
Cluster 0: these are transactions mainly with standard Value added tax, related to work. Are involved with the CY&PF Schools directorate.

We analyze cluster 1:

[174]:
```
analyze_cluster(features,df2014_kmodes8,1)
```


8000
7000
6000
5000
4000
3000
2000
1000
0
count
VR
VZ
other
TRANS VAT DESC


5000
4000
3000
2000
1000
0
count
Vehicles
Work related
Residence
Personal needs
Transport
Family&School
Legal
Services
TRANS CAC DESC 1

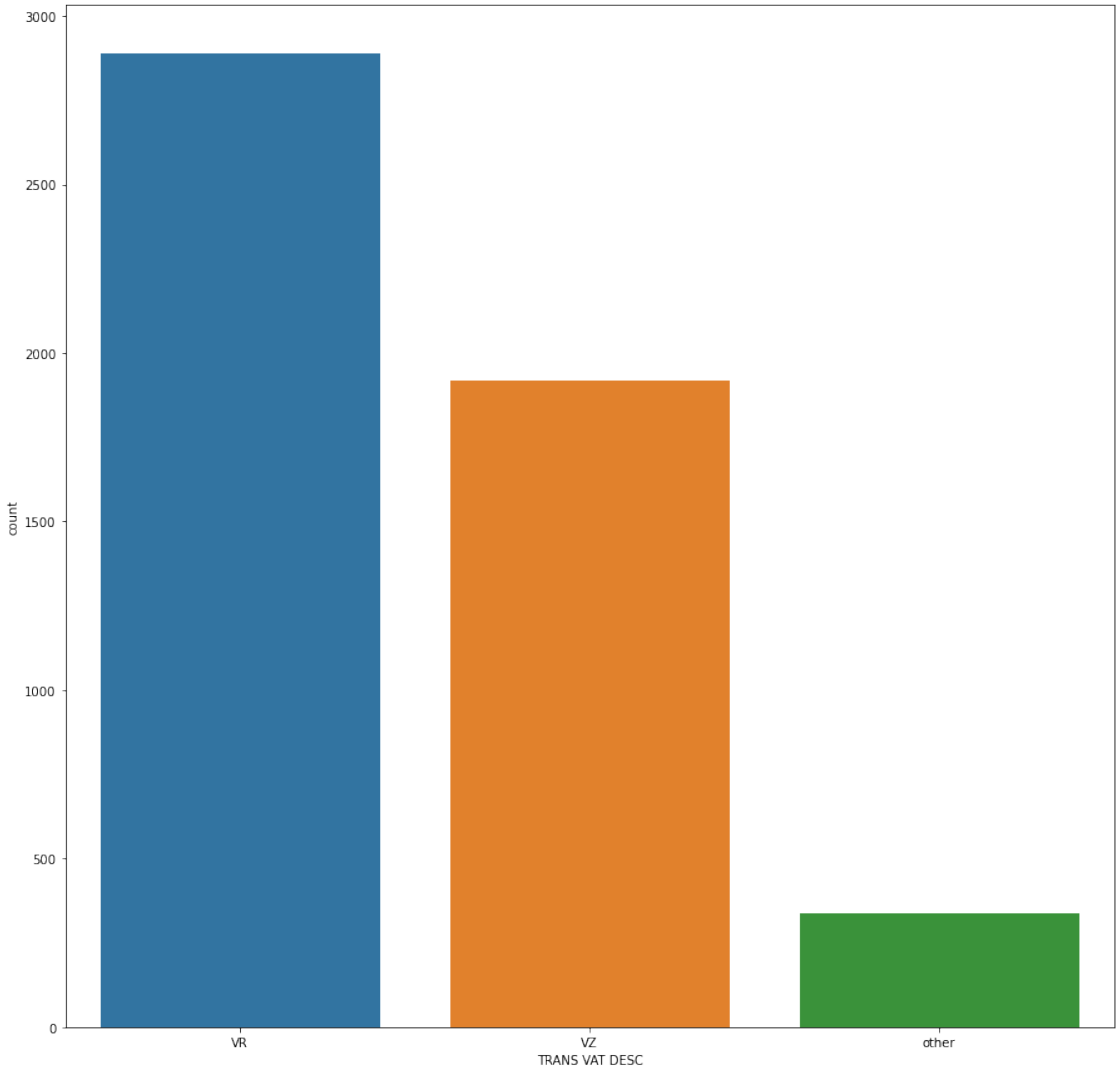


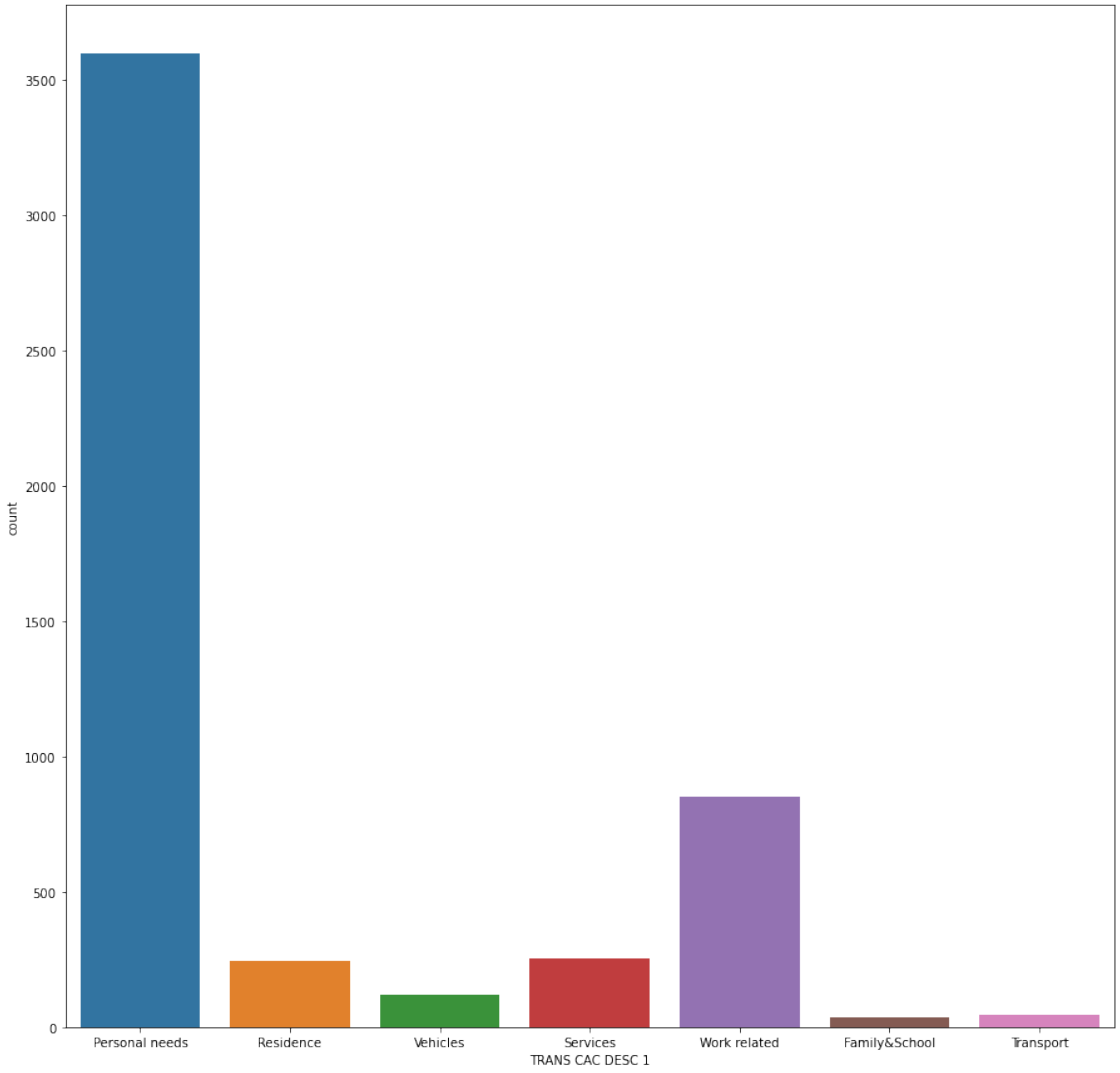
Profiling transactions

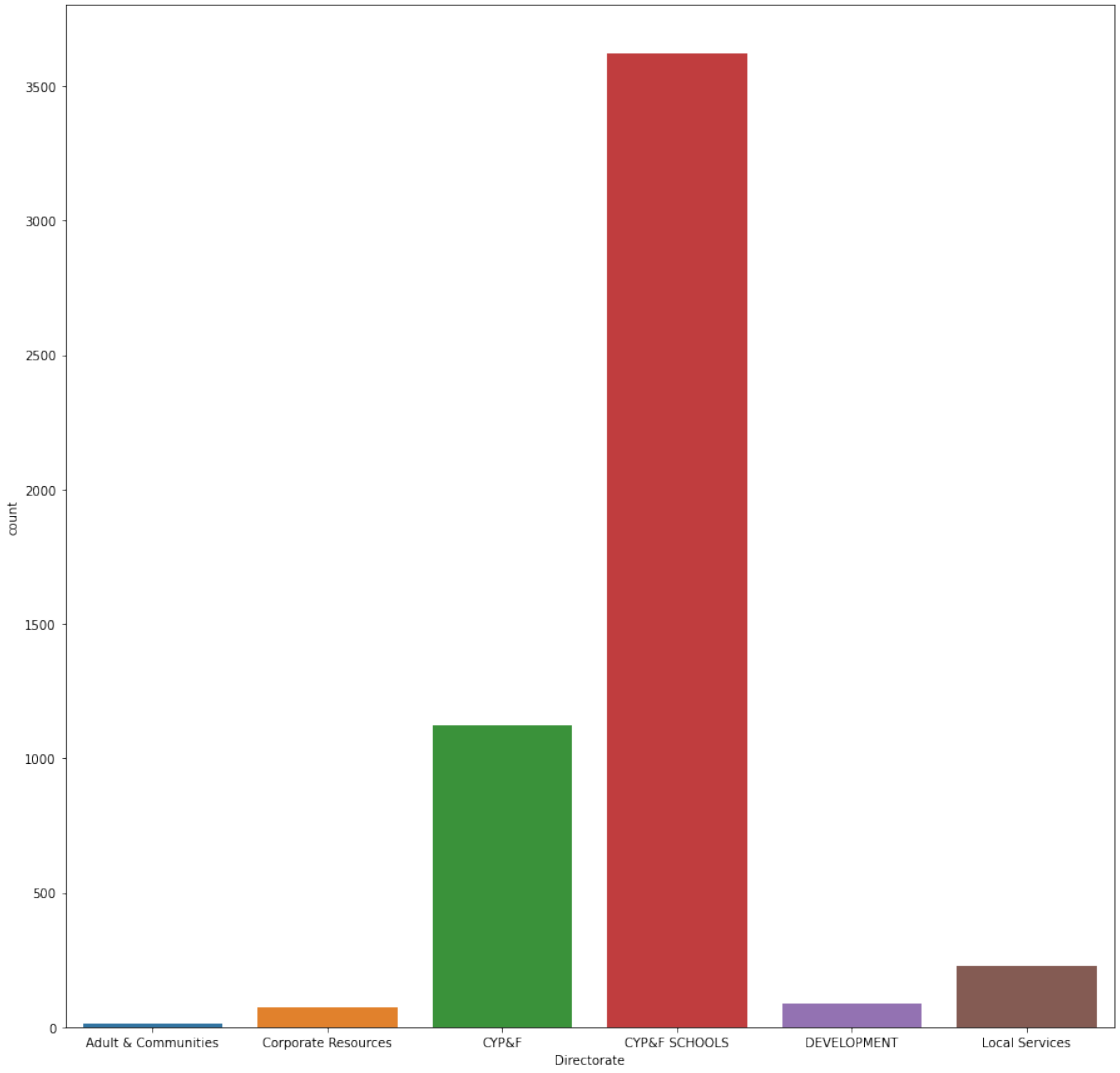
Cluster 1: these are transactions with standard Value added tax, related to vehicles. Are involved with the local services directorate. A bit about Adult & Communities.

We analyze cluster 2:

[175]:
```
analyze_cluster(features,df2014_kmodes8,2)
```


3000
2500
2000
1500
1000
500
0
count
VR
VZ
other
TRANS VAT DESC


3500
3000
2500
2000
1500
1000
500
0
count
Personal needs
Residence
Vehicles
Services
Work related
Family&School
Transport
TRANS CAC DESC 1

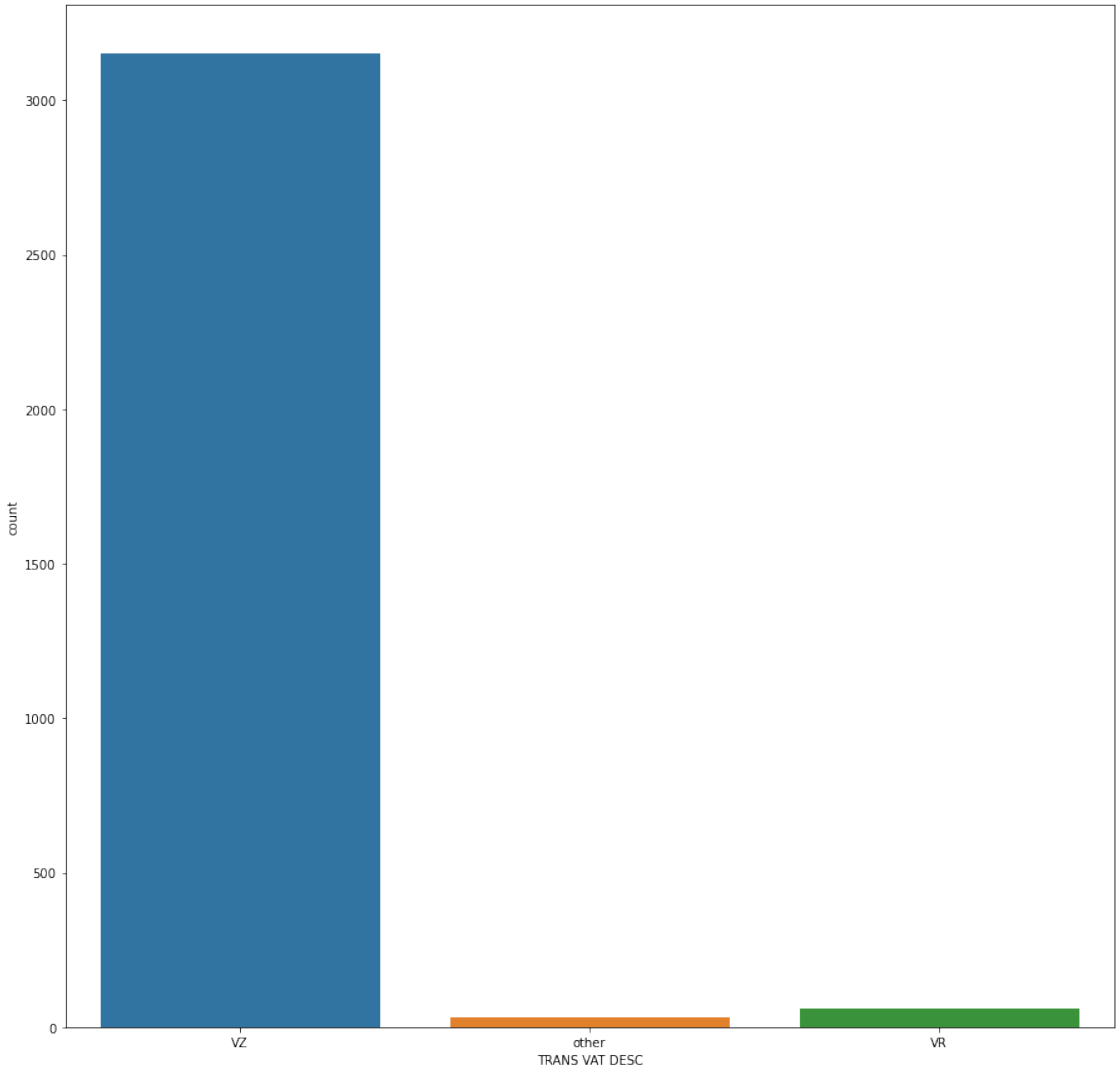


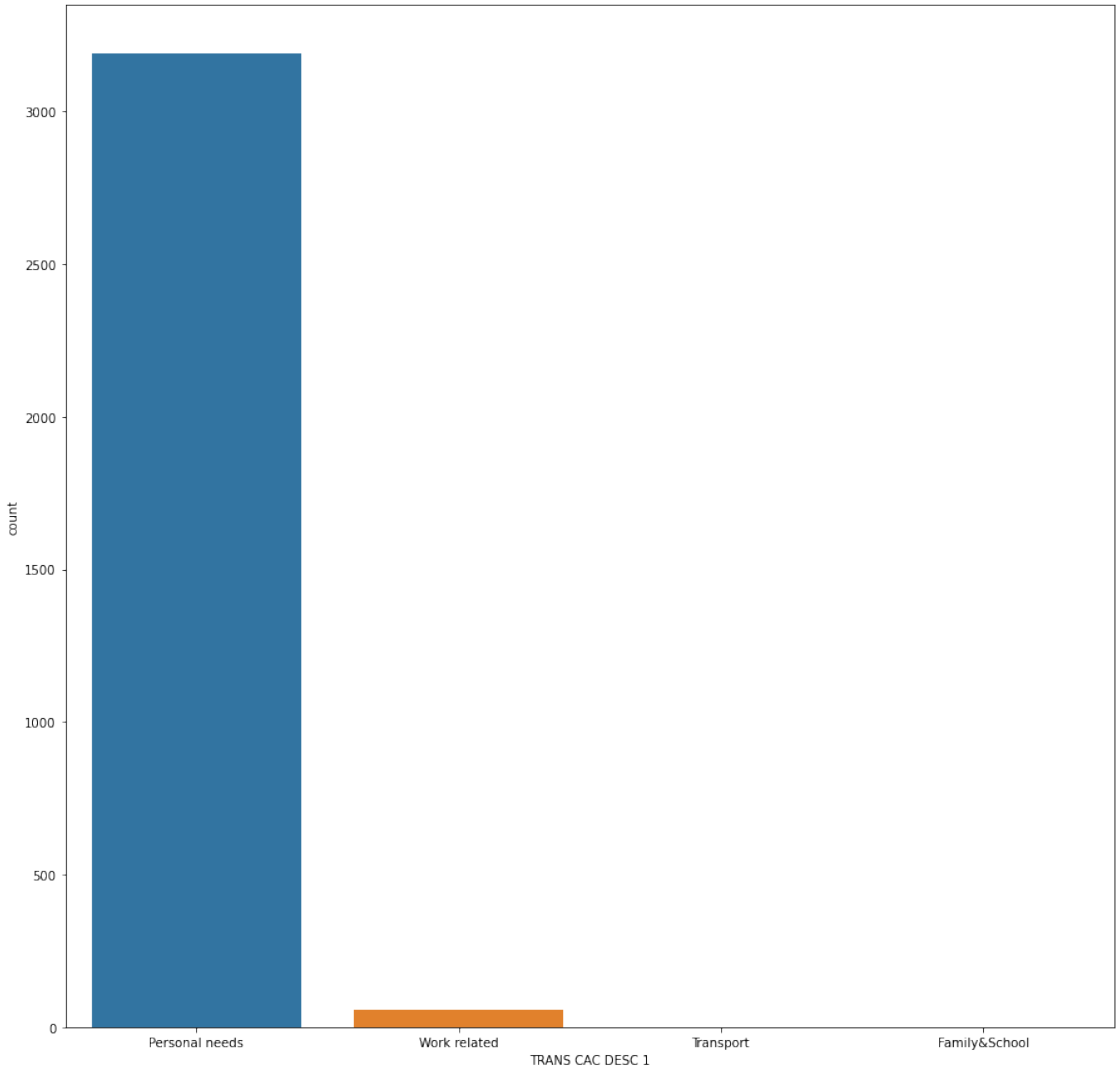
Profiling transactions

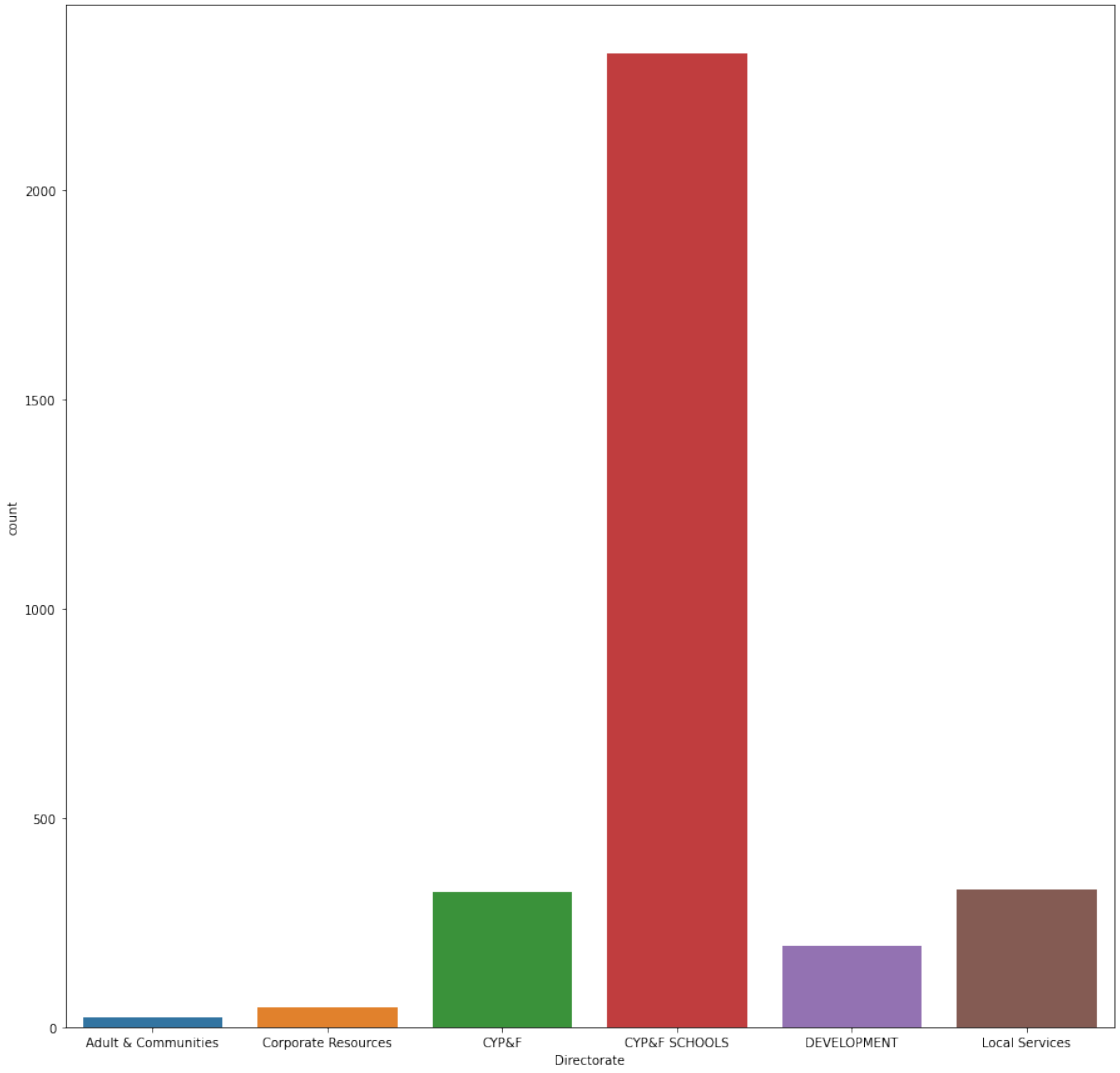
Cluster 2: these are transactions with standard Value added tax and some considerable amount of zero value added tax. Are related to personal needs. Are involved with the CY&PF Schools directorate.

We analyze cluster 3:

[176]:
```
analyze_cluster(features,df2014_kmodes8,3)
```


3000
2500
2000
1500
1000
500
0
count
VZ
other
VR
TRANS VAT DESC


3000
2500
2000
1500
1000
500
0
count
Personal needs
Work related
Transport
Family&School
TRANS CAC DESC 1

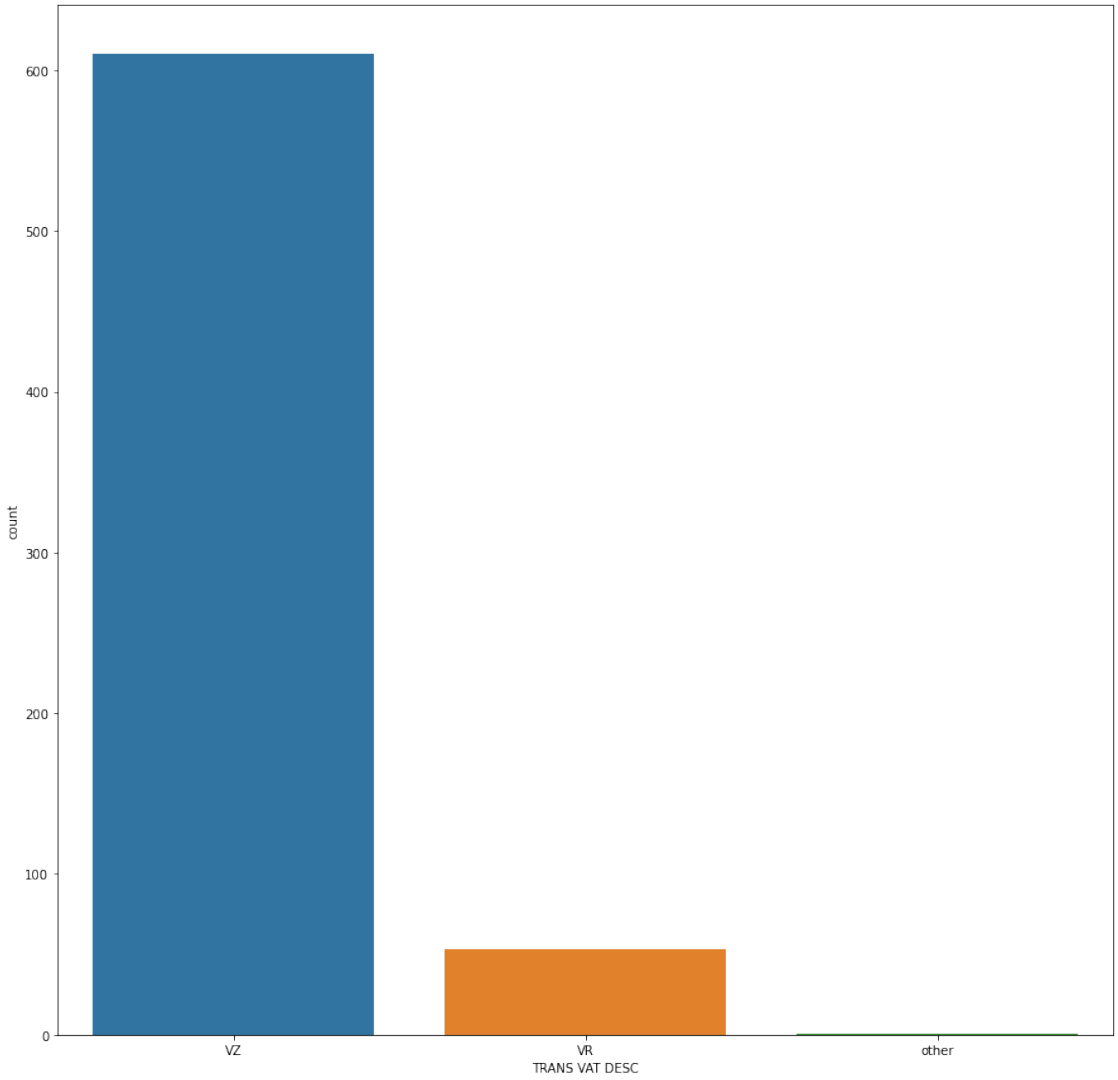


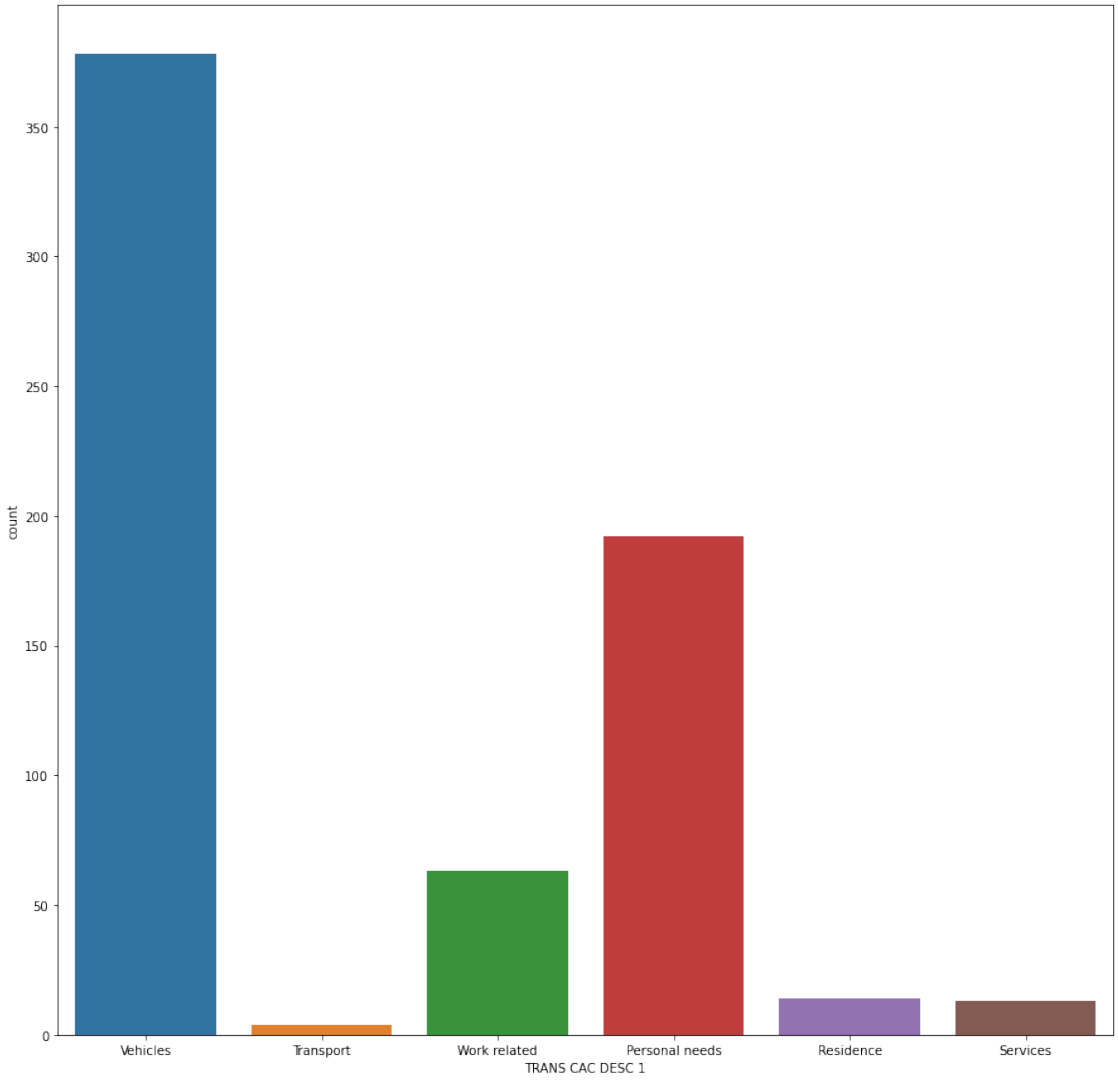
Profiling transactions

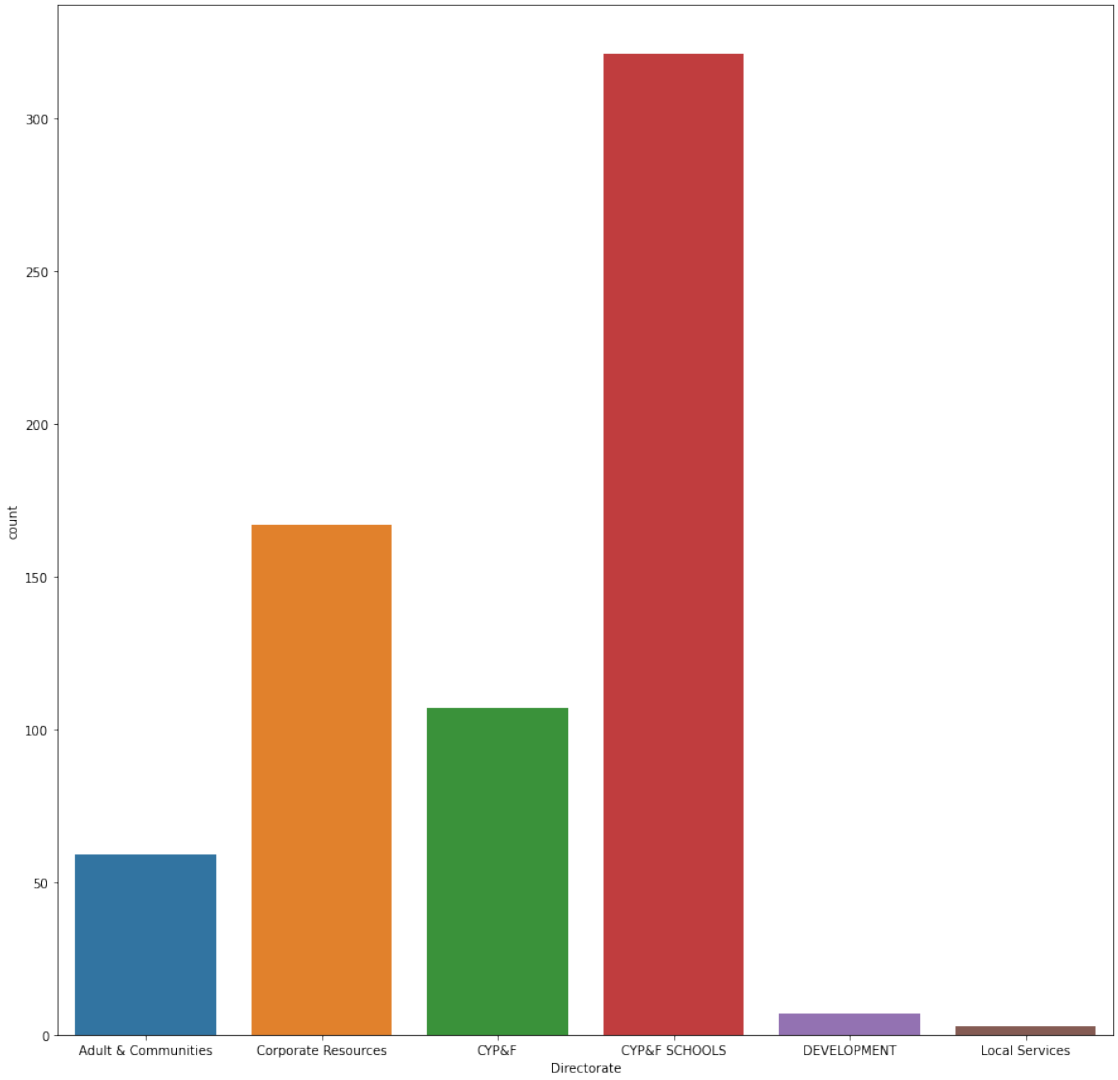
Cluster 3: these are transactions with zero Value added tax, related to personal needs. Are involved with the CY&PF Schools directorate.

We analyze cluster 4:

[177]:
```
analyze_cluster(features,df2014_kmodes8,4)
```


600
500
400
300
200
100
0
count
VZ
VR
other
TRANS VAT DESC


350
300
250
200
150
100
50
0
count
Vehicles
Transport
Work related
Personal needs
Residence
Services
TRANS CAC DESC 1

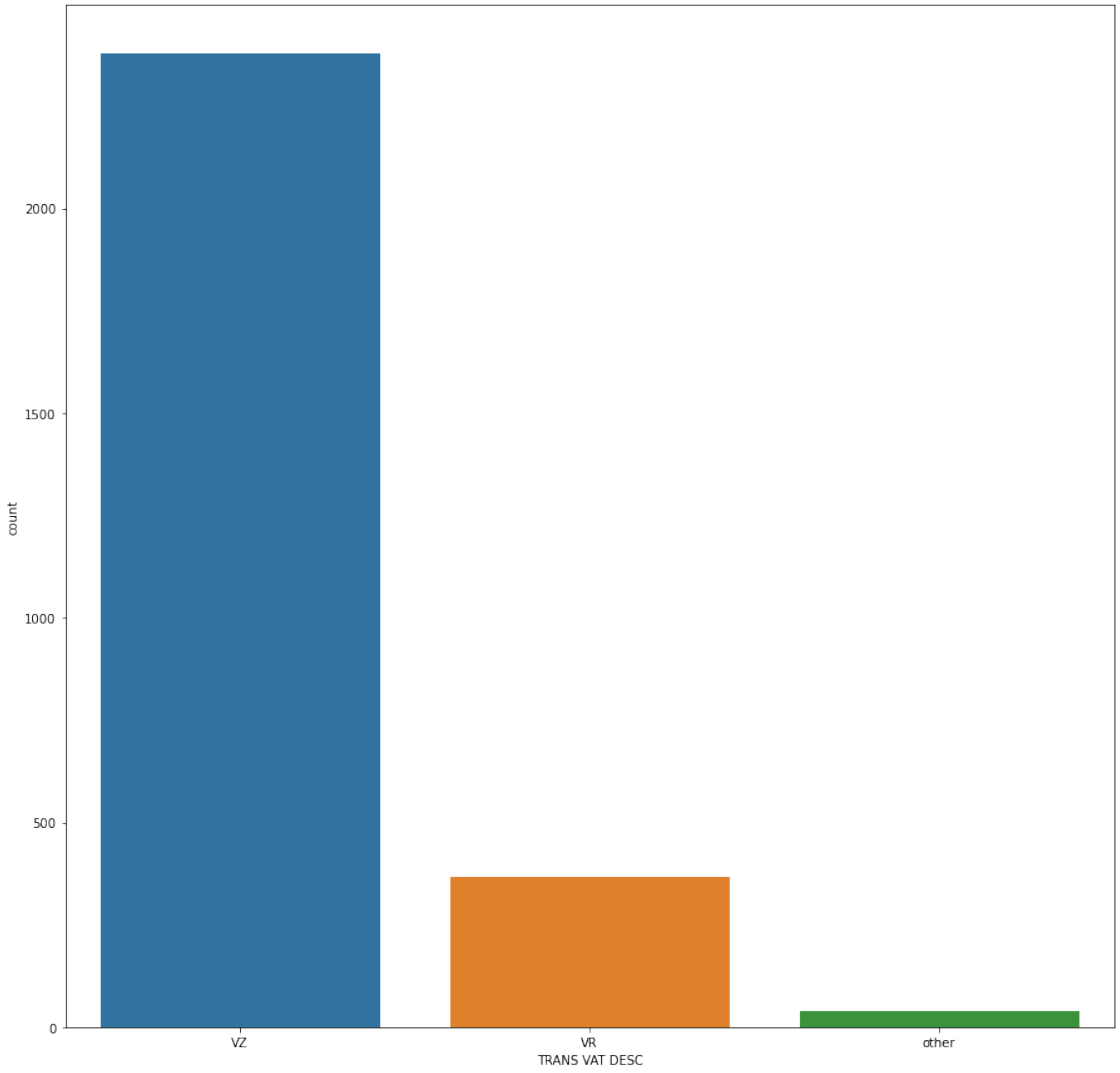


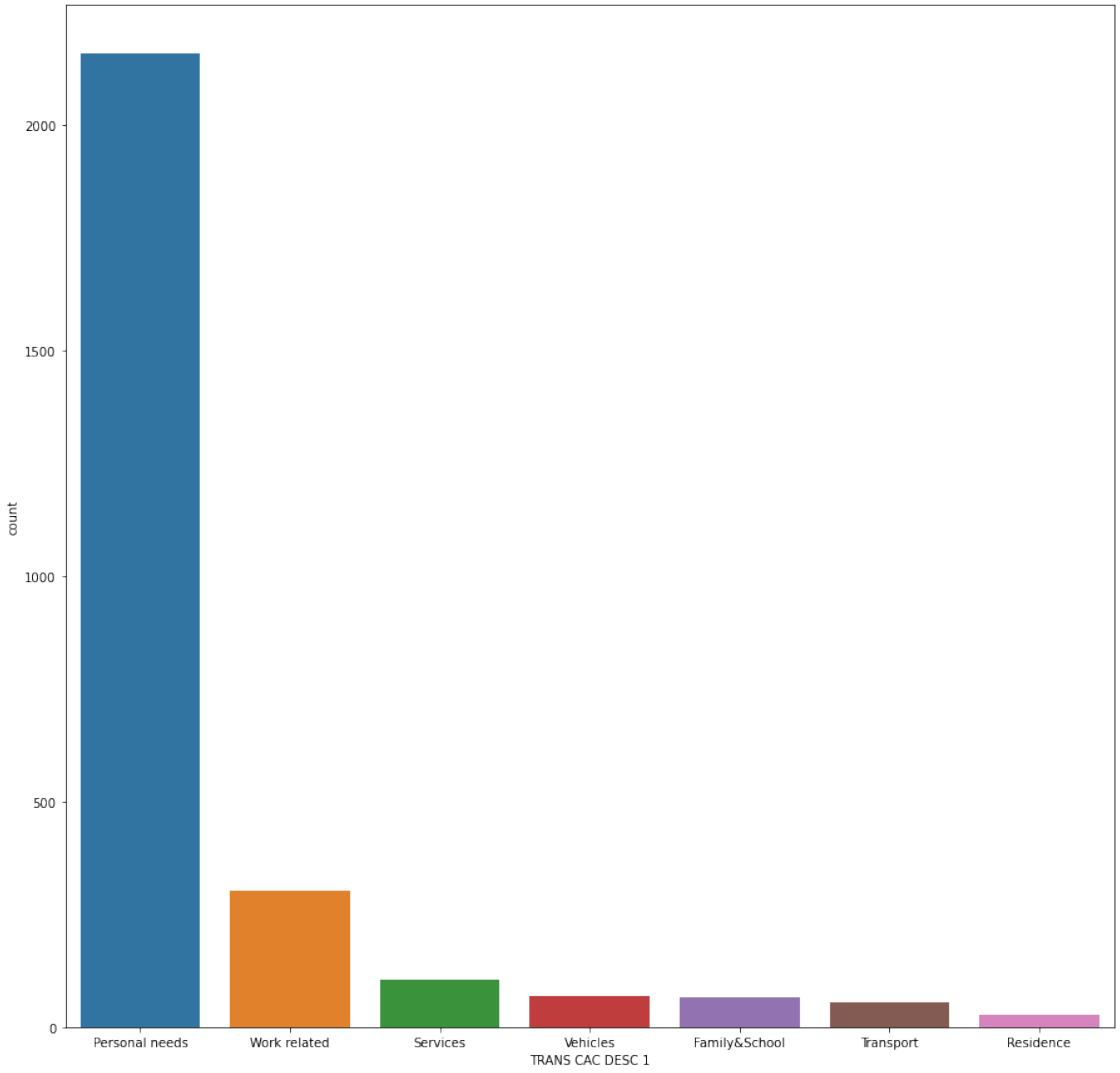
Profiling transactions

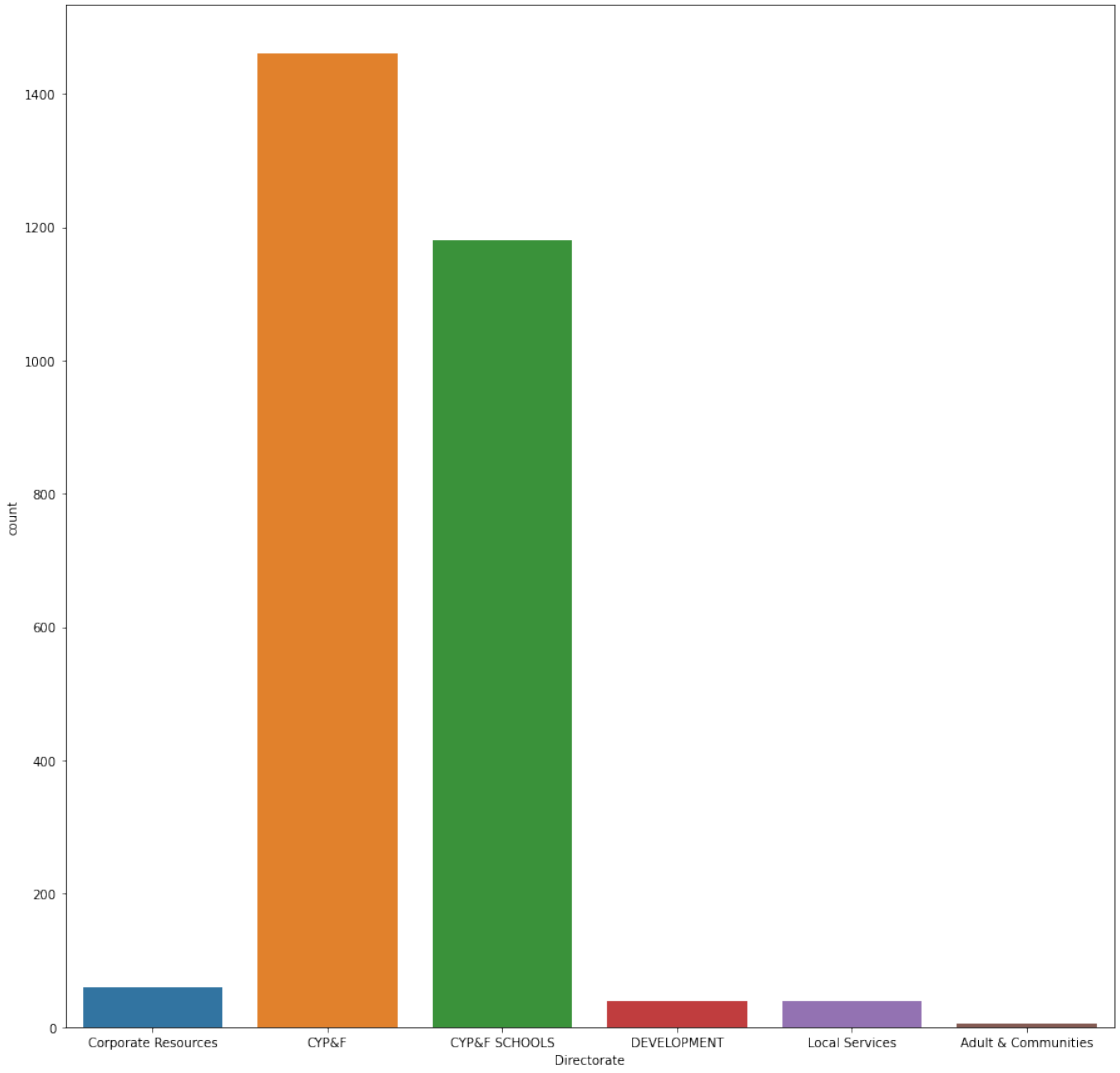
Cluster 4: these are transactions with zero Value added tax, related to vehicles and a considerable amount of personal needs purchases. Are involved with the CY&PF Schools and Corporate Resources directorates.

We analyze cluster 5:

[178]:
```
analyze_cluster(features,df2014_kmodes8,5)
```


2000
1500
1000
500
0
count
VZ
VR
other
TRANS VAT DESC


2000
1500
1000
500
0
count
Personal needs
Work related
Services
Vehicles
Family&School
Transport
Residence
TRANS CAC DESC 1

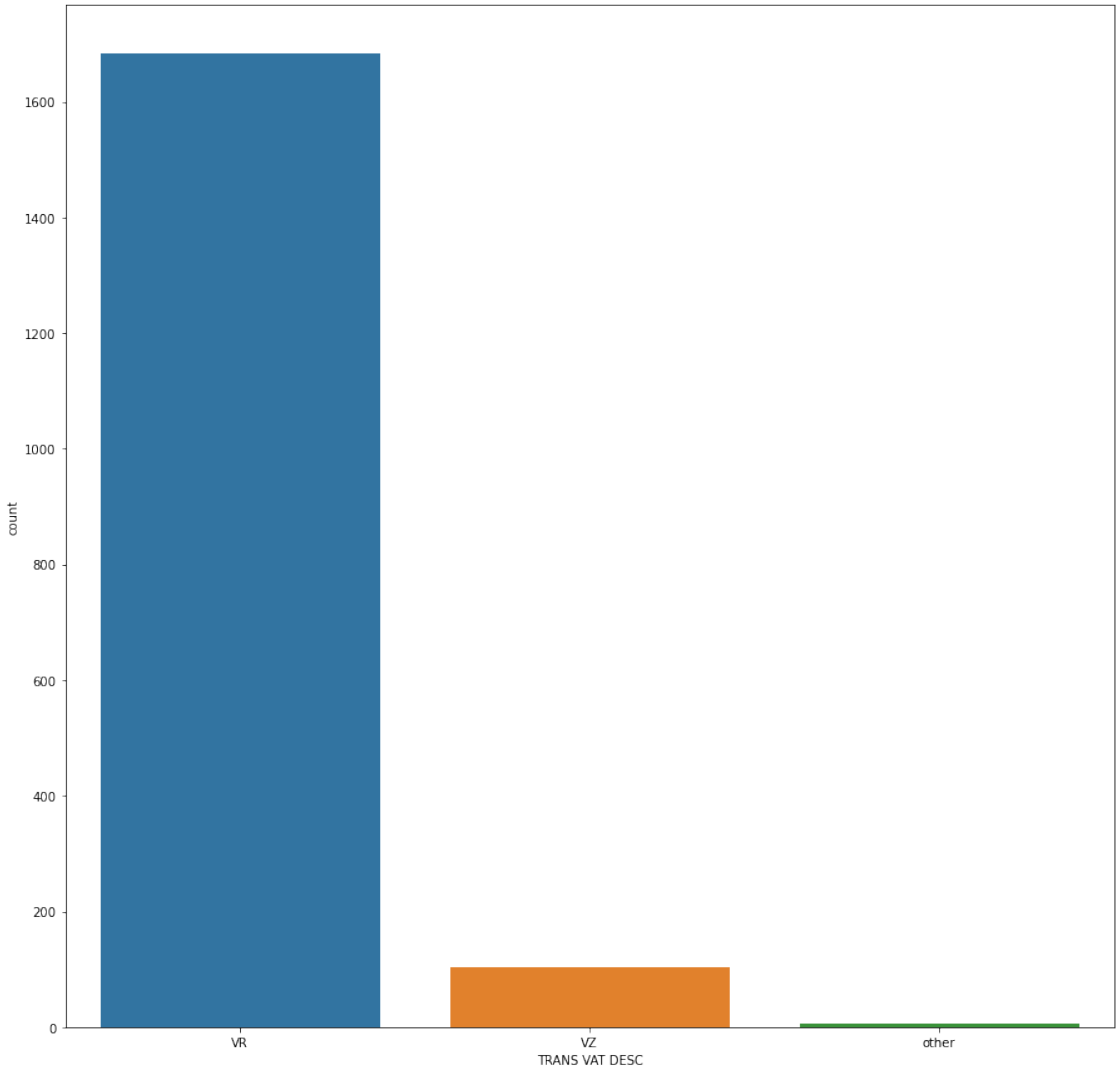


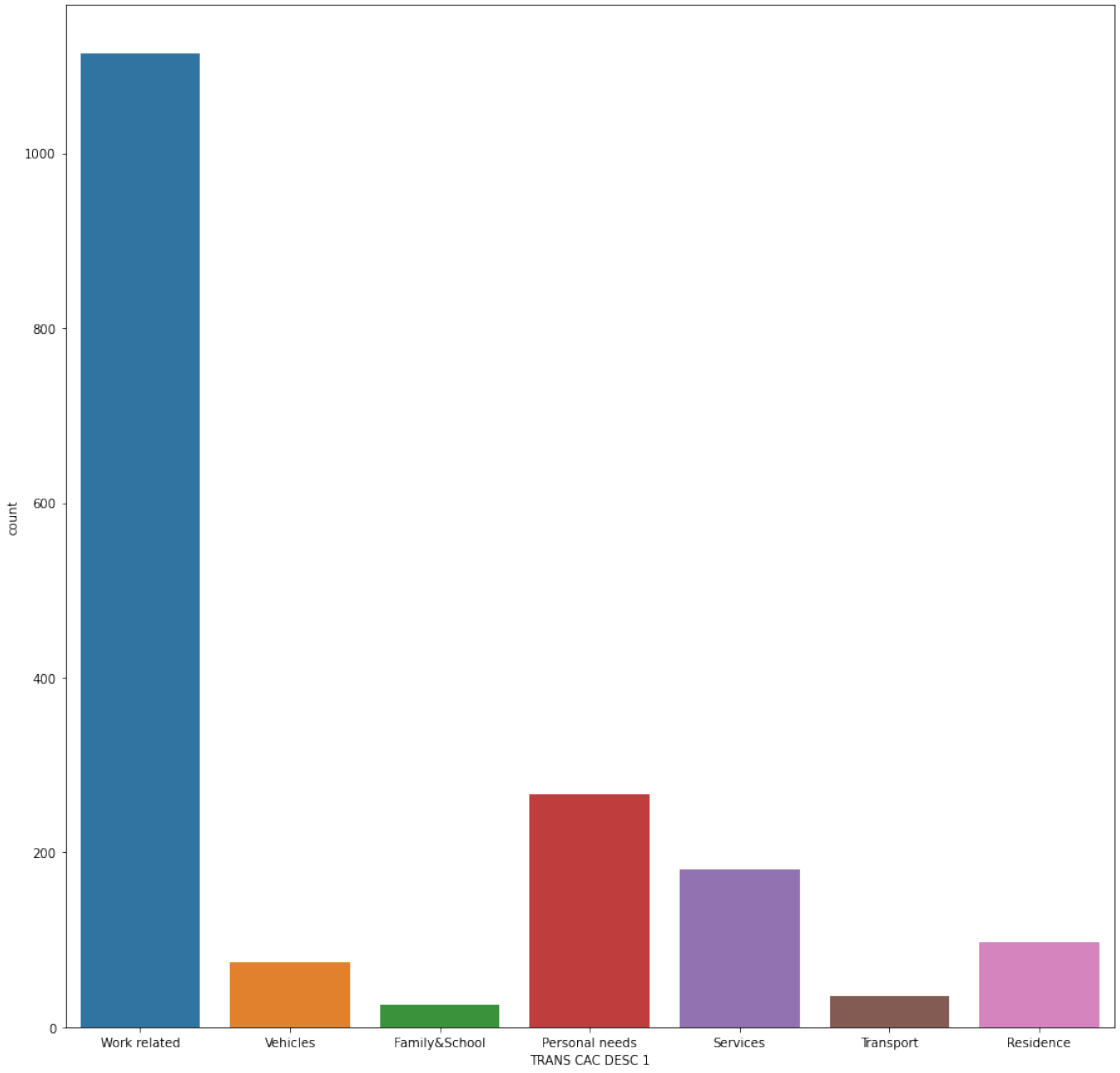
Profiling transactions

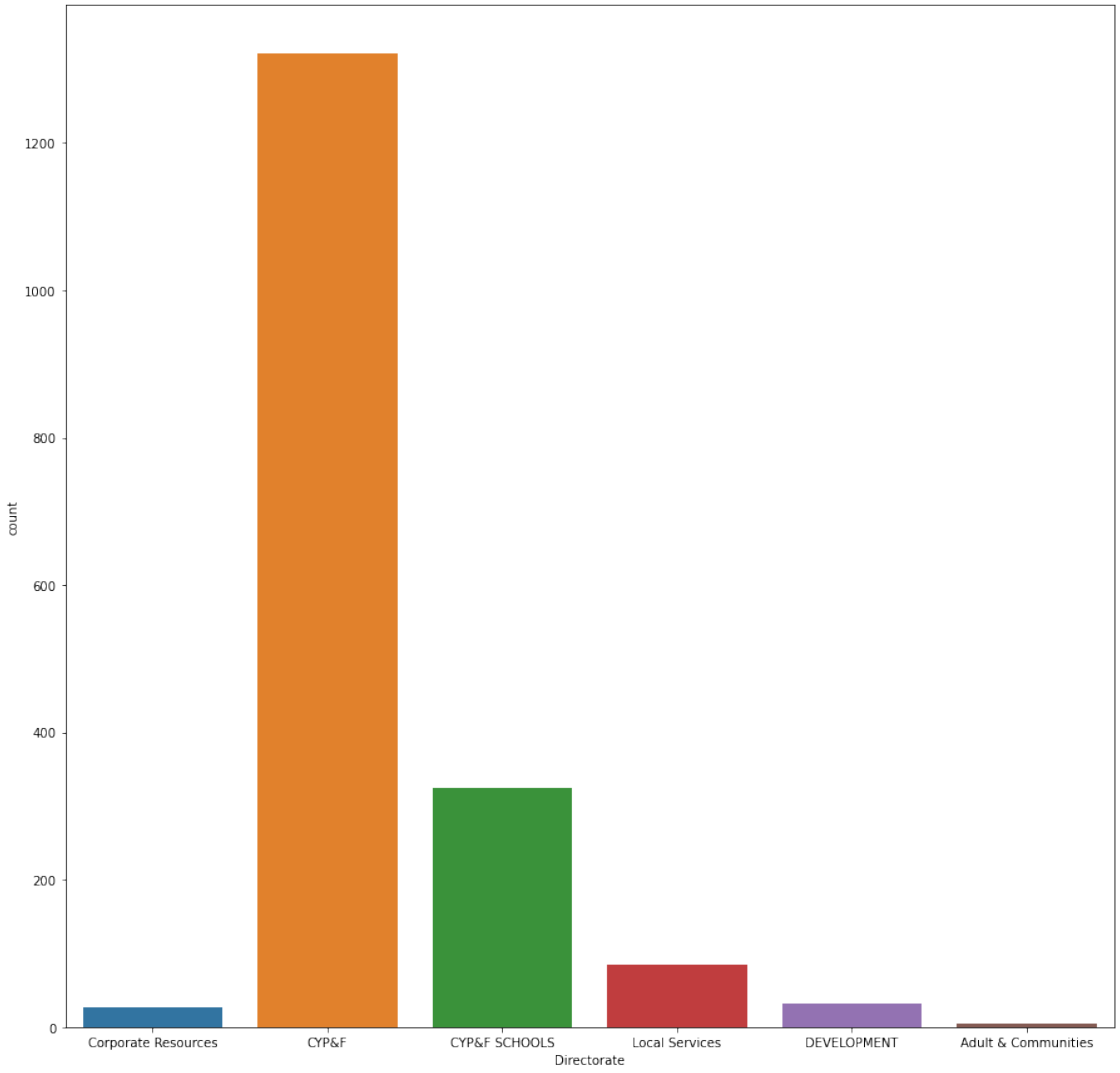
Cluster 5: these are transactions with zero Value added tax, related to personal needs. Are involved with the CY&PF Schools and CY&PF directorates.

We analyze cluster 6:

[179]:
```
analyze_cluster(features,df2014_kmodes8,6)
```


1600
1400
1200
1000
800
600
400
200
0
count
VR
VZ
other
TRANS VAT DESC


1000
800
600
400
200
0
count
Work related
Vehicles
Family&School
Personal needs
Services
Transport
Residence
TRANS CAC DESC 1

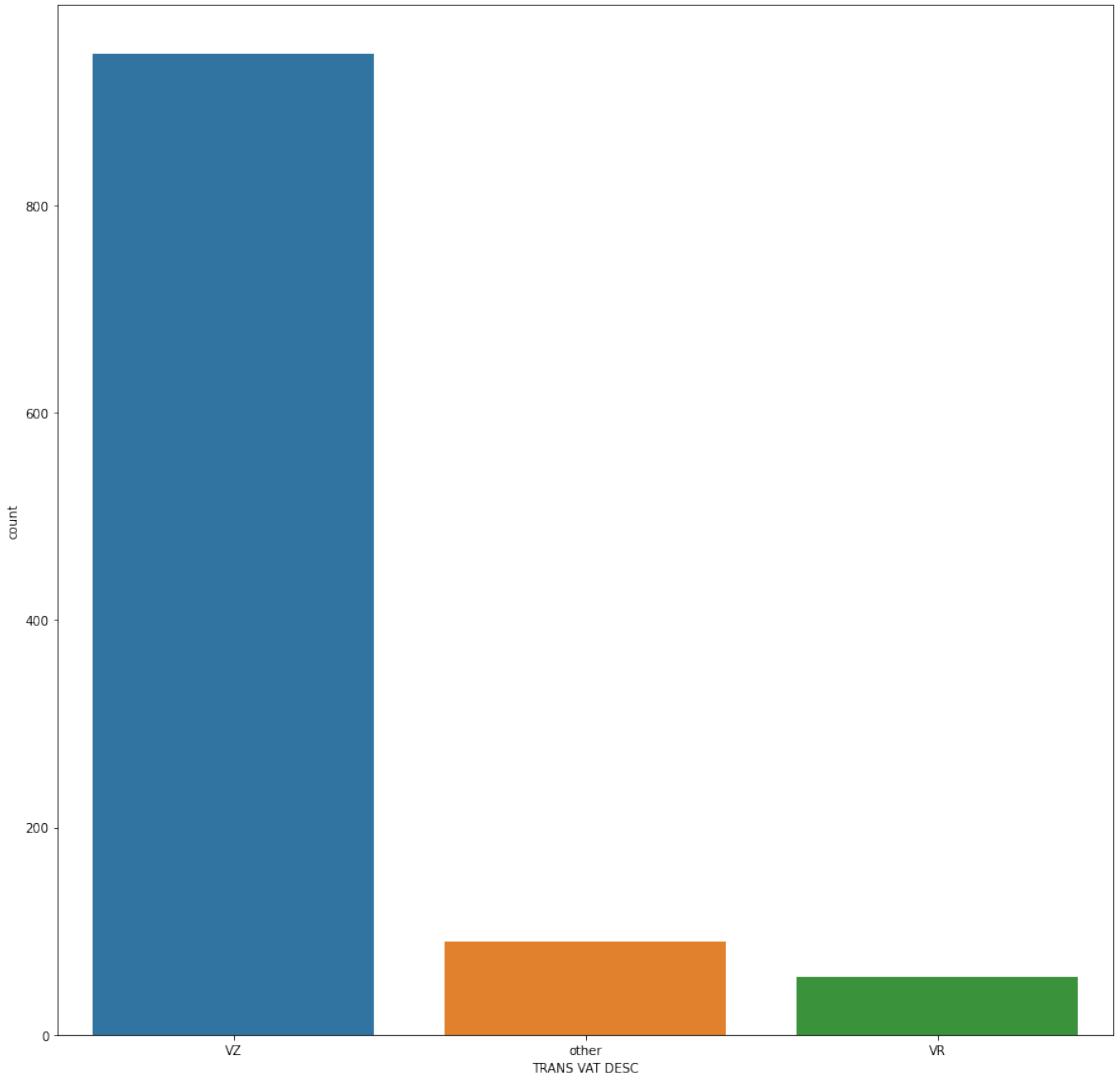


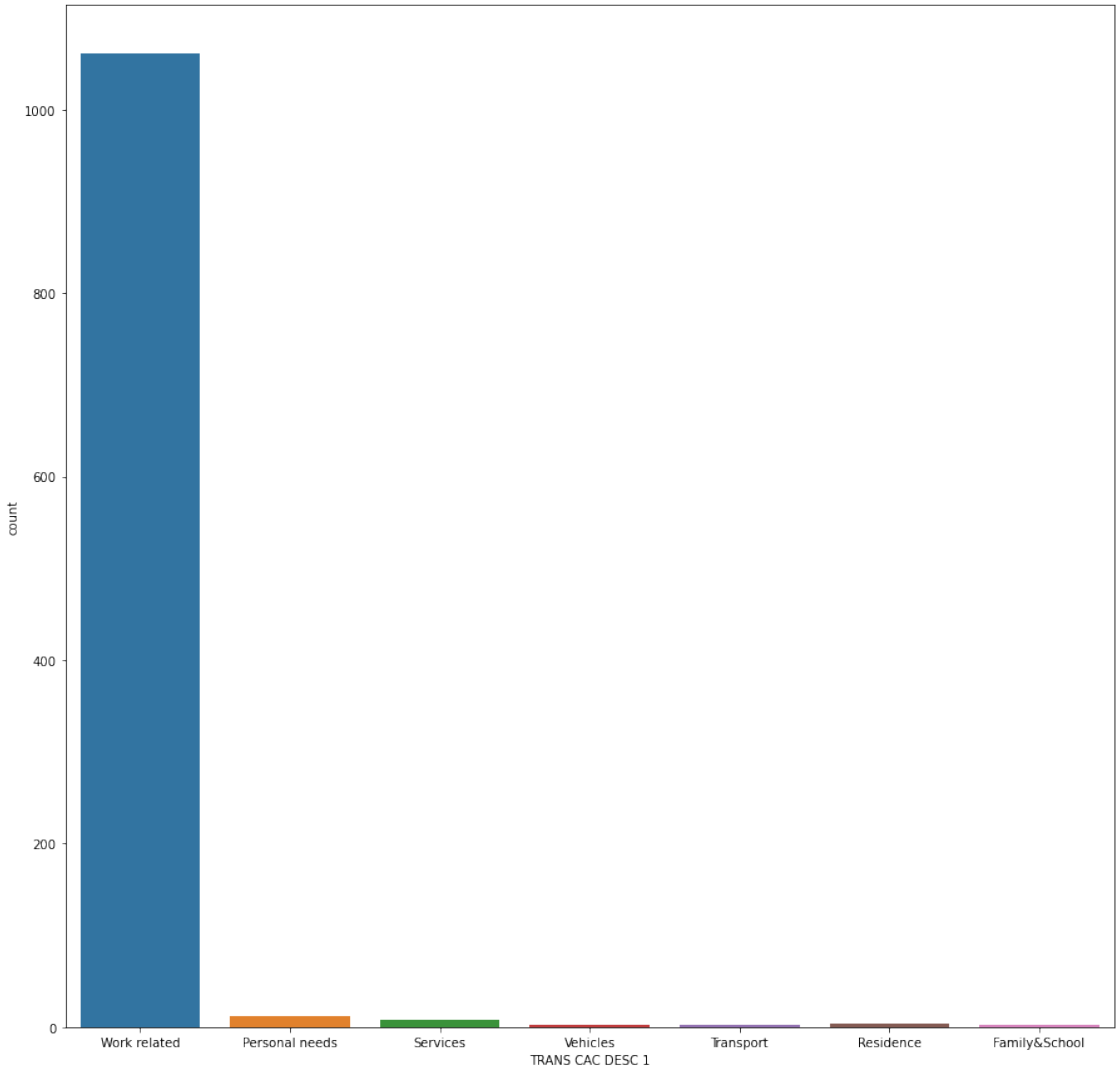
Profiling transactions

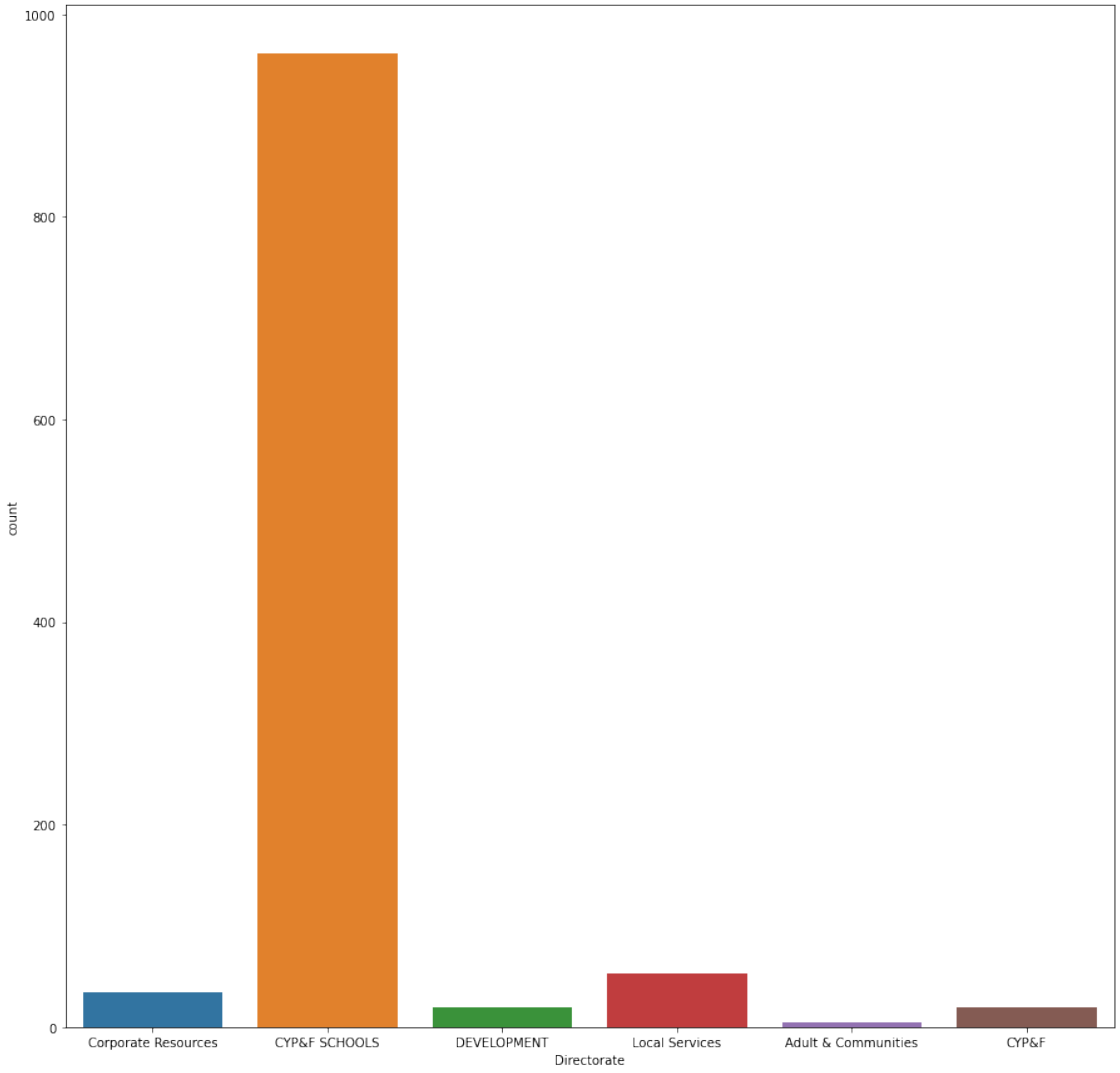
Cluster 6: these are transactions with standard Value added tax, work related and some personal needs and services purchases. Are involved with the CY&PF directorate.

We analyze cluster 7:

[180]:
```
analyze_cluster(features,df2014_kmodes8,7)
```


800
600
400
200
0
count
VZ
other
VR
TRANS VAT DESC


1000
800
600
400
200
0
count
Work related
Personal needs
Services
Vehicles
Transport
Residence
Family&School
TRANS CAC DESC 1



Profiling transactions

Cluster 7: these are transactions with zero Value added tax, work related. Are involved with the CY&PF Schools directorate.

1.4.9 4.3 Forecasting

We load the dataframe of the year 2014.

[181]:
```
data2014_fcst = df2014_pkl_loaded.copy()
data2014_fcst.set_index('TRANS DATE',inplace=True)
```

We reset the index.

[182]:
```
data2014_fcst.reset_index(inplace=True)
```

We define a function to get a list of dataframes associated to a particular set of clients.

[183]:
```
def get_time_series(data,clients):
    time_series = list()
    for client in clients:
        df = pd.DataFrame()
        df = data[data['CARD NUMBER'] == client]
        df.columns = data.columns
        time_series.append(df)
    return time_series
```

We group by clients and define those customers that have more or equal than 250 transactions in the year 2014

[184]:
```
clients = data2014_fcst.groupby('CARD NUMBER').agg(transactions=('CARD NUMBER',␣
↪'count'))
selected_clients = clients[(clients['transactions'] >= 250)].index
selected_clients
```

[184]:
```
Index(['************2771', '************5412', '************6678'],
dtype='object', name='CARD NUMBER')
```

We define the data as the index. And we establlish the features that we are going to use for forecasting.

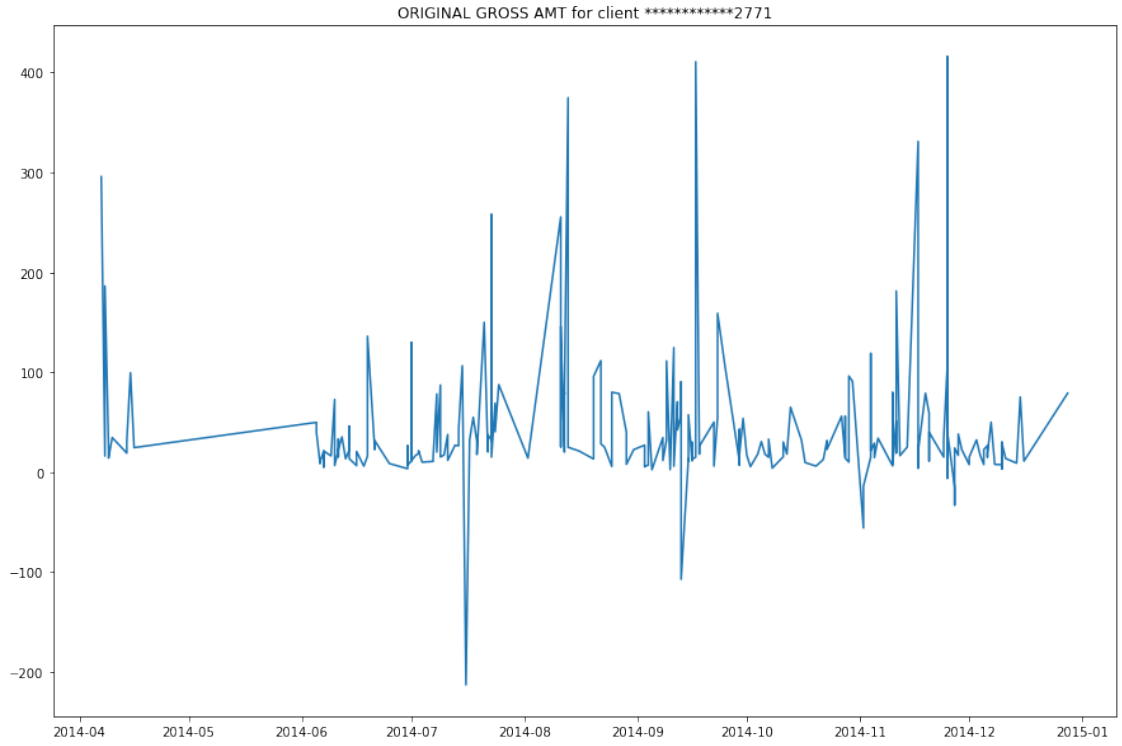
[185]:
```
forecasting = data2014_fcst[['TRANS DATE','CARD NUMBER','ORIGINAL GROSS AMT']]
forecasting.set_index('TRANS DATE',inplace=True,drop=False)
forecasting.sort_index(inplace=True)
```

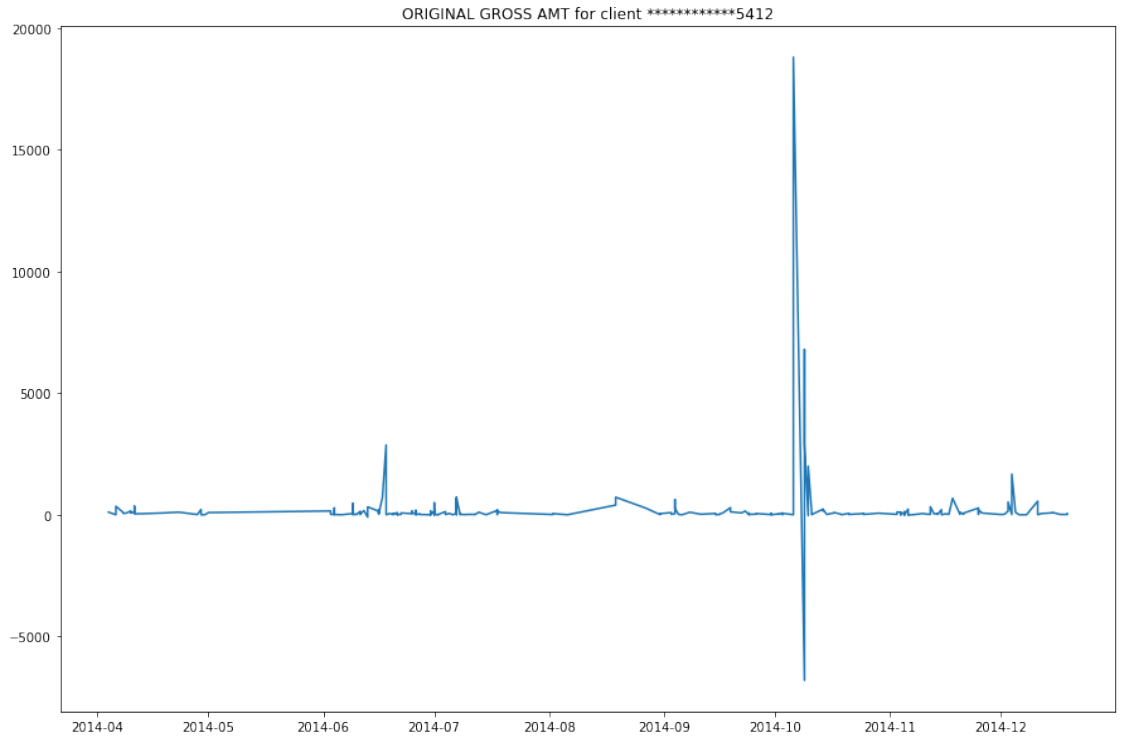
We get the list of dataframes associated to a particular group of clients that fulfill some requisites.

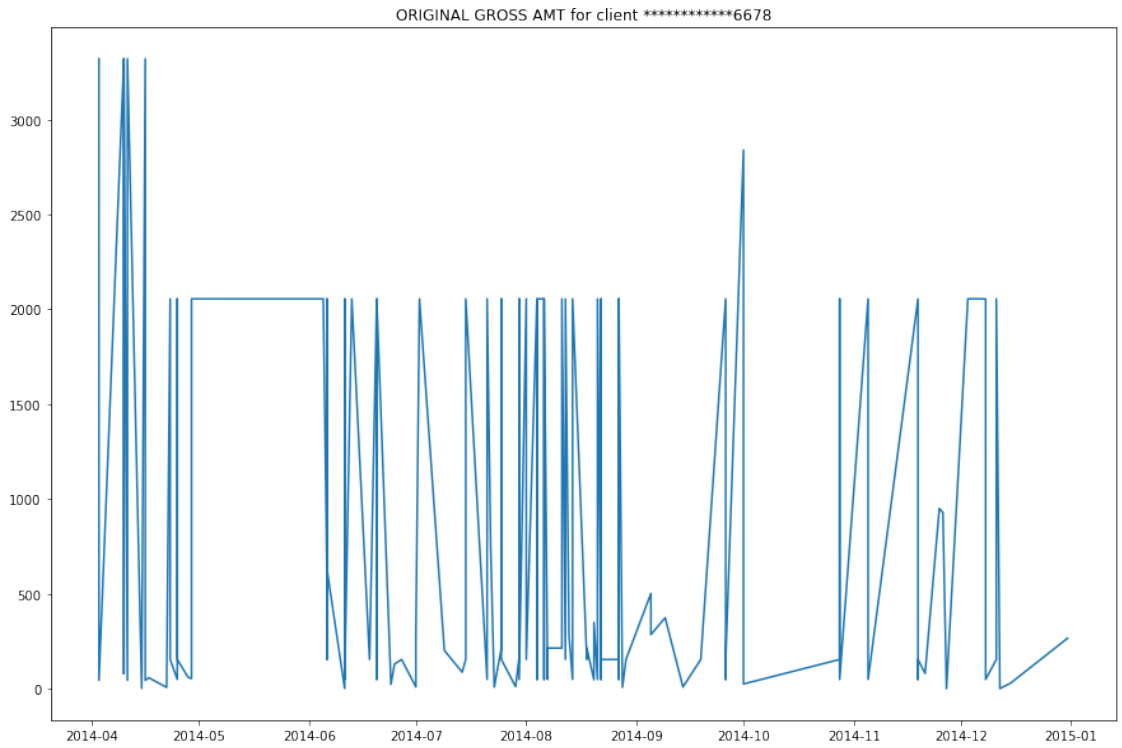
[186]:
```
ts = get_time_series(forecasting,selected_clients)
```

We plot the selected time series (3).

[187]:
```
for i in range(len(ts)):
    plt.figure(figsize=(15,10))
    plt.plot(ts[i].index,ts[i]['ORIGINAL GROSS AMT'])
    plt.title(f"ORIGINAL GROSS AMT for client {selected_clients[i]}")
    plt.show()
```


ORIGINAL GROSS AMT for client **********2771
400
300
200
100
0
−100
−200
2014-04
2014-05
2014-06
2014-07
2014-08
2014-09
2014-10
2014-11
2014-12
2015-01


ORIGINAL GROSS AMT for client **********5412
20000
15000
10000
5000
0
−5000
2014-04
2014-05
2014-06
2014-07
2014-08
2014-09
2014-10
2014-11
2014-12



We split the data into test and training. We cannot use the train_test_split function because we need the data sorted by date.

[188]:
```
start_date = '2014-10-11'
train_ts0 = ts[0].loc[ts[0].index < pd.to_datetime(start_date)]['ORIGINAL GROSS␣
 ↪AMT']
test_ts0 = ts[0].loc[ts[0].index >= pd.to_datetime(start_date)]['ORIGINAL GROSS␣
 ↪AMT']
train_ts1 = ts[1].loc[ts[1].index < pd.to_datetime(start_date)]['ORIGINAL GROSS␣
 ↪AMT']
test_ts1 = ts[1].loc[ts[1].index >= pd.to_datetime(start_date)]['ORIGINAL GROSS␣
 ↪AMT']
train_ts2 = ts[2].loc[ts[2].index < pd.to_datetime(start_date)]['ORIGINAL GROSS␣
 ↪AMT']
test_ts2 = ts[2].loc[ts[2].index >= pd.to_datetime(start_date)]['ORIGINAL GROSS␣
 ↪AMT']
```

In order to find the best values of p (number of lag observations included in the model), q (size of the moving average window) and d (degree of differencing) that will minimize the error, we are going to use the auto_arima function. This comes from the pmdarima. That is how we will be able to find the optimal parameter combintation and return the best model for the 2014 dataframe.

We will use the stepwise algorithm (Hyndman and Khandakar) to identify the optimal model parameters.

We will fit 50 Arima models.

[189]:
```
model0 = pm.auto_arima(train_ts0, start_p=0, start_q=1,
                       max_p=5, max_d=5, max_q=5, start_P=0,
                       D=1, start_Q=0, max_P=5, max_D=5,
                       max_Q=5, m=12, seasonal=True, error_action='warn',
                       trace=True, suppress_warnings=True, stepwise=True,
                       random_state=42, n_fits=50)

print(model0.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(0,0,1)(0,1,0)[12] intercept   : AIC=2156.252, Time=0.12 sec
 ARIMA(0,0,0)(0,1,0)[12] intercept   : AIC=2154.276, Time=0.01 sec
 ARIMA(1,0,0)(1,1,0)[12] intercept   : AIC=2109.104, Time=0.19 sec
 ARIMA(0,0,1)(0,1,1)[12] intercept   : AIC=inf, Time=0.30 sec
 ARIMA(0,0,0)(0,1,0)[12]             : AIC=2152.510, Time=0.01 sec
 ARIMA(1,0,0)(0,1,0)[12] intercept   : AIC=2156.247, Time=0.08 sec
 ARIMA(1,0,0)(2,1,0)[12] intercept   : AIC=2092.157, Time=0.66 sec
 ARIMA(1,0,0)(3,1,0)[12] intercept   : AIC=2081.789, Time=1.58 sec
 ARIMA(1,0,0)(4,1,0)[12] intercept   : AIC=2071.575, Time=3.20 sec
 ARIMA(1,0,0)(5,1,0)[12] intercept   : AIC=2073.423, Time=5.15 sec
 ARIMA(1,0,0)(4,1,1)[12] intercept   : AIC=2072.546, Time=4.02 sec
 ARIMA(1,0,0)(3,1,1)[12] intercept   : AIC=2070.617, Time=2.62 sec
 ARIMA(1,0,0)(2,1,1)[12] intercept   : AIC=2069.100, Time=1.35 sec
 ARIMA(1,0,0)(1,1,1)[12] intercept   : AIC=inf, Time=0.44 sec
 ARIMA(1,0,0)(2,1,2)[12] intercept   : AIC=inf, Time=2.49 sec
 ARIMA(1,0,0)(1,1,2)[12] intercept   : AIC=inf, Time=1.50 sec
 ARIMA(1,0,0)(3,1,2)[12] intercept   : AIC=inf, Time=2.71 sec
 ARIMA(0,0,0)(2,1,1)[12] intercept   : AIC=2067.485, Time=0.81 sec
 ARIMA(0,0,0)(1,1,1)[12] intercept   : AIC=inf, Time=0.36 sec
 ARIMA(0,0,0)(2,1,0)[12] intercept   : AIC=2090.742, Time=0.60 sec
 ARIMA(0,0,0)(3,1,1)[12] intercept   : AIC=2069.055, Time=2.08 sec
 ARIMA(0,0,0)(2,1,2)[12] intercept   : AIC=inf, Time=1.94 sec
 ARIMA(0,0,0)(1,1,0)[12] intercept   : AIC=2107.553, Time=0.23 sec
 ARIMA(0,0,0)(1,1,2)[12] intercept   : AIC=inf, Time=1.21 sec
 ARIMA(0,0,0)(3,1,0)[12] intercept   : AIC=2081.274, Time=1.47 sec
 ARIMA(0,0,0)(3,1,2)[12] intercept   : AIC=inf, Time=3.25 sec
 ARIMA(0,0,1)(2,1,1)[12] intercept   : AIC=2069.182, Time=1.35 sec
 ARIMA(1,0,1)(2,1,1)[12] intercept   : AIC=2066.625, Time=2.53 sec
 ARIMA(1,0,1)(1,1,1)[12] intercept   : AIC=2065.188, Time=0.98 sec
 ARIMA(1,0,1)(0,1,1)[12] intercept   : AIC=inf, Time=1.10 sec
 ARIMA(1,0,1)(1,1,0)[12] intercept   : AIC=2106.037, Time=0.37 sec
 ARIMA(1,0,1)(1,1,2)[12] intercept   : AIC=inf, Time=3.30 sec
 ARIMA(1,0,1)(0,1,0)[12] intercept   : AIC=2155.136, Time=0.19 sec
 ARIMA(1,0,1)(0,1,2)[12] intercept   : AIC=inf, Time=2.35 sec
 ARIMA(1,0,1)(2,1,0)[12] intercept   : AIC=2088.068, Time=1.70 sec
 ARIMA(1,0,1)(2,1,2)[12] intercept   : AIC=inf, Time=2.71 sec
```

```
 ARIMA(0,0,1)(1,1,1)[12] intercept   : AIC=inf, Time=0.72 sec
 ARIMA(2,0,1)(1,1,1)[12] intercept   : AIC=inf, Time=1.19 sec
 ARIMA(1,0,2)(1,1,1)[12] intercept   : AIC=inf, Time=1.01 sec
 ARIMA(0,0,2)(1,1,1)[12] intercept   : AIC=inf, Time=0.80 sec
 ARIMA(2,0,0)(1,1,1)[12] intercept   : AIC=inf, Time=0.59 sec
 ARIMA(2,0,2)(1,1,1)[12] intercept   : AIC=2066.820, Time=0.90 sec
 ARIMA(1,0,1)(1,1,1)[12]             : AIC=2063.392, Time=1.08 sec
 ARIMA(1,0,1)(0,1,1)[12]             : AIC=inf, Time=0.53 sec
 ARIMA(1,0,1)(1,1,0)[12]             : AIC=2104.121, Time=0.21 sec
 ARIMA(1,0,1)(2,1,1)[12]             : AIC=inf, Time=2.71 sec
 ARIMA(1,0,1)(1,1,2)[12]             : AIC=inf, Time=2.97 sec
 ARIMA(1,0,1)(0,1,0)[12]             : AIC=2153.346, Time=0.12 sec
 ARIMA(1,0,1)(0,1,2)[12]             : AIC=inf, Time=1.14 sec
 ARIMA(1,0,1)(2,1,0)[12]             : AIC=2086.144, Time=0.92 sec
 ARIMA(1,0,1)(2,1,2)[12]             : AIC=inf, Time=2.92 sec
 ARIMA(0,0,1)(1,1,1)[12]             : AIC=2066.195, Time=0.30 sec
 ARIMA(1,0,0)(1,1,1)[12]             : AIC=2066.096, Time=0.31 sec
 ARIMA(2,0,1)(1,1,1)[12]             : AIC=inf, Time=1.11 sec
 ARIMA(1,0,2)(1,1,1)[12]             : AIC=inf, Time=0.44 sec
 ARIMA(0,0,0)(1,1,1)[12]             : AIC=inf, Time=0.28 sec
 ARIMA(0,0,2)(1,1,1)[12]             : AIC=inf, Time=0.33 sec
 ARIMA(2,0,0)(1,1,1)[12]             : AIC=inf, Time=0.52 sec
 ARIMA(2,0,2)(1,1,1)[12]             : AIC=2064.828, Time=0.76 sec

Best model:  ARIMA(1,0,1)(1,1,1)[12]
Total fit time: 76.874 seconds
                                     SARIMAX Results
==========================================================================================
==========
Dep. Variable:                                     y   No. Observations:
194
Model:             SARIMAX(1, 0, 1)x(1, 1, 1, 12)   Log Likelihood
-1026.696
Date:                            Fri, 19 Nov 2021   AIC
2063.392
Time:                                    23:44:25   BIC
2079.412
Sample:                                         0   HQIC
2069.886
                                              - 194
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.9009      0.118      7.643      0.000       0.670       1.132
ma.L1         -0.8239      0.157     -5.243      0.000      -1.132      -0.516
ar.S.L12      -0.0417      0.126     -0.330      0.741      -0.289       0.206
ma.S.L12      -0.8714      0.117     -7.428      0.000      -1.101      -0.641
```

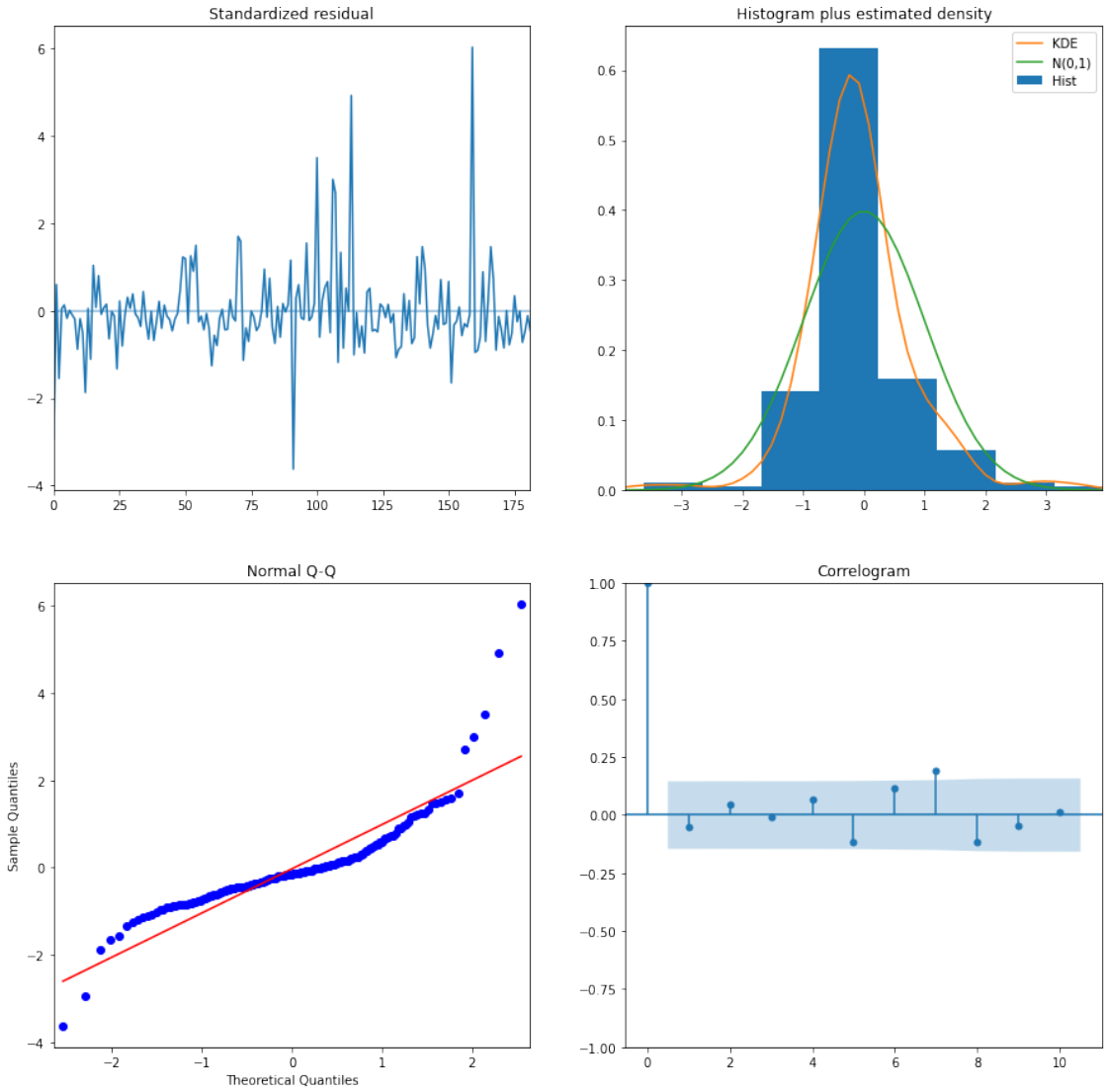
```
sigma2      4124.0438    183.526     22.471      0.000    3764.339    4483.748
==============================================================================
===
Ljung-Box (L1) (Q):                   0.51   Jarque-Bera (JB):
962.74
Prob(Q):                              0.47   Prob(JB):
0.00
Heteroskedasticity (H):               1.81   Skew:
1.96
Prob(H) (two-sided):                  0.02   Kurtosis:
13.56
==============================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```

We save our ARIMA model.

[190]:
```
ts_model_arima_filename = '/Users/andresaristi/Documents/
 ↪BirminghamCardTransactions/pickle_files/ts_model_arima.pkl'
ts_model_arima_pkl = open(ts_model_arima_filename, 'wb')
pickle.dump(model0, ts_model_arima_pkl)
ts_model_arima_pkl.close()
```

The AIC has been reduced. The P-values of the X terms are less the < 0.05 which the exception of one, which is great.

[191]:
```
model0.plot_diagnostics(figsize=(15,15))
plt.show()
```



- The residual errors seem to fluctuate around a mean of zero and have a uniform variance.
- The density plot suggest normal distribution with mean zero.
- All the dots must fall perfectly with the red line. This is not the case although the deviations are not that significant. Any significant deviations would imply the distribution is skewed.
- The Correlogram shows the residual errors are not autocorrelated.

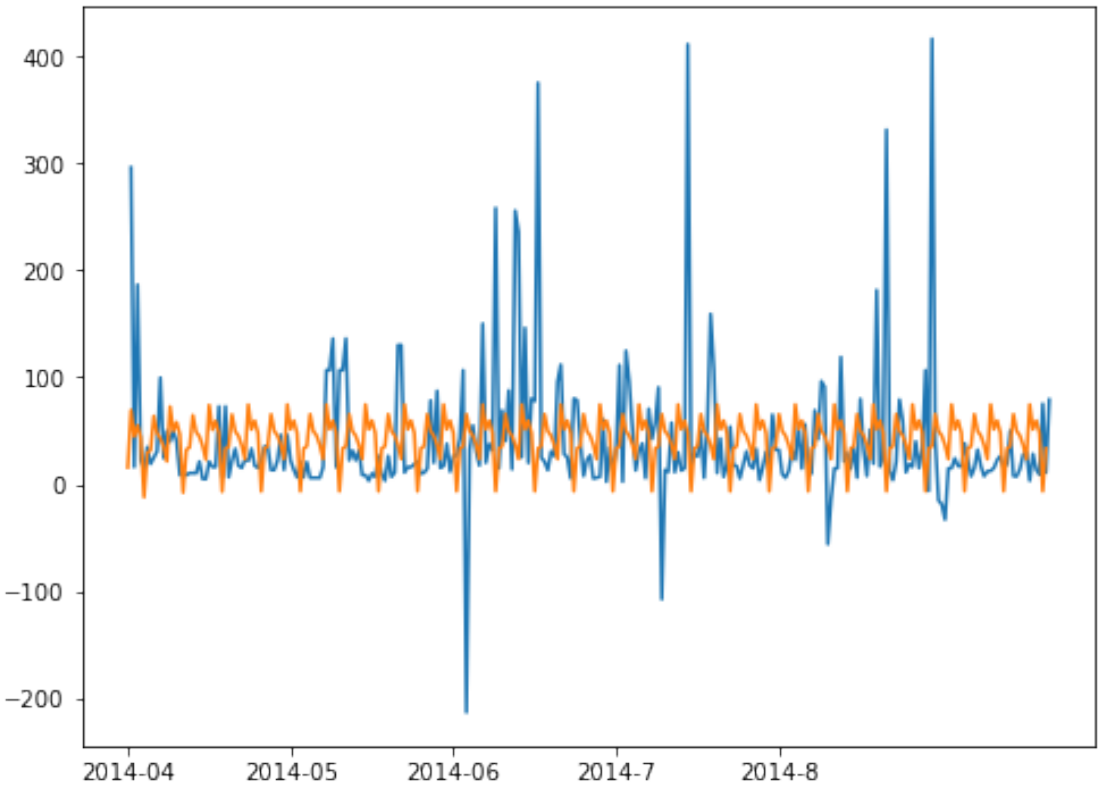
We predict by using the best combination of the ARIMA model ARIMA(1,0,1)(1,1,1)[12].

[192]:
```
arima_prediction = model0.
↪predict(params=train_ts0,dynamic=True,n_periods=283,alpha=0.3)
```

[193]:
```
arima_pred = model0.predict(params=train_ts0,dynamic=True,n_periods=89,alpha=0.
↪3)
```

We plot the time series and the prediction.

[194]:
```
plt.figure(figsize=(8,6))
plt.plot(range(1, 284, 1), ts[0][['ORIGINAL GROSS AMT']],label='ORIGINAL GROSS␣
↪AMT')
plt.plot(arima_prediction,label='ARIMA')
plt.xticks(np.arange(0, 250, step=50),␣
↪['2014-04','2014-05','2014-06','2014-7','2014-8']);
```



We calculate the MAE, MSE and RMSE for our model.

[195]:
```
print('ARIMA MAE = ', mean_absolute_error(arima_prediction,ts[0]['ORIGINAL␣
↪GROSS AMT']))
print('ARIMA MSE = ', mean_squared_error(arima_prediction,ts[0]['ORIGINAL GROSS␣
↪AMT']))
print('ARIMA RMSE = ', np.
↪sqrt(mean_squared_error(arima_prediction,ts[0]['ORIGINAL GROSS AMT'])))
```

```
ARIMA MAE =  42.060421648581546
```

```
ARIMA MSE =  4407.168013856404
ARIMA RMSE =  66.38650475703932
```

1.4.10 Prophet

This is the open source time series library released by Facebook.

It does not require to search for hyperparameters. The model can act as a black box that does all the required computations on its own.

Prophet expects the data frame to have 2 columns, ds and y.

We create the training, the test and the full data set.

[196]:
```
train = pd.DataFrame(train_ts0)
train.columns = ['y']
train['ds'] = train.index

test = pd.DataFrame(test_ts0)
test.columns = ['y']
test['ds'] = test.index

dataset = ts[0].copy()
dataset['y'] = dataset['ORIGINAL GROSS AMT']
dataset['ds'] = dataset.index
dataset.drop(['CARD NUMBER','ORIGINAL GROSS AMT','TRANS␣
 ↪DATE'],axis=1,inplace=True)
dataset1 = dataset.copy()
dataset1.drop(['ds'],axis=1,inplace=True)
```

We train our model with prophet.

[197]:
```
m = Prophet()
m.fit(train)
m
```

```
INFO:fbprophet:Disabling yearly seasonality. Run prophet with
yearly_seasonality=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with
daily_seasonality=True to override this.
```

[197]:
```
<fbprophet.forecaster.Prophet at 0x7f9cea15d0a0>
```

We save our Prophet model.

[198]:
```
ts_model_prophet_filename = '/Users/andresaristi/Documents/
 ↪BirminghamCardTransactions/pickle_files/ts_model_prophet.pkl'
ts_model_prophet_pkl = open(ts_model_prophet_filename, 'wb')
pickle.dump(m, ts_model_prophet_pkl)
```

```
ts_model_prophet_pkl.close()
```

We make our forecast and see the summary of it.

[199]:
```
future = m.make_future_dataframe(periods=test.shape[0])
prophet_prediction = m.predict(future)
prophet_prediction.sort_values(['ds'],inplace=True)
prophet_prediction
```

[199]:

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	\
0	2014-04-07	35.376374	-40.623646	116.848019	35.376374	35.376374	
1	2014-04-08	35.404373	-25.706041	129.379173	35.404373	35.404373	
2	2014-04-09	35.432372	-44.317680	113.413814	35.432372	35.432372	
3	2014-04-10	35.460371	-26.409251	135.310946	35.460371	35.460371	
4	2014-04-14	35.572368	-39.063339	122.360272	35.572368	35.572368	
..	…	…	…	…	…	…	
164	2015-01-01	42.911054	-25.234617	140.104235	42.910878	42.911227	
165	2015-01-02	42.939068	-43.246652	116.488770	42.938888	42.939244	
166	2015-01-03	42.967081	-50.924649	102.668563	42.966899	42.967262	
167	2015-01-04	42.995095	-54.414914	104.583711	42.994910	42.995279	
168	2015-01-05	43.023109	-30.668457	131.295695	43.022921	43.023296	

	additive_terms	additive_terms_lower	additive_terms_upper	weekly	\
0	6.137847	6.137847	6.137847	6.137847	
1	20.309930	20.309930	20.309930	20.309930	
2	1.225086	1.225086	1.225086	1.225086	
3	13.859857	13.859857	13.859857	13.859857	
4	6.137847	6.137847	6.137847	6.137847	
..	…	…	…	…	
164	13.859857	13.859857	13.859857	13.859857	
165	-8.214942	-8.214942	-8.214942	-8.214942	
166	-16.705761	-16.705761	-16.705761	-16.705761	
167	-16.612017	-16.612017	-16.612017	-16.612017	
168	6.137847	6.137847	6.137847	6.137847	

	weekly_lower	weekly_upper	multiplicative_terms	\
0	6.137847	6.137847	0.0	
1	20.309930	20.309930	0.0	
2	1.225086	1.225086	0.0	
3	13.859857	13.859857	0.0	
4	6.137847	6.137847	0.0	
..	…	…	…	
164	13.859857	13.859857	0.0	
165	-8.214942	-8.214942	0.0	
166	-16.705761	-16.705761	0.0	
167	-16.612017	-16.612017	0.0	
168	6.137847	6.137847	0.0	

```
     multiplicative_terms_lower  multiplicative_terms_upper       yhat
0                           0.0                         0.0  41.514221
1                           0.0                         0.0  55.714303
2                           0.0                         0.0  36.657458
3                           0.0                         0.0  49.320228
4                           0.0                         0.0  41.710215
..                          …                           …        …
164                         0.0                         0.0  56.770910
165                         0.0                         0.0  34.724126
166                         0.0                         0.0  26.261320
167                         0.0                         0.0  26.383078
168                         0.0                         0.0  49.160957

[169 rows x 16 columns]
```

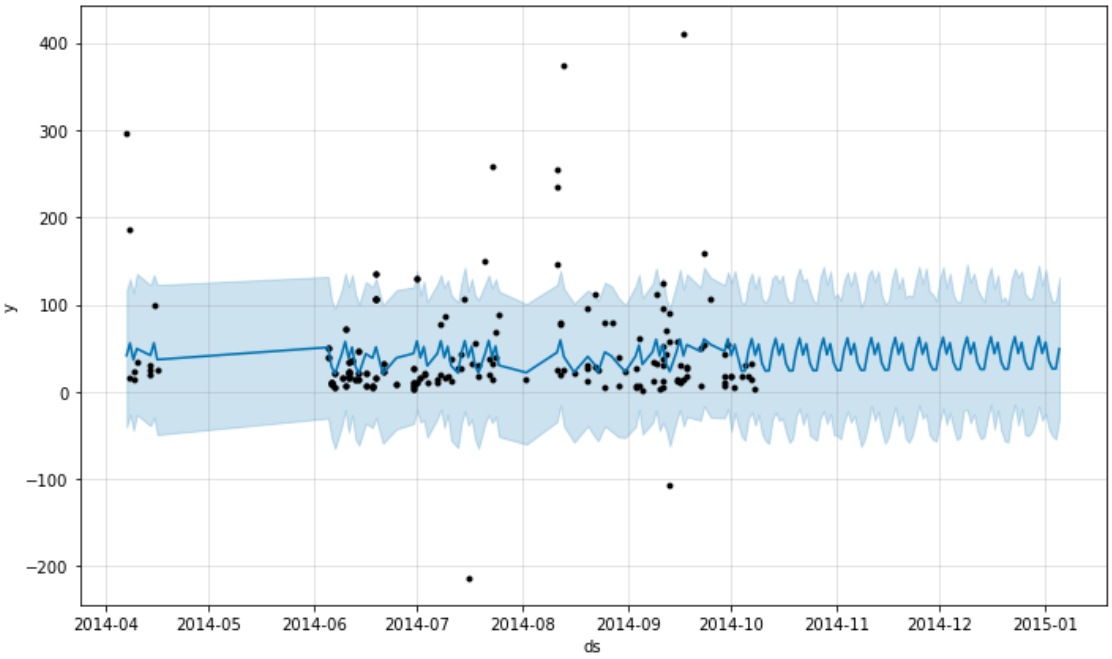
We calculate the MAE of the model and see that is a bit better than ARIMA.

[200]:
```
prophet_future = pd.DataFrame(prophet_prediction.yhat.iloc[0:89])
prophet_future
prophet_future1 = prophet_prediction.yhat
test_results = test.copy()
test_results.drop(['ds'],axis=1,inplace=True)
print('Prophet MAE = ',mean_absolute_error(prophet_future,test_results))
print('Prophet MSE = ', mean_squared_error(prophet_future,test_results))
print('Prophet RMSE = ', np.
 ↪sqrt(mean_squared_error(prophet_future,test_results)))
```

```
Prophet MAE =  38.14150563463835
Prophet MSE =  3961.0575290471243
Prophet RMSE =  62.9369329491605
```

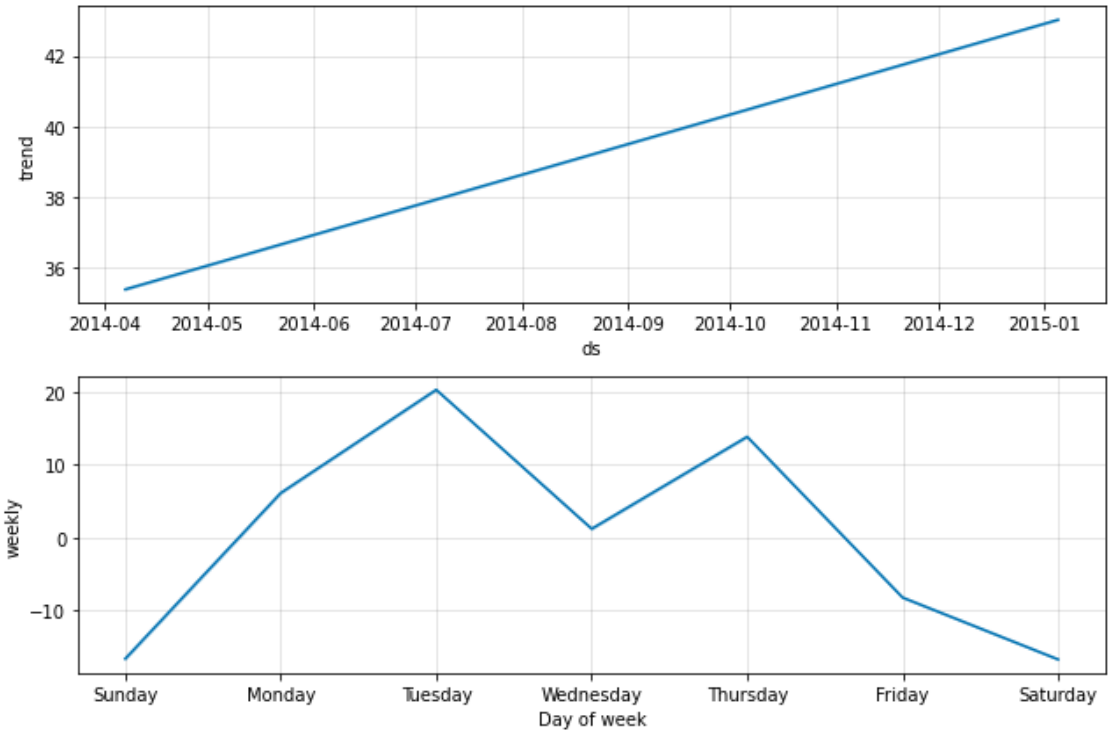
We plot the forecast.

[201]:
```
fig1 = m.plot(prophet_prediction)
```



We observe the forecast components. The trend and the weekly seasonality of the time series.

[202]:
```
fig2 = m.plot_components(prophet_prediction)
```



We plot an interactive figure of the forecast and components created with plotly.

[203]:
```
plot_plotly(m, prophet_prediction)
```

1.4.11 Preparing the dataset for XGBOOST and NN

These two previously stated models are supervised machine learning models that deal with independent data points, or examples. They assume that each data point i independent from the other data points in the dataset.

With the following method we extract the features from a given dataframe.

[204]:
```
dataset_fs_sup = pd.DataFrame()
dataset_fs_sup['day'] = ts[0].index.day
dataset_fs_sup['month'] = ts[0].index.month
dataset_fs_sup['quarter'] = ts[0].index.quarter
dataset_fs_sup['dayofweek'] = ts[0].index.dayofweek
dataset_fs_sup['dayofyear'] = ts[0].index.dayofyear
dataset_fs_sup['weekofyear'] = ts[0].index.weekofyear
dataset_fs_sup['ORIGINAL GROSS AMT'] = ts[0]['ORIGINAL GROSS AMT'].values
dataset_fs_sup['TRANS DATE'] = ts[0]['TRANS DATE'].values
dataset_fs_sup.set_index('TRANS DATE',inplace=True)
dataset_fs_sup
```

[204]:

TRANS DATE	day	month	quarter	dayofweek	dayofyear	weekofyear	ORIGINAL GROSS AMT
2014-04-07	7	4	2	0	97	15	295.80
2014-04-08	8	4	2	1	98	15	16.10
2014-04-08	8	4	2	1	98	15	186.35
2014-04-09	9	4	2	2	99	15	22.75
2014-04-09	9	4	2	2	99	15	14.00
…	…	…	…	…	…	…	…
2014-12-11	11	12	4	3	345	50	
2014-12-14	14	12	4	6	348	50	
2014-12-15	15	12	4	0	349	51	
2014-12-16	16	12	4	1	350	51	
2014-12-28	28	12	4	6	362	52	

```
2014-12-11                13.59
2014-12-14                 9.00
2014-12-15                75.20
2014-12-16                10.95
2014-12-28                79.00

[283 rows x 7 columns]
```

We split the dataset into train and test (we cannot use the train_test_split function because it doesn't work with time series).

[205]:
```
start_date = '2014-10-11'
train = dataset_fs_sup.loc[dataset_fs_sup.index < pd.to_datetime(start_date)]
test = dataset_fs_sup.loc[dataset_fs_sup.index >= pd.to_datetime(start_date)]
```

[206]:
```
X_train = train.iloc[:,:6]
X_test = test.iloc[:,:6]
y_train = train.iloc[:,6]
y_test = test.iloc[:,6]
X_train.shape
```

[206]:
```
(194, 6)
```

We Standardize the data.

[207]:
```
scaler = StandardScaler()
scaler.fit(X_train)
```

[207]:
```
StandardScaler()
```

[208]:
```
scaled_train = scaler.transform(X_train)
scaled_test = scaler.transform(X_test)
```

We define an XGBoost model, a set of parameters that we will pass to the GridSearchCV function in order to find the best model. We tune the hyperparameters to get that model.

[209]:
```
model = XGBRegressor()
param_search = {'max_depth': [3,6,10],
           'learning_rate': [0.01, 0.05, 0.1],
           'n_estimators': [100, 500, 1000],
           'colsample_bytree': [0.3, 0.7]}

tscv = TimeSeriesSplit(n_splits=2)
gsearch = GridSearchCV(estimator=model, cv=tscv,
                       param_grid=param_search)
gsearch.fit(scaled_train, y_train)
gsearch.best_estimator_
```

[209]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.3, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.01, max_delta_step=0, max_depth=3,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=500, n_jobs=8, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
```

We get the best model.

[210]:
```
XGBOOST_model = gsearch.best_estimator_
```

We train the model and make the predictions taking the test set.

[211]:
```
XGBOOST_model.fit(scaled_train, y_train.values,
                  eval_set=[(scaled_train,y_train.values),(scaled_test,y_test.
 ↪values)],
                  verbose=False);
XGBOOST_prediction = XGBOOST_model.predict(scaled_test)
```

We save the XGBoost model in a pkl file.

[212]:
```
model_xg_pkl_filename = '/Users/andresaristi/Documents/
 ↪BirminghamCardTransactions/pickle_files/model_xg.pkl'
model_xg_pkl = open(model_xg_pkl_filename, 'wb')
pickle.dump(XGBOOST_model, model_xg_pkl)
model_xg_pkl.close()
```

We gave the test errors for the model.

[213]:
```
print('XGBOOST MAE = ',mean_absolute_error(XGBOOST_prediction,y_test))
print('XGBOOST MSE = ', mean_squared_error(XGBOOST_prediction,y_test))
print('XGBOOST RMSE = ', np.sqrt(mean_squared_error(XGBOOST_prediction,y_test)))
```

```
XGBOOST MAE =  31.367442750502175
XGBOOST MSE =  3659.5421719196574
XGBOOST RMSE =  60.494149898313786
```

1.4.12 Neural Network

We define a function that creates a neural network that will be used to get the best one according to some parameters. We have 4 layers and the last one has just one unit in order to make a continous prediction, according to what the problem asks.

We use an adam optimizer a mean absolute error loss and as a metric.

[214]:
```
def build_nn(unit):
    nn = Sequential([Dense(unit, activation='relu',name='layer1'),
            Dense(unit, activation='relu',name='layer2'),
            Dense(unit, activation='relu',name='layer3'),
            Dense(1,name='layer4')])
    nn.compile(optimizer = 'adam', loss = 'mean_absolute_error', metrics =␣
 ↪['mae'])
    return nn
```

- We create the KerasRegressor model taking as a parameter our predefined function.
- We define the parameters.
- We use the GridSearchCV to find the best model by means of the parameters.
- We train the model.

[215]:
```
model = KerasRegressor(build_fn=build_nn)

params={'batch_size':[100, 20, 50, 25, 32],
        'nb_epoch':[200, 100, 300, 400],
        'unit':[5,6, 10, 11, 12, 15],
        }

gs=GridSearchCV(estimator=model, param_grid=params, cv=10,verbose=False)

gs = gs.fit(scaled_train, y_train,verbose=False)
```

```
1/1 [==============================] - 0s 128ms/step - loss: 49.2102 - mae:
49.2102
1/1 [==============================] - 0s 81ms/step - loss: 22.7370 - mae:
22.7370
1/1 [==============================] - 0s 79ms/step - loss: 18.9681 - mae:
18.9681
1/1 [==============================] - 0s 80ms/step - loss: 46.0765 - mae:
46.0765
WARNING:tensorflow:5 out of the last 9 calls to <function
Model.make_train_function.<locals>.train_function at 0x7f9c60f6b0d0> triggered
tf.function retracing. Tracing is expensive and the excessive number of tracings
could be due to (1) creating @tf.function repeatedly in a loop, (2) passing
tensors with different shapes, (3) passing Python objects instead of tensors.
For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.

WARNING:tensorflow:5 out of the last 9 calls to <function
Model.make_train_function.<locals>.train_function at 0x7f9c60f6b0d0> triggered
tf.function retracing. Tracing is expensive and the excessive number of tracings
```

```
could be due to (1) creating @tf.function repeatedly in a loop, (2) passing
tensors with different shapes, (3) passing Python objects instead of tensors.
For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.

WARNING:tensorflow:5 out of the last 5 calls to <function
Model.make_test_function.<locals>.test_function at 0x7f9c61617280> triggered
tf.function retracing. Tracing is expensive and the excessive number of tracings
could be due to (1) creating @tf.function repeatedly in a loop, (2) passing
tensors with different shapes, (3) passing Python objects instead of tensors.
For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.

WARNING:tensorflow:5 out of the last 5 calls to <function
Model.make_test_function.<locals>.test_function at 0x7f9c61617280> triggered
tf.function retracing. Tracing is expensive and the excessive number of tracings
could be due to (1) creating @tf.function repeatedly in a loop, (2) passing
tensors with different shapes, (3) passing Python objects instead of tensors.
For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.

1/1 [==============================] - 0s 83ms/step - loss: 34.8598 - mae:
34.8598
WARNING:tensorflow:6 out of the last 11 calls to <function
Model.make_train_function.<locals>.train_function at 0x7f9c616603a0> triggered
tf.function retracing. Tracing is expensive and the excessive number of tracings
could be due to (1) creating @tf.function repeatedly in a loop, (2) passing
tensors with different shapes, (3) passing Python objects instead of tensors.
For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.

WARNING:tensorflow:6 out of the last 11 calls to <function
Model.make_train_function.<locals>.train_function at 0x7f9c616603a0> triggered
tf.function retracing. Tracing is expensive and the excessive number of tracings
could be due to (1) creating @tf.function repeatedly in a loop, (2) passing
tensors with different shapes, (3) passing Python objects instead of tensors.
For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has experimental_relax_shapes=True option that relaxes argument
```

```
shapes that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.

WARNING:tensorflow:6 out of the last 6 calls to <function
Model.make_test_function.<locals>.test_function at 0x7f9c61660dc0> triggered
tf.function retracing. Tracing is expensive and the excessive number of tracings
could be due to (1) creating @tf.function repeatedly in a loop, (2) passing
tensors with different shapes, (3) passing Python objects instead of tensors.
For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.

WARNING:tensorflow:6 out of the last 6 calls to <function
Model.make_test_function.<locals>.test_function at 0x7f9c61660dc0> triggered
tf.function retracing. Tracing is expensive and the excessive number of tracings
could be due to (1) creating @tf.function repeatedly in a loop, (2) passing
tensors with different shapes, (3) passing Python objects instead of tensors.
For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.

1/1 [==============================] - 0s 84ms/step - loss: 67.0725 - mae:
67.0725
1/1 [==============================] - 0s 78ms/step - loss: 88.8956 - mae:
88.8956
1/1 [==============================] - 0s 76ms/step - loss: 37.9529 - mae:
37.9529
1/1 [==============================] - 0s 81ms/step - loss: 56.5824 - mae:
56.5824
1/1 [==============================] - 0s 76ms/step - loss: 34.0248 - mae:
34.0248
1/1 [==============================] - 0s 79ms/step - loss: 48.9164 - mae:
48.9164
1/1 [==============================] - 0s 78ms/step - loss: 22.3501 - mae:
22.3501
1/1 [==============================] - 0s 76ms/step - loss: 18.7119 - mae:
18.7119
1/1 [==============================] - 0s 76ms/step - loss: 46.1558 - mae:
46.1558
1/1 [==============================] - 0s 75ms/step - loss: 35.1038 - mae:
35.1038
1/1 [==============================] - 0s 79ms/step - loss: 67.1757 - mae:
67.1757
1/1 [==============================] - 0s 74ms/step - loss: 88.2466 - mae:
```

```
88.2466
1/1 [==============================] - 0s 75ms/step - loss: 38.5794 - mae:
38.5794
1/1 [==============================] - 0s 76ms/step - loss: 56.1485 - mae:
56.1485
1/1 [==============================] - 0s 75ms/step - loss: 34.9005 - mae:
34.9005
1/1 [==============================] - 0s 74ms/step - loss: 48.4328 - mae:
48.4328
1/1 [==============================] - 0s 80ms/step - loss: 22.6186 - mae:
22.6186
1/1 [==============================] - 0s 77ms/step - loss: 18.9086 - mae:
18.9086
1/1 [==============================] - 0s 90ms/step - loss: 46.5045 - mae:
46.5045
1/1 [==============================] - 0s 76ms/step - loss: 34.7297 - mae:
34.7297
1/1 [==============================] - 0s 84ms/step - loss: 66.8083 - mae:
66.8083
1/1 [==============================] - 0s 82ms/step - loss: 88.2897 - mae:
88.2897
1/1 [==============================] - 0s 78ms/step - loss: 39.4211 - mae:
39.4211
1/1 [==============================] - 0s 80ms/step - loss: 56.6059 - mae:
56.6059
1/1 [==============================] - 0s 79ms/step - loss: 34.7156 - mae:
34.7156
1/1 [==============================] - 0s 77ms/step - loss: 49.6898 - mae:
49.6898
1/1 [==============================] - 0s 76ms/step - loss: 22.4489 - mae:
22.4489
1/1 [==============================] - 0s 79ms/step - loss: 18.6704 - mae:
18.6704
1/1 [==============================] - 0s 76ms/step - loss: 46.6272 - mae:
46.6272
1/1 [==============================] - 0s 79ms/step - loss: 34.7918 - mae:
34.7918
1/1 [==============================] - 0s 82ms/step - loss: 67.1089 - mae:
67.1089
1/1 [==============================] - 0s 84ms/step - loss: 88.5461 - mae:
88.5461
1/1 [==============================] - 0s 82ms/step - loss: 38.8379 - mae:
38.8379
1/1 [==============================] - 0s 81ms/step - loss: 56.4986 - mae:
56.4986
1/1 [==============================] - 0s 77ms/step - loss: 35.3206 - mae:
35.3206
1/1 [==============================] - 0s 83ms/step - loss: 49.3361 - mae:
```

```
49.3361
1/1 [==============================] - 0s 80ms/step - loss: 22.6812 - mae:
22.6812
1/1 [==============================] - 0s 74ms/step - loss: 18.5545 - mae:
18.5545
1/1 [==============================] - 0s 81ms/step - loss: 46.5893 - mae:
46.5893
1/1 [==============================] - 0s 78ms/step - loss: 34.9325 - mae:
34.9325
1/1 [==============================] - 0s 77ms/step - loss: 66.8771 - mae:
66.8771
1/1 [==============================] - 0s 81ms/step - loss: 88.2695 - mae:
88.2695
1/1 [==============================] - 0s 76ms/step - loss: 38.5236 - mae:
38.5236
1/1 [==============================] - 0s 74ms/step - loss: 56.1360 - mae:
56.1360
1/1 [==============================] - 0s 77ms/step - loss: 35.5319 - mae:
35.5319
1/1 [==============================] - 0s 75ms/step - loss: 49.5955 - mae:
49.5955
1/1 [==============================] - 0s 74ms/step - loss: 22.9524 - mae:
22.9524
1/1 [==============================] - 0s 77ms/step - loss: 19.1064 - mae:
19.1064
1/1 [==============================] - 0s 77ms/step - loss: 46.1541 - mae:
46.1541
1/1 [==============================] - 0s 83ms/step - loss: 34.9405 - mae:
34.9405
1/1 [==============================] - 0s 75ms/step - loss: 67.2447 - mae:
67.2447
1/1 [==============================] - 0s 81ms/step - loss: 88.0872 - mae:
88.0872
1/1 [==============================] - 0s 79ms/step - loss: 38.4284 - mae:
38.4284
1/1 [==============================] - 0s 78ms/step - loss: 56.7684 - mae:
56.7684
1/1 [==============================] - 0s 75ms/step - loss: 35.0462 - mae:
35.0462
1/1 [==============================] - 0s 77ms/step - loss: 49.6722 - mae:
49.6722
1/1 [==============================] - 0s 84ms/step - loss: 22.6079 - mae:
22.6079
1/1 [==============================] - 0s 79ms/step - loss: 18.9805 - mae:
18.9805
1/1 [==============================] - 0s 75ms/step - loss: 45.8652 - mae:
45.8652
1/1 [==============================] - 0s 79ms/step - loss: 34.9668 - mae:
```

```
34.9668
1/1 [==============================] - 0s 77ms/step - loss: 67.0815 - mae:
67.0815
1/1 [==============================] - 0s 77ms/step - loss: 87.5665 - mae:
87.5665
1/1 [==============================] - 0s 74ms/step - loss: 38.7738 - mae:
38.7738
1/1 [==============================] - 0s 74ms/step - loss: 56.6007 - mae:
56.6007
1/1 [==============================] - 0s 76ms/step - loss: 34.5337 - mae:
34.5337
1/1 [==============================] - 0s 79ms/step - loss: 49.5560 - mae:
49.5560
1/1 [==============================] - 0s 77ms/step - loss: 22.7517 - mae:
22.7517
1/1 [==============================] - 0s 83ms/step - loss: 18.9667 - mae:
18.9667
1/1 [==============================] - 0s 76ms/step - loss: 45.9260 - mae:
45.9260
1/1 [==============================] - 0s 81ms/step - loss: 35.2026 - mae:
35.2026
1/1 [==============================] - 1s 535ms/step - loss: 66.7264 - mae:
66.7264
1/1 [==============================] - 0s 75ms/step - loss: 88.2013 - mae:
88.2013
1/1 [==============================] - 0s 78ms/step - loss: 38.3650 - mae:
38.3650
1/1 [==============================] - 0s 76ms/step - loss: 56.7252 - mae:
56.7252
1/1 [==============================] - 0s 83ms/step - loss: 35.2433 - mae:
35.2433
1/1 [==============================] - 0s 75ms/step - loss: 49.6740 - mae:
49.6740
1/1 [==============================] - 0s 76ms/step - loss: 22.9035 - mae:
22.9035
1/1 [==============================] - 0s 77ms/step - loss: 18.9056 - mae:
18.9056
1/1 [==============================] - 0s 78ms/step - loss: 46.2504 - mae:
46.2504
1/1 [==============================] - 0s 75ms/step - loss: 34.9524 - mae:
34.9524
1/1 [==============================] - 0s 79ms/step - loss: 67.4923 - mae:
67.4923
1/1 [==============================] - 0s 80ms/step - loss: 88.3410 - mae:
88.3410
1/1 [==============================] - 0s 74ms/step - loss: 39.1320 - mae:
39.1320
1/1 [==============================] - 0s 75ms/step - loss: 56.7290 - mae:
```

```
56.7290
1/1 [==============================] - 0s 75ms/step - loss: 35.0840 - mae:
35.0840
1/1 [==============================] - 0s 81ms/step - loss: 49.4590 - mae:
49.4590
1/1 [==============================] - 0s 77ms/step - loss: 22.2702 - mae:
22.2702
1/1 [==============================] - 0s 78ms/step - loss: 18.1719 - mae:
18.1719
1/1 [==============================] - 0s 75ms/step - loss: 46.0713 - mae:
46.0713
1/1 [==============================] - 0s 76ms/step - loss: 34.6419 - mae:
34.6419
1/1 [==============================] - 0s 75ms/step - loss: 66.9404 - mae:
66.9404
1/1 [==============================] - 0s 80ms/step - loss: 88.2382 - mae:
88.2382
1/1 [==============================] - 0s 78ms/step - loss: 38.1301 - mae:
38.1301
1/1 [==============================] - 0s 75ms/step - loss: 57.0248 - mae:
57.0248
1/1 [==============================] - 0s 75ms/step - loss: 36.1022 - mae:
36.1022
1/1 [==============================] - 0s 80ms/step - loss: 49.9684 - mae:
49.9684
1/1 [==============================] - 0s 73ms/step - loss: 22.8689 - mae:
22.8689
1/1 [==============================] - 0s 75ms/step - loss: 19.0145 - mae:
19.0145
1/1 [==============================] - 0s 75ms/step - loss: 46.3735 - mae:
46.3735
1/1 [==============================] - 0s 79ms/step - loss: 35.0493 - mae:
35.0493
1/1 [==============================] - 0s 79ms/step - loss: 67.1419 - mae:
67.1419
1/1 [==============================] - 0s 74ms/step - loss: 88.3571 - mae:
88.3571
1/1 [==============================] - 0s 77ms/step - loss: 38.4725 - mae:
38.4725
1/1 [==============================] - 0s 84ms/step - loss: 56.5913 - mae:
56.5913
1/1 [==============================] - 0s 78ms/step - loss: 34.8104 - mae:
34.8104
1/1 [==============================] - 0s 97ms/step - loss: 48.9461 - mae:
48.9461
1/1 [==============================] - 0s 78ms/step - loss: 22.7099 - mae:
22.7099
1/1 [==============================] - 0s 79ms/step - loss: 18.7812 - mae:
```

```
18.7812
1/1 [==============================] - 0s 82ms/step - loss: 46.0872 - mae:
46.0872
1/1 [==============================] - 0s 84ms/step - loss: 34.7841 - mae:
34.7841
1/1 [==============================] - 0s 92ms/step - loss: 67.1266 - mae:
67.1266
1/1 [==============================] - 0s 77ms/step - loss: 88.0698 - mae:
88.0698
1/1 [==============================] - 0s 79ms/step - loss: 38.1887 - mae:
38.1887
1/1 [==============================] - 0s 82ms/step - loss: 56.7231 - mae:
56.7231
1/1 [==============================] - 0s 75ms/step - loss: 36.2014 - mae:
36.2014
1/1 [==============================] - 0s 77ms/step - loss: 49.7246 - mae:
49.7246
1/1 [==============================] - 0s 78ms/step - loss: 22.5798 - mae:
22.5798
1/1 [==============================] - 0s 81ms/step - loss: 18.8908 - mae:
18.8908
1/1 [==============================] - 0s 77ms/step - loss: 46.8960 - mae:
46.8960
1/1 [==============================] - 0s 78ms/step - loss: 34.8276 - mae:
34.8276
1/1 [==============================] - 0s 83ms/step - loss: 66.9071 - mae:
66.9071
1/1 [==============================] - 0s 83ms/step - loss: 88.4254 - mae:
88.4254
1/1 [==============================] - 0s 80ms/step - loss: 38.8795 - mae:
38.8795
1/1 [==============================] - 0s 79ms/step - loss: 56.7442 - mae:
56.7442
1/1 [==============================] - 0s 78ms/step - loss: 35.4259 - mae:
35.4259
1/1 [==============================] - 0s 80ms/step - loss: 49.7610 - mae:
49.7610
1/1 [==============================] - 0s 81ms/step - loss: 22.9595 - mae:
22.9595
1/1 [==============================] - 0s 75ms/step - loss: 18.8023 - mae:
18.8023
1/1 [==============================] - 0s 77ms/step - loss: 46.2156 - mae:
46.2156
1/1 [==============================] - 0s 81ms/step - loss: 35.1939 - mae:
35.1939
1/1 [==============================] - 0s 77ms/step - loss: 66.9914 - mae:
66.9914
1/1 [==============================] - 0s 78ms/step - loss: 88.0929 - mae:
```

```
88.0929
1/1 [==============================] - 0s 74ms/step - loss: 38.6019 - mae:
38.6019
1/1 [==============================] - 0s 76ms/step - loss: 56.7337 - mae:
56.7337
1/1 [==============================] - 0s 75ms/step - loss: 35.7860 - mae:
35.7860
1/1 [==============================] - 0s 75ms/step - loss: 49.7471 - mae:
49.7471
1/1 [==============================] - 0s 84ms/step - loss: 22.4432 - mae:
22.4432
1/1 [==============================] - 0s 74ms/step - loss: 18.6531 - mae:
18.6531
1/1 [==============================] - 0s 79ms/step - loss: 46.4228 - mae:
46.4228
1/1 [==============================] - 0s 81ms/step - loss: 35.1977 - mae:
35.1977
1/1 [==============================] - 0s 82ms/step - loss: 67.2232 - mae:
67.2232
1/1 [==============================] - 0s 76ms/step - loss: 88.0588 - mae:
88.0588
1/1 [==============================] - 0s 78ms/step - loss: 38.6793 - mae:
38.6793
1/1 [==============================] - 0s 78ms/step - loss: 56.7035 - mae:
56.7035
1/1 [==============================] - 0s 77ms/step - loss: 35.0638 - mae:
35.0638
1/1 [==============================] - 0s 75ms/step - loss: 49.8195 - mae:
49.8195
1/1 [==============================] - 0s 75ms/step - loss: 22.7581 - mae:
22.7581
1/1 [==============================] - 0s 82ms/step - loss: 18.9409 - mae:
18.9409
1/1 [==============================] - 0s 77ms/step - loss: 45.9583 - mae:
45.9583
1/1 [==============================] - 0s 75ms/step - loss: 34.9510 - mae:
34.9510
1/1 [==============================] - 0s 79ms/step - loss: 67.1228 - mae:
67.1228
1/1 [==============================] - 1s 586ms/step - loss: 88.2583 - mae:
88.2583
1/1 [==============================] - 0s 75ms/step - loss: 39.2374 - mae:
39.2374
1/1 [==============================] - 0s 75ms/step - loss: 56.9294 - mae:
56.9294
1/1 [==============================] - 0s 81ms/step - loss: 34.6363 - mae:
34.6363
1/1 [==============================] - 0s 78ms/step - loss: 49.6156 - mae:
```

```
49.6156
1/1 [==============================] - 0s 78ms/step - loss: 22.4933 - mae:
22.4933
1/1 [==============================] - 0s 80ms/step - loss: 18.9826 - mae:
18.9826
1/1 [==============================] - 0s 74ms/step - loss: 46.2484 - mae:
46.2484
1/1 [==============================] - 0s 76ms/step - loss: 35.0220 - mae:
35.0220
1/1 [==============================] - 0s 76ms/step - loss: 67.2548 - mae:
67.2548
1/1 [==============================] - 0s 75ms/step - loss: 88.4051 - mae:
88.4051
1/1 [==============================] - 0s 78ms/step - loss: 38.8584 - mae:
38.8584
1/1 [==============================] - 0s 74ms/step - loss: 57.3298 - mae:
57.3298
1/1 [==============================] - 0s 77ms/step - loss: 35.3618 - mae:
35.3618
1/1 [==============================] - 0s 79ms/step - loss: 49.9356 - mae:
49.9356
1/1 [==============================] - 0s 76ms/step - loss: 21.6325 - mae:
21.6325
1/1 [==============================] - 0s 75ms/step - loss: 19.0137 - mae:
19.0137
1/1 [==============================] - 0s 86ms/step - loss: 46.2034 - mae:
46.2034
1/1 [==============================] - 0s 81ms/step - loss: 34.9426 - mae:
34.9426
1/1 [==============================] - 0s 77ms/step - loss: 67.1375 - mae:
67.1375
1/1 [==============================] - 0s 78ms/step - loss: 88.2310 - mae:
88.2310
1/1 [==============================] - 0s 83ms/step - loss: 38.4128 - mae:
38.4128
1/1 [==============================] - 0s 84ms/step - loss: 56.7638 - mae:
56.7638
1/1 [==============================] - 0s 77ms/step - loss: 35.0925 - mae:
35.0925
1/1 [==============================] - 0s 74ms/step - loss: 49.7635 - mae:
49.7635
1/1 [==============================] - 0s 77ms/step - loss: 22.0033 - mae:
22.0033
1/1 [==============================] - 0s 75ms/step - loss: 18.8333 - mae:
18.8333
1/1 [==============================] - 0s 78ms/step - loss: 45.8991 - mae:
45.8991
1/1 [==============================] - 0s 76ms/step - loss: 34.8865 - mae:
```

```
34.8865
1/1 [==============================] - 0s 82ms/step - loss: 67.1815 - mae:
67.1815
1/1 [==============================] - 0s 75ms/step - loss: 88.2691 - mae:
88.2691
1/1 [==============================] - 0s 75ms/step - loss: 38.3481 - mae:
38.3481
1/1 [==============================] - 0s 80ms/step - loss: 56.2942 - mae:
56.2942
1/1 [==============================] - 0s 75ms/step - loss: 35.1097 - mae:
35.1097
1/1 [==============================] - 0s 78ms/step - loss: 49.9670 - mae:
49.9670
1/1 [==============================] - 0s 77ms/step - loss: 22.3532 - mae:
22.3532
1/1 [==============================] - 0s 80ms/step - loss: 18.8415 - mae:
18.8415
1/1 [==============================] - 0s 76ms/step - loss: 46.4021 - mae:
46.4021
1/1 [==============================] - 0s 78ms/step - loss: 34.8788 - mae:
34.8788
1/1 [==============================] - 0s 79ms/step - loss: 67.0809 - mae:
67.0809
1/1 [==============================] - 0s 78ms/step - loss: 88.1821 - mae:
88.1821
1/1 [==============================] - 0s 75ms/step - loss: 37.8945 - mae:
37.8945
1/1 [==============================] - 0s 79ms/step - loss: 56.6202 - mae:
56.6202
1/1 [==============================] - 0s 78ms/step - loss: 35.0674 - mae:
35.0674
1/1 [==============================] - 0s 74ms/step - loss: 49.6707 - mae:
49.6707
1/1 [==============================] - 0s 78ms/step - loss: 22.5783 - mae:
22.5783
1/1 [==============================] - 0s 77ms/step - loss: 18.4086 - mae:
18.4086
1/1 [==============================] - 0s 81ms/step - loss: 46.3941 - mae:
46.3941
1/1 [==============================] - 0s 80ms/step - loss: 35.0581 - mae:
35.0581
1/1 [==============================] - 0s 78ms/step - loss: 67.2148 - mae:
67.2148
1/1 [==============================] - 0s 77ms/step - loss: 88.4545 - mae:
88.4545
1/1 [==============================] - 0s 77ms/step - loss: 38.2006 - mae:
38.2006
1/1 [==============================] - 0s 79ms/step - loss: 56.5164 - mae:
```

```
56.5164
1/1 [==============================] - 0s 79ms/step - loss: 34.6515 - mae:
34.6515
1/1 [==============================] - 0s 83ms/step - loss: 50.1296 - mae:
50.1296
1/1 [==============================] - 0s 83ms/step - loss: 22.5107 - mae:
22.5107
1/1 [==============================] - 0s 83ms/step - loss: 19.2317 - mae:
19.2317
1/1 [==============================] - 0s 90ms/step - loss: 46.1996 - mae:
46.1996
1/1 [==============================] - 0s 86ms/step - loss: 35.3574 - mae:
35.3574
1/1 [==============================] - 0s 87ms/step - loss: 66.8056 - mae:
66.8056
1/1 [==============================] - 0s 81ms/step - loss: 88.5055 - mae:
88.5055
1/1 [==============================] - 0s 84ms/step - loss: 38.8421 - mae:
38.8421
1/1 [==============================] - 0s 76ms/step - loss: 56.5227 - mae:
56.5227
1/1 [==============================] - 0s 86ms/step - loss: 35.2378 - mae:
35.2378
1/1 [==============================] - 0s 87ms/step - loss: 50.1755 - mae:
50.1755
1/1 [==============================] - 0s 82ms/step - loss: 22.2773 - mae:
22.2773
1/1 [==============================] - 0s 83ms/step - loss: 18.5485 - mae:
18.5485
1/1 [==============================] - 0s 83ms/step - loss: 46.6340 - mae:
46.6340
1/1 [==============================] - 0s 89ms/step - loss: 34.8621 - mae:
34.8621
1/1 [==============================] - 0s 87ms/step - loss: 67.1622 - mae:
67.1622
1/1 [==============================] - 0s 83ms/step - loss: 88.1351 - mae:
88.1351
1/1 [==============================] - 0s 86ms/step - loss: 39.3137 - mae:
39.3137
1/1 [==============================] - 0s 84ms/step - loss: 56.8367 - mae:
56.8367
1/1 [==============================] - 0s 91ms/step - loss: 34.7512 - mae:
34.7512
1/1 [==============================] - 0s 80ms/step - loss: 50.1359 - mae:
50.1359
1/1 [==============================] - 0s 76ms/step - loss: 22.4368 - mae:
22.4368
1/1 [==============================] - 0s 80ms/step - loss: 18.8166 - mae:
```

```
18.8166
1/1 [==============================] - 0s 81ms/step - loss: 46.2549 - mae:
46.2549
1/1 [==============================] - 0s 78ms/step - loss: 35.0023 - mae:
35.0023
1/1 [==============================] - 0s 84ms/step - loss: 67.0788 - mae:
67.0788
1/1 [==============================] - 0s 80ms/step - loss: 88.4235 - mae:
88.4235
1/1 [==============================] - 1s 564ms/step - loss: 38.6753 - mae:
38.6753
1/1 [==============================] - 0s 83ms/step - loss: 56.7789 - mae:
56.7789
1/1 [==============================] - 0s 83ms/step - loss: 35.0189 - mae:
35.0189
1/1 [==============================] - 0s 80ms/step - loss: 50.5585 - mae:
50.5585
1/1 [==============================] - 0s 80ms/step - loss: 23.0936 - mae:
23.0936
1/1 [==============================] - 0s 76ms/step - loss: 18.8317 - mae:
18.8317
1/1 [==============================] - 0s 74ms/step - loss: 45.1112 - mae:
45.1112
1/1 [==============================] - 0s 80ms/step - loss: 34.4562 - mae:
34.4562
1/1 [==============================] - 0s 79ms/step - loss: 67.1762 - mae:
67.1762
1/1 [==============================] - 0s 76ms/step - loss: 87.6427 - mae:
87.6427
1/1 [==============================] - 0s 78ms/step - loss: 38.8384 - mae:
38.8384
1/1 [==============================] - 0s 85ms/step - loss: 56.7094 - mae:
56.7094
1/1 [==============================] - 0s 81ms/step - loss: 31.9112 - mae:
31.9112
1/1 [==============================] - 0s 74ms/step - loss: 49.7737 - mae:
49.7737
1/1 [==============================] - 0s 74ms/step - loss: 23.3611 - mae:
23.3611
1/1 [==============================] - 0s 77ms/step - loss: 18.7194 - mae:
18.7194
1/1 [==============================] - 0s 76ms/step - loss: 45.8815 - mae:
45.8815
1/1 [==============================] - 0s 82ms/step - loss: 34.8141 - mae:
34.8141
1/1 [==============================] - 0s 77ms/step - loss: 67.1008 - mae:
67.1008
1/1 [==============================] - 0s 76ms/step - loss: 88.3559 - mae:
```

```
88.3559
1/1 [==============================] - 0s 80ms/step - loss: 38.6177 - mae:
38.6177
1/1 [==============================] - 0s 78ms/step - loss: 56.7183 - mae:
56.7183
1/1 [==============================] - 0s 81ms/step - loss: 34.8661 - mae:
34.8661
1/1 [==============================] - 0s 74ms/step - loss: 49.3025 - mae:
49.3025
1/1 [==============================] - 0s 77ms/step - loss: 22.5870 - mae:
22.5870
1/1 [==============================] - 0s 81ms/step - loss: 18.8913 - mae:
18.8913
1/1 [==============================] - 0s 73ms/step - loss: 46.0252 - mae:
46.0252
1/1 [==============================] - 0s 78ms/step - loss: 35.0732 - mae:
35.0732
1/1 [==============================] - 0s 77ms/step - loss: 66.9633 - mae:
66.9633
1/1 [==============================] - 0s 77ms/step - loss: 88.4161 - mae:
88.4161
1/1 [==============================] - 0s 82ms/step - loss: 38.4752 - mae:
38.4752
1/1 [==============================] - 0s 76ms/step - loss: 56.8028 - mae:
56.8028
1/1 [==============================] - 0s 80ms/step - loss: 33.8894 - mae:
33.8894
1/1 [==============================] - 0s 77ms/step - loss: 49.7447 - mae:
49.7447
1/1 [==============================] - 0s 78ms/step - loss: 22.1748 - mae:
22.1748
1/1 [==============================] - 0s 74ms/step - loss: 18.0711 - mae:
18.0711
1/1 [==============================] - 0s 79ms/step - loss: 46.5607 - mae:
46.5607
1/1 [==============================] - 0s 77ms/step - loss: 34.8678 - mae:
34.8678
1/1 [==============================] - 0s 80ms/step - loss: 66.7411 - mae:
66.7411
1/1 [==============================] - 0s 84ms/step - loss: 88.3428 - mae:
88.3428
1/1 [==============================] - 0s 80ms/step - loss: 38.4932 - mae:
38.4932
1/1 [==============================] - 0s 82ms/step - loss: 56.5803 - mae:
56.5803
1/1 [==============================] - 0s 76ms/step - loss: 34.9988 - mae:
34.9988
1/1 [==============================] - 0s 79ms/step - loss: 49.1961 - mae:
```

```
49.1961
1/1 [==============================] - 0s 77ms/step - loss: 22.4072 - mae:
22.4072
1/1 [==============================] - 0s 76ms/step - loss: 18.1295 - mae:
18.1295
1/1 [==============================] - 0s 73ms/step - loss: 46.2599 - mae:
46.2599
1/1 [==============================] - 0s 77ms/step - loss: 34.9964 - mae:
34.9964
1/1 [==============================] - 0s 78ms/step - loss: 66.9843 - mae:
66.9843
1/1 [==============================] - 0s 75ms/step - loss: 88.4646 - mae:
88.4646
1/1 [==============================] - 0s 81ms/step - loss: 38.1395 - mae:
38.1395
1/1 [==============================] - 0s 74ms/step - loss: 56.3472 - mae:
56.3472
1/1 [==============================] - 0s 75ms/step - loss: 34.4200 - mae:
34.4200
1/1 [==============================] - 0s 72ms/step - loss: 49.3008 - mae:
49.3008
1/1 [==============================] - 0s 77ms/step - loss: 22.2929 - mae:
22.2929
1/1 [==============================] - 0s 82ms/step - loss: 18.6884 - mae:
18.6884
1/1 [==============================] - 0s 74ms/step - loss: 45.9389 - mae:
45.9389
1/1 [==============================] - 0s 74ms/step - loss: 34.5054 - mae:
34.5054
1/1 [==============================] - 0s 79ms/step - loss: 66.9781 - mae:
66.9781
1/1 [==============================] - 0s 75ms/step - loss: 88.3400 - mae:
88.3400
1/1 [==============================] - 0s 75ms/step - loss: 38.2866 - mae:
38.2866
1/1 [==============================] - 0s 77ms/step - loss: 57.2366 - mae:
57.2366
1/1 [==============================] - 0s 76ms/step - loss: 34.7078 - mae:
34.7078
1/1 [==============================] - 0s 72ms/step - loss: 50.0424 - mae:
50.0424
1/1 [==============================] - 0s 75ms/step - loss: 22.6342 - mae:
22.6342
1/1 [==============================] - 0s 81ms/step - loss: 18.8404 - mae:
18.8404
1/1 [==============================] - 0s 73ms/step - loss: 46.2975 - mae:
46.2975
1/1 [==============================] - 0s 75ms/step - loss: 34.8887 - mae:
```

```
34.8887
1/1 [==============================] - 0s 82ms/step - loss: 67.8303 - mae:
67.8303
1/1 [==============================] - 0s 74ms/step - loss: 88.2703 - mae:
88.2703
1/1 [==============================] - 0s 75ms/step - loss: 38.9609 - mae:
38.9609
1/1 [==============================] - 0s 78ms/step - loss: 57.0717 - mae:
57.0717
1/1 [==============================] - 0s 75ms/step - loss: 35.0249 - mae:
35.0249
1/1 [==============================] - 0s 72ms/step - loss: 49.7967 - mae:
49.7967
1/1 [==============================] - 0s 71ms/step - loss: 22.5433 - mae:
22.5433
1/1 [==============================] - 0s 73ms/step - loss: 18.8690 - mae:
18.8690
1/1 [==============================] - 0s 72ms/step - loss: 47.3224 - mae:
47.3224
1/1 [==============================] - 0s 77ms/step - loss: 34.7828 - mae:
34.7828
1/1 [==============================] - 0s 76ms/step - loss: 66.8977 - mae:
66.8977
1/1 [==============================] - 0s 82ms/step - loss: 88.2631 - mae:
88.2631
1/1 [==============================] - 0s 75ms/step - loss: 38.3882 - mae:
38.3882
1/1 [==============================] - 0s 76ms/step - loss: 56.4574 - mae:
56.4574
1/1 [==============================] - 0s 78ms/step - loss: 32.5058 - mae:
32.5058
1/1 [==============================] - 0s 72ms/step - loss: 49.6075 - mae:
49.6075
1/1 [==============================] - 0s 73ms/step - loss: 21.9905 - mae:
21.9905
1/1 [==============================] - 0s 72ms/step - loss: 18.5116 - mae:
18.5116
1/1 [==============================] - 0s 74ms/step - loss: 46.4914 - mae:
46.4914
1/1 [==============================] - 0s 75ms/step - loss: 34.8102 - mae:
34.8102
1/1 [==============================] - 0s 77ms/step - loss: 66.8811 - mae:
66.8811
1/1 [==============================] - 0s 76ms/step - loss: 88.3893 - mae:
88.3893
1/1 [==============================] - 0s 75ms/step - loss: 38.4831 - mae:
38.4831
1/1 [==============================] - 0s 76ms/step - loss: 55.6283 - mae:
```

```
55.6283
1/1 [==============================] - 0s 80ms/step - loss: 35.1456 - mae:
35.1456
1/1 [==============================] - 0s 72ms/step - loss: 49.2574 - mae:
49.2574
1/1 [==============================] - 0s 73ms/step - loss: 22.4265 - mae:
22.4265
1/1 [==============================] - 0s 79ms/step - loss: 18.3841 - mae:
18.3841
1/1 [==============================] - 0s 72ms/step - loss: 45.9262 - mae:
45.9262
1/1 [==============================] - 0s 77ms/step - loss: 35.1228 - mae:
35.1228
1/1 [==============================] - 0s 74ms/step - loss: 67.2340 - mae:
67.2340
1/1 [==============================] - 0s 75ms/step - loss: 87.9912 - mae:
87.9912
1/1 [==============================] - 0s 78ms/step - loss: 38.5202 - mae:
38.5202
1/1 [==============================] - 0s 82ms/step - loss: 56.4783 - mae:
56.4783
1/1 [==============================] - 0s 75ms/step - loss: 34.6392 - mae:
34.6392
1/1 [==============================] - 0s 74ms/step - loss: 49.8666 - mae:
49.8666
1/1 [==============================] - 0s 76ms/step - loss: 22.6679 - mae:
22.6679
1/1 [==============================] - 0s 72ms/step - loss: 18.6851 - mae:
18.6851
1/1 [==============================] - 0s 75ms/step - loss: 46.6808 - mae:
46.6808
1/1 [==============================] - 0s 74ms/step - loss: 34.8127 - mae:
34.8127
1/1 [==============================] - 0s 86ms/step - loss: 66.9301 - mae:
66.9301
1/1 [==============================] - 0s 80ms/step - loss: 88.1509 - mae:
88.1509
1/1 [==============================] - 0s 74ms/step - loss: 38.5269 - mae:
38.5269
1/1 [==============================] - 0s 80ms/step - loss: 56.6078 - mae:
56.6078
1/1 [==============================] - 0s 76ms/step - loss: 34.9223 - mae:
34.9223
1/1 [==============================] - 0s 72ms/step - loss: 49.1562 - mae:
49.1562
1/1 [==============================] - 0s 74ms/step - loss: 22.6594 - mae:
22.6594
1/1 [==============================] - 0s 72ms/step - loss: 18.5906 - mae:
```

```
18.5906
1/1 [==============================] - 0s 75ms/step - loss: 46.3595 - mae:
46.3595
1/1 [==============================] - 0s 79ms/step - loss: 34.4455 - mae:
34.4455
1/1 [==============================] - 0s 82ms/step - loss: 67.0145 - mae:
67.0145
1/1 [==============================] - 0s 78ms/step - loss: 87.8079 - mae:
87.8079
1/1 [==============================] - 0s 75ms/step - loss: 38.4312 - mae:
38.4312
1/1 [==============================] - 0s 79ms/step - loss: 56.5768 - mae:
56.5768
1/1 [==============================] - 0s 74ms/step - loss: 35.7453 - mae:
35.7453
1/1 [==============================] - 0s 74ms/step - loss: 49.2176 - mae:
49.2176
1/1 [==============================] - 0s 72ms/step - loss: 22.4958 - mae:
22.4958
1/1 [==============================] - 0s 75ms/step - loss: 18.7806 - mae:
18.7806
1/1 [==============================] - 0s 74ms/step - loss: 46.1148 - mae:
46.1148
1/1 [==============================] - 0s 75ms/step - loss: 34.7387 - mae:
34.7387
1/1 [==============================] - 0s 76ms/step - loss: 67.4821 - mae:
67.4821
1/1 [==============================] - 0s 82ms/step - loss: 88.0348 - mae:
88.0348
1/1 [==============================] - 1s 542ms/step - loss: 38.5446 - mae:
38.5446
1/1 [==============================] - 0s 73ms/step - loss: 56.2617 - mae:
56.2617
1/1 [==============================] - 0s 74ms/step - loss: 35.0925 - mae:
35.0925
1/1 [==============================] - 0s 73ms/step - loss: 49.6771 - mae:
49.6771
1/1 [==============================] - 0s 75ms/step - loss: 23.9625 - mae:
23.9625
1/1 [==============================] - 0s 89ms/step - loss: 18.7629 - mae:
18.7629
1/1 [==============================] - 0s 81ms/step - loss: 46.2349 - mae:
46.2349
1/1 [==============================] - 0s 81ms/step - loss: 34.3790 - mae:
34.3790
1/1 [==============================] - 0s 79ms/step - loss: 66.9957 - mae:
66.9957
1/1 [==============================] - 0s 79ms/step - loss: 88.0031 - mae:
```

```
88.0031
1/1 [==============================] - 0s 78ms/step - loss: 38.6330 - mae:
38.6330
1/1 [==============================] - 0s 76ms/step - loss: 56.7962 - mae:
56.7962
1/1 [==============================] - 0s 77ms/step - loss: 34.6917 - mae:
34.6917
1/1 [==============================] - 0s 73ms/step - loss: 49.9303 - mae:
49.9303
1/1 [==============================] - 0s 78ms/step - loss: 22.9028 - mae:
22.9028
1/1 [==============================] - 0s 82ms/step - loss: 19.0692 - mae:
19.0692
1/1 [==============================] - 0s 74ms/step - loss: 46.1843 - mae:
46.1843
1/1 [==============================] - 0s 78ms/step - loss: 34.8455 - mae:
34.8455
1/1 [==============================] - 0s 78ms/step - loss: 66.9047 - mae:
66.9047
1/1 [==============================] - 0s 76ms/step - loss: 88.6722 - mae:
88.6722
1/1 [==============================] - 0s 78ms/step - loss: 38.1463 - mae:
38.1463
1/1 [==============================] - 0s 77ms/step - loss: 56.9074 - mae:
56.9074
1/1 [==============================] - 0s 78ms/step - loss: 34.9299 - mae:
34.9299
1/1 [==============================] - 0s 73ms/step - loss: 48.9779 - mae:
48.9779
1/1 [==============================] - 0s 78ms/step - loss: 22.5403 - mae:
22.5403
1/1 [==============================] - 0s 75ms/step - loss: 18.9826 - mae:
18.9826
1/1 [==============================] - 0s 75ms/step - loss: 45.8177 - mae:
45.8177
1/1 [==============================] - 0s 74ms/step - loss: 34.9400 - mae:
34.9400
1/1 [==============================] - 0s 75ms/step - loss: 67.2641 - mae:
67.2641
1/1 [==============================] - 0s 77ms/step - loss: 88.2241 - mae:
88.2241
1/1 [==============================] - 0s 75ms/step - loss: 38.3720 - mae:
38.3720
1/1 [==============================] - 0s 79ms/step - loss: 56.5041 - mae:
56.5041
1/1 [==============================] - 0s 82ms/step - loss: 34.4077 - mae:
34.4077
1/1 [==============================] - 0s 73ms/step - loss: 49.4390 - mae:
```

```
49.4390
1/1 [==============================] - 0s 72ms/step - loss: 22.4636 - mae:
22.4636
1/1 [==============================] - 0s 73ms/step - loss: 19.1520 - mae:
19.1520
1/1 [==============================] - 0s 78ms/step - loss: 46.9295 - mae:
46.9295
1/1 [==============================] - 0s 74ms/step - loss: 34.6490 - mae:
34.6490
1/1 [==============================] - 0s 73ms/step - loss: 66.7884 - mae:
66.7884
1/1 [==============================] - 0s 83ms/step - loss: 88.3192 - mae:
88.3192
1/1 [==============================] - 0s 75ms/step - loss: 38.3814 - mae:
38.3814
1/1 [==============================] - 0s 77ms/step - loss: 56.5682 - mae:
56.5682
1/1 [==============================] - 0s 81ms/step - loss: 35.4150 - mae:
35.4150
1/1 [==============================] - 0s 77ms/step - loss: 50.0866 - mae:
50.0866
1/1 [==============================] - 0s 75ms/step - loss: 22.6080 - mae:
22.6080
1/1 [==============================] - 0s 76ms/step - loss: 18.1072 - mae:
18.1072
1/1 [==============================] - 0s 77ms/step - loss: 46.2949 - mae:
46.2949
1/1 [==============================] - 0s 78ms/step - loss: 35.0302 - mae:
35.0302
1/1 [==============================] - 0s 76ms/step - loss: 66.9395 - mae:
66.9395
1/1 [==============================] - 0s 80ms/step - loss: 88.2959 - mae:
88.2959
1/1 [==============================] - 0s 74ms/step - loss: 38.4484 - mae:
38.4484
1/1 [==============================] - 0s 82ms/step - loss: 56.2314 - mae:
56.2314
1/1 [==============================] - 0s 83ms/step - loss: 35.3687 - mae:
35.3687
1/1 [==============================] - 0s 78ms/step - loss: 49.8673 - mae:
49.8673
1/1 [==============================] - 0s 72ms/step - loss: 22.5949 - mae:
22.5949
1/1 [==============================] - 0s 79ms/step - loss: 18.7864 - mae:
18.7864
1/1 [==============================] - 0s 74ms/step - loss: 45.9055 - mae:
45.9055
1/1 [==============================] - 0s 82ms/step - loss: 34.6815 - mae:
```

```
34.6815
1/1 [==============================] - 0s 75ms/step - loss: 67.1330 - mae:
67.1330
1/1 [==============================] - 0s 75ms/step - loss: 88.3390 - mae:
88.3390
1/1 [==============================] - 0s 80ms/step - loss: 38.1283 - mae:
38.1283
1/1 [==============================] - 0s 75ms/step - loss: 56.6828 - mae:
56.6828
1/1 [==============================] - 0s 75ms/step - loss: 35.1637 - mae:
35.1637
1/1 [==============================] - 0s 72ms/step - loss: 50.1635 - mae:
50.1635
1/1 [==============================] - 0s 74ms/step - loss: 21.5180 - mae:
21.5180
1/1 [==============================] - 0s 74ms/step - loss: 18.4556 - mae:
18.4556
1/1 [==============================] - 0s 73ms/step - loss: 46.4014 - mae:
46.4014
1/1 [==============================] - 0s 81ms/step - loss: 34.9627 - mae:
34.9627
1/1 [==============================] - 0s 75ms/step - loss: 67.2253 - mae:
67.2253
1/1 [==============================] - 0s 74ms/step - loss: 87.4147 - mae:
87.4147
1/1 [==============================] - 0s 81ms/step - loss: 38.8727 - mae:
38.8727
1/1 [==============================] - 0s 77ms/step - loss: 55.2140 - mae:
55.2140
1/1 [==============================] - 0s 77ms/step - loss: 35.4045 - mae:
35.4045
1/1 [==============================] - 0s 75ms/step - loss: 49.1001 - mae:
49.1001
1/1 [==============================] - 0s 71ms/step - loss: 22.3915 - mae:
22.3915
1/1 [==============================] - 0s 79ms/step - loss: 18.7817 - mae:
18.7817
1/1 [==============================] - 0s 72ms/step - loss: 45.8989 - mae:
45.8989
1/1 [==============================] - 0s 80ms/step - loss: 34.3391 - mae:
34.3391
1/1 [==============================] - 0s 81ms/step - loss: 66.6763 - mae:
66.6763
1/1 [==============================] - 0s 74ms/step - loss: 88.0816 - mae:
88.0816
1/1 [==============================] - 0s 78ms/step - loss: 38.7392 - mae:
38.7392
1/1 [==============================] - 0s 76ms/step - loss: 56.8039 - mae:
```

```
56.8039
1/1 [==============================] - 0s 75ms/step - loss: 35.2642 - mae:
35.2642
1/1 [==============================] - 0s 76ms/step - loss: 49.6130 - mae:
49.6130
1/1 [==============================] - 0s 74ms/step - loss: 22.1723 - mae:
22.1723
1/1 [==============================] - 0s 80ms/step - loss: 18.6218 - mae:
18.6218
1/1 [==============================] - 0s 73ms/step - loss: 46.0119 - mae:
46.0119
1/1 [==============================] - 0s 77ms/step - loss: 34.6206 - mae:
34.6206
1/1 [==============================] - 0s 79ms/step - loss: 67.1905 - mae:
67.1905
1/1 [==============================] - 0s 81ms/step - loss: 88.3430 - mae:
88.3430
1/1 [==============================] - 0s 78ms/step - loss: 37.7550 - mae:
37.7550
1/1 [==============================] - 0s 76ms/step - loss: 56.3708 - mae:
56.3708
1/1 [==============================] - 0s 77ms/step - loss: 35.3982 - mae:
35.3982
1/1 [==============================] - 0s 75ms/step - loss: 49.3568 - mae:
49.3568
1/1 [==============================] - 0s 71ms/step - loss: 22.7412 - mae:
22.7412
1/1 [==============================] - 0s 78ms/step - loss: 18.7228 - mae:
18.7228
1/1 [==============================] - 0s 78ms/step - loss: 46.2319 - mae:
46.2319
1/1 [==============================] - 0s 75ms/step - loss: 34.7925 - mae:
34.7925
1/1 [==============================] - 0s 74ms/step - loss: 66.6151 - mae:
66.6151
1/1 [==============================] - 0s 74ms/step - loss: 88.2520 - mae:
88.2520
1/1 [==============================] - 0s 75ms/step - loss: 38.9652 - mae:
38.9652
1/1 [==============================] - 0s 75ms/step - loss: 56.6407 - mae:
56.6407
1/1 [==============================] - 0s 79ms/step - loss: 35.0785 - mae:
35.0785
1/1 [==============================] - 0s 78ms/step - loss: 49.3746 - mae:
49.3746
1/1 [==============================] - 0s 73ms/step - loss: 22.9150 - mae:
22.9150
1/1 [==============================] - 0s 74ms/step - loss: 18.7501 - mae:
```

```
18.7501
1/1 [==============================] - 0s 77ms/step - loss: 45.8710 - mae:
45.8710
1/1 [==============================] - 0s 76ms/step - loss: 34.6176 - mae:
34.6176
1/1 [==============================] - 0s 81ms/step - loss: 67.0421 - mae:
67.0421
1/1 [==============================] - 0s 76ms/step - loss: 88.1966 - mae:
88.1966
1/1 [==============================] - 0s 75ms/step - loss: 38.4217 - mae:
38.4217
1/1 [==============================] - 0s 77ms/step - loss: 55.9684 - mae:
55.9684
1/1 [==============================] - 0s 75ms/step - loss: 34.9161 - mae:
34.9161
1/1 [==============================] - 0s 76ms/step - loss: 51.0259 - mae:
51.0259
1/1 [==============================] - 0s 79ms/step - loss: 22.8047 - mae:
22.8047
1/1 [==============================] - 0s 79ms/step - loss: 19.2186 - mae:
19.2186
1/1 [==============================] - 0s 75ms/step - loss: 47.2147 - mae:
47.2147
1/1 [==============================] - 0s 83ms/step - loss: 34.8562 - mae:
34.8562
1/1 [==============================] - 0s 86ms/step - loss: 67.2601 - mae:
67.2601
1/1 [==============================] - 0s 77ms/step - loss: 88.2117 - mae:
88.2117
1/1 [==============================] - 0s 81ms/step - loss: 38.4442 - mae:
38.4442
1/1 [==============================] - 0s 76ms/step - loss: 56.6367 - mae:
56.6367
1/1 [==============================] - 0s 117ms/step - loss: 34.3890 - mae:
34.3890
1/1 [==============================] - 0s 81ms/step - loss: 49.2473 - mae:
49.2473
1/1 [==============================] - 0s 76ms/step - loss: 22.6279 - mae:
22.6279
1/1 [==============================] - 0s 76ms/step - loss: 18.6327 - mae:
18.6327
1/1 [==============================] - 0s 81ms/step - loss: 47.0208 - mae:
47.0208
1/1 [==============================] - 0s 78ms/step - loss: 34.8120 - mae:
34.8120
1/1 [==============================] - 0s 77ms/step - loss: 67.0490 - mae:
67.0490
1/1 [==============================] - 0s 82ms/step - loss: 88.4825 - mae:
```

```
88.4825
1/1 [==============================] - 1s 544ms/step - loss: 38.7768 - mae:
38.7768
1/1 [==============================] - 0s 74ms/step - loss: 57.0472 - mae:
57.0472
1/1 [==============================] - 0s 78ms/step - loss: 34.8388 - mae:
34.8388
1/1 [==============================] - 0s 80ms/step - loss: 49.7145 - mae:
49.7145
1/1 [==============================] - 0s 76ms/step - loss: 22.2266 - mae:
22.2266
1/1 [==============================] - 0s 77ms/step - loss: 18.6472 - mae:
18.6472
1/1 [==============================] - 0s 80ms/step - loss: 45.8306 - mae:
45.8306
1/1 [==============================] - 0s 76ms/step - loss: 35.2859 - mae:
35.2859
1/1 [==============================] - 0s 76ms/step - loss: 67.0732 - mae:
67.0732
1/1 [==============================] - 0s 75ms/step - loss: 88.0778 - mae:
88.0778
1/1 [==============================] - 0s 77ms/step - loss: 37.6342 - mae:
37.6342
1/1 [==============================] - 0s 75ms/step - loss: 56.5216 - mae:
56.5216
1/1 [==============================] - 0s 91ms/step - loss: 34.5728 - mae:
34.5728
1/1 [==============================] - 0s 78ms/step - loss: 50.2905 - mae:
50.2905
1/1 [==============================] - 0s 84ms/step - loss: 22.7793 - mae:
22.7793
1/1 [==============================] - 0s 75ms/step - loss: 19.1521 - mae:
19.1521
1/1 [==============================] - 0s 76ms/step - loss: 45.8618 - mae:
45.8618
1/1 [==============================] - 0s 77ms/step - loss: 35.3480 - mae:
35.3480
1/1 [==============================] - 0s 78ms/step - loss: 66.7996 - mae:
66.7996
1/1 [==============================] - 0s 78ms/step - loss: 88.0926 - mae:
88.0926
1/1 [==============================] - 0s 76ms/step - loss: 38.6098 - mae:
38.6098
1/1 [==============================] - 0s 80ms/step - loss: 57.0333 - mae:
57.0333
1/1 [==============================] - 0s 74ms/step - loss: 34.9620 - mae:
34.9620
1/1 [==============================] - 0s 76ms/step - loss: 50.4861 - mae:
```

```
50.4861
1/1 [==============================] - 0s 79ms/step - loss: 22.6505 - mae:
22.6505
1/1 [==============================] - 0s 75ms/step - loss: 18.9264 - mae:
18.9264
1/1 [==============================] - 0s 75ms/step - loss: 46.2747 - mae:
46.2747
1/1 [==============================] - 0s 75ms/step - loss: 34.8775 - mae:
34.8775
1/1 [==============================] - 0s 76ms/step - loss: 67.1439 - mae:
67.1439
1/1 [==============================] - 0s 82ms/step - loss: 88.2273 - mae:
88.2273
1/1 [==============================] - 0s 76ms/step - loss: 38.9784 - mae:
38.9784
1/1 [==============================] - 0s 76ms/step - loss: 56.6890 - mae:
56.6890
1/1 [==============================] - 0s 79ms/step - loss: 35.4825 - mae:
35.4825
1/1 [==============================] - 0s 78ms/step - loss: 49.3630 - mae:
49.3630
1/1 [==============================] - 0s 78ms/step - loss: 22.5122 - mae:
22.5122
1/1 [==============================] - 0s 76ms/step - loss: 19.2362 - mae:
19.2362
1/1 [==============================] - 0s 76ms/step - loss: 46.2330 - mae:
46.2330
1/1 [==============================] - 0s 74ms/step - loss: 34.9956 - mae:
34.9956
1/1 [==============================] - 0s 76ms/step - loss: 67.0135 - mae:
67.0135
1/1 [==============================] - 0s 78ms/step - loss: 87.9834 - mae:
87.9834
1/1 [==============================] - 0s 77ms/step - loss: 39.3531 - mae:
39.3531
1/1 [==============================] - 0s 80ms/step - loss: 56.9305 - mae:
56.9305
1/1 [==============================] - 0s 83ms/step - loss: 33.4114 - mae:
33.4114
1/1 [==============================] - 0s 78ms/step - loss: 49.7287 - mae:
49.7287
1/1 [==============================] - 0s 78ms/step - loss: 22.6223 - mae:
22.6223
1/1 [==============================] - 0s 78ms/step - loss: 18.6154 - mae:
18.6154
1/1 [==============================] - 0s 77ms/step - loss: 46.1132 - mae:
46.1132
1/1 [==============================] - 0s 78ms/step - loss: 34.9841 - mae:
```

```
34.9841
1/1 [==============================] - 0s 78ms/step - loss: 67.2093 - mae:
67.2093
1/1 [==============================] - 0s 76ms/step - loss: 88.2720 - mae:
88.2720
1/1 [==============================] - 0s 77ms/step - loss: 38.0409 - mae:
38.0409
1/1 [==============================] - 0s 78ms/step - loss: 56.2661 - mae:
56.2661
1/1 [==============================] - 0s 82ms/step - loss: 35.1040 - mae:
35.1040
1/1 [==============================] - 0s 78ms/step - loss: 50.0137 - mae:
50.0137
1/1 [==============================] - 0s 74ms/step - loss: 22.1146 - mae:
22.1146
1/1 [==============================] - 0s 78ms/step - loss: 18.9011 - mae:
18.9011
1/1 [==============================] - 0s 74ms/step - loss: 46.1491 - mae:
46.1491
1/1 [==============================] - 0s 74ms/step - loss: 35.0102 - mae:
35.0102
1/1 [==============================] - 0s 81ms/step - loss: 67.0700 - mae:
67.0700
1/1 [==============================] - 0s 77ms/step - loss: 88.1260 - mae:
88.1260
1/1 [==============================] - 0s 74ms/step - loss: 38.5265 - mae:
38.5265
1/1 [==============================] - 0s 83ms/step - loss: 56.6466 - mae:
56.6466
1/1 [==============================] - 0s 75ms/step - loss: 35.1240 - mae:
35.1240
1/1 [==============================] - 0s 76ms/step - loss: 49.1173 - mae:
49.1173
1/1 [==============================] - 0s 76ms/step - loss: 21.8006 - mae:
21.8006
1/1 [==============================] - 0s 78ms/step - loss: 18.7724 - mae:
18.7724
1/1 [==============================] - 0s 81ms/step - loss: 46.2700 - mae:
46.2700
1/1 [==============================] - 0s 78ms/step - loss: 34.5365 - mae:
34.5365
1/1 [==============================] - 0s 81ms/step - loss: 67.2036 - mae:
67.2036
1/1 [==============================] - 0s 77ms/step - loss: 88.3481 - mae:
88.3481
1/1 [==============================] - 0s 76ms/step - loss: 38.8048 - mae:
38.8048
1/1 [==============================] - 0s 80ms/step - loss: 56.4054 - mae:
```

```
56.4054
1/1 [==============================] - 0s 75ms/step - loss: 35.0697 - mae:
35.0697
1/1 [==============================] - 0s 75ms/step - loss: 49.3215 - mae:
49.3215
1/1 [==============================] - 0s 81ms/step - loss: 22.5474 - mae:
22.5474
1/1 [==============================] - 0s 76ms/step - loss: 19.5052 - mae:
19.5052
1/1 [==============================] - 0s 76ms/step - loss: 46.2476 - mae:
46.2476
1/1 [==============================] - 0s 75ms/step - loss: 34.7533 - mae:
34.7533
1/1 [==============================] - 0s 76ms/step - loss: 67.2061 - mae:
67.2061
1/1 [==============================] - 0s 75ms/step - loss: 88.1250 - mae:
88.1250
1/1 [==============================] - 0s 76ms/step - loss: 38.5210 - mae:
38.5210
1/1 [==============================] - 1s 563ms/step - loss: 56.6345 - mae:
56.6345
1/1 [==============================] - 0s 76ms/step - loss: 34.9379 - mae:
34.9379
1/1 [==============================] - 0s 75ms/step - loss: 49.5178 - mae:
49.5178
1/1 [==============================] - 0s 76ms/step - loss: 22.4941 - mae:
22.4941
1/1 [==============================] - 0s 78ms/step - loss: 18.8175 - mae:
18.8175
1/1 [==============================] - 0s 75ms/step - loss: 46.1204 - mae:
46.1204
1/1 [==============================] - 0s 76ms/step - loss: 35.4122 - mae:
35.4122
1/1 [==============================] - 0s 75ms/step - loss: 66.6250 - mae:
66.6250
1/1 [==============================] - 0s 82ms/step - loss: 88.1798 - mae:
88.1798
1/1 [==============================] - 0s 74ms/step - loss: 38.8066 - mae:
38.8066
1/1 [==============================] - 0s 75ms/step - loss: 56.2295 - mae:
56.2295
1/1 [==============================] - 0s 79ms/step - loss: 34.3253 - mae:
34.3253
1/1 [==============================] - 0s 75ms/step - loss: 49.7487 - mae:
49.7487
1/1 [==============================] - 0s 76ms/step - loss: 22.4491 - mae:
22.4491
1/1 [==============================] - 0s 74ms/step - loss: 18.8259 - mae:
```

```
18.8259
1/1 [==============================] - 0s 74ms/step - loss: 46.3178 - mae:
46.3178
1/1 [==============================] - 0s 74ms/step - loss: 34.8117 - mae:
34.8117
1/1 [==============================] - 0s 77ms/step - loss: 66.4012 - mae:
66.4012
1/1 [==============================] - 0s 81ms/step - loss: 88.1444 - mae:
88.1444
1/1 [==============================] - 0s 75ms/step - loss: 37.7944 - mae:
37.7944
1/1 [==============================] - 0s 75ms/step - loss: 56.7221 - mae:
56.7221
1/1 [==============================] - 0s 82ms/step - loss: 35.7528 - mae:
35.7528
1/1 [==============================] - 0s 76ms/step - loss: 49.7788 - mae:
49.7788
1/1 [==============================] - 0s 79ms/step - loss: 22.6252 - mae:
22.6252
1/1 [==============================] - 0s 77ms/step - loss: 19.5361 - mae:
19.5361
1/1 [==============================] - 0s 75ms/step - loss: 46.2180 - mae:
46.2180
1/1 [==============================] - 0s 77ms/step - loss: 34.7744 - mae:
34.7744
1/1 [==============================] - 0s 75ms/step - loss: 67.0367 - mae:
67.0367
1/1 [==============================] - 0s 76ms/step - loss: 88.0424 - mae:
88.0424
1/1 [==============================] - 0s 80ms/step - loss: 39.0451 - mae:
39.0451
1/1 [==============================] - 0s 73ms/step - loss: 56.6007 - mae:
56.6007
1/1 [==============================] - 0s 78ms/step - loss: 36.6830 - mae:
36.6830
1/1 [==============================] - 0s 80ms/step - loss: 49.9481 - mae:
49.9481
1/1 [==============================] - 0s 80ms/step - loss: 22.7336 - mae:
22.7336
1/1 [==============================] - 0s 76ms/step - loss: 18.5086 - mae:
18.5086
1/1 [==============================] - 0s 78ms/step - loss: 46.1818 - mae:
46.1818
1/1 [==============================] - 0s 83ms/step - loss: 34.7671 - mae:
34.7671
1/1 [==============================] - 0s 78ms/step - loss: 66.8821 - mae:
66.8821
1/1 [==============================] - 0s 77ms/step - loss: 88.3081 - mae:
```

```
88.3081
1/1 [==============================] - 0s 76ms/step - loss: 39.1998 - mae:
39.1998
1/1 [==============================] - 0s 76ms/step - loss: 55.8790 - mae:
55.8790
1/1 [==============================] - 0s 76ms/step - loss: 34.9427 - mae:
34.9427
1/1 [==============================] - 0s 76ms/step - loss: 49.0164 - mae:
49.0164
1/1 [==============================] - 0s 73ms/step - loss: 23.0332 - mae:
23.0332
1/1 [==============================] - 0s 74ms/step - loss: 18.7546 - mae:
18.7546
1/1 [==============================] - 0s 74ms/step - loss: 47.0352 - mae:
47.0352
1/1 [==============================] - 0s 75ms/step - loss: 34.9328 - mae:
34.9328
1/1 [==============================] - 0s 76ms/step - loss: 67.0888 - mae:
67.0888
1/1 [==============================] - 0s 78ms/step - loss: 88.1930 - mae:
88.1930
1/1 [==============================] - 0s 84ms/step - loss: 38.4067 - mae:
38.4067
1/1 [==============================] - 0s 81ms/step - loss: 57.0301 - mae:
57.0301
1/1 [==============================] - 0s 77ms/step - loss: 35.7298 - mae:
35.7298
1/1 [==============================] - 0s 76ms/step - loss: 49.5690 - mae:
49.5690
1/1 [==============================] - 0s 88ms/step - loss: 21.5995 - mae:
21.5995
1/1 [==============================] - 0s 79ms/step - loss: 18.8018 - mae:
18.8018
1/1 [==============================] - 0s 82ms/step - loss: 45.8215 - mae:
45.8215
1/1 [==============================] - 0s 77ms/step - loss: 34.7991 - mae:
34.7991
1/1 [==============================] - 0s 82ms/step - loss: 67.2421 - mae:
67.2421
1/1 [==============================] - 0s 77ms/step - loss: 87.9775 - mae:
87.9775
1/1 [==============================] - 0s 79ms/step - loss: 38.5620 - mae:
38.5620
1/1 [==============================] - 0s 75ms/step - loss: 56.4046 - mae:
56.4046
1/1 [==============================] - 0s 82ms/step - loss: 35.2239 - mae:
35.2239
1/1 [==============================] - 0s 78ms/step - loss: 49.5253 - mae:
```

```
49.5253
1/1 [==============================] - 0s 95ms/step - loss: 22.5188 - mae:
22.5188
1/1 [==============================] - 0s 81ms/step - loss: 18.4780 - mae:
18.4780
1/1 [==============================] - 0s 75ms/step - loss: 46.4876 - mae:
46.4876
1/1 [==============================] - 0s 77ms/step - loss: 34.3181 - mae:
34.3181
1/1 [==============================] - 0s 76ms/step - loss: 67.1508 - mae:
67.1508
1/1 [==============================] - 0s 79ms/step - loss: 88.2002 - mae:
88.2002
1/1 [==============================] - 0s 75ms/step - loss: 38.2076 - mae:
38.2076
1/1 [==============================] - 0s 80ms/step - loss: 56.7480 - mae:
56.7480
1/1 [==============================] - 0s 74ms/step - loss: 34.6971 - mae:
34.6971
1/1 [==============================] - 0s 82ms/step - loss: 49.3359 - mae:
49.3359
1/1 [==============================] - 0s 75ms/step - loss: 22.2372 - mae:
22.2372
1/1 [==============================] - 0s 76ms/step - loss: 19.0679 - mae:
19.0679
1/1 [==============================] - 0s 81ms/step - loss: 45.8540 - mae:
45.8540
1/1 [==============================] - 0s 96ms/step - loss: 35.3292 - mae:
35.3292
1/1 [==============================] - 0s 76ms/step - loss: 66.7043 - mae:
66.7043
1/1 [==============================] - 0s 75ms/step - loss: 88.2440 - mae:
88.2440
1/1 [==============================] - 0s 76ms/step - loss: 38.5091 - mae:
38.5091
1/1 [==============================] - 0s 75ms/step - loss: 56.5203 - mae:
56.5203
1/1 [==============================] - 1s 559ms/step - loss: 35.2709 - mae:
35.2709
1/1 [==============================] - 0s 76ms/step - loss: 49.9620 - mae:
49.9620
1/1 [==============================] - 0s 75ms/step - loss: 21.8652 - mae:
21.8652
1/1 [==============================] - 0s 77ms/step - loss: 18.8599 - mae:
18.8599
1/1 [==============================] - 0s 86ms/step - loss: 46.3198 - mae:
46.3198
1/1 [==============================] - 0s 74ms/step - loss: 34.9155 - mae:
```

```
34.9155
1/1 [==============================] - 0s 77ms/step - loss: 66.3788 - mae:
66.3788
1/1 [==============================] - 0s 75ms/step - loss: 88.0947 - mae:
88.0947
1/1 [==============================] - 0s 74ms/step - loss: 38.6705 - mae:
38.6705
1/1 [==============================] - 0s 74ms/step - loss: 56.7797 - mae:
56.7797
1/1 [==============================] - 0s 75ms/step - loss: 35.5672 - mae:
35.5672
1/1 [==============================] - 0s 76ms/step - loss: 50.1931 - mae:
50.1931
1/1 [==============================] - 0s 76ms/step - loss: 21.7897 - mae:
21.7897
1/1 [==============================] - 0s 78ms/step - loss: 18.6859 - mae:
18.6859
1/1 [==============================] - 0s 75ms/step - loss: 45.8481 - mae:
45.8481
1/1 [==============================] - 0s 78ms/step - loss: 34.8981 - mae:
34.8981
1/1 [==============================] - 0s 79ms/step - loss: 67.0022 - mae:
67.0022
1/1 [==============================] - 0s 79ms/step - loss: 88.1319 - mae:
88.1319
1/1 [==============================] - 0s 76ms/step - loss: 38.4102 - mae:
38.4102
1/1 [==============================] - 0s 76ms/step - loss: 56.8660 - mae:
56.8660
1/1 [==============================] - 0s 78ms/step - loss: 34.9125 - mae:
34.9125
1/1 [==============================] - 0s 76ms/step - loss: 49.7417 - mae:
49.7417
1/1 [==============================] - 0s 79ms/step - loss: 21.9391 - mae:
21.9391
1/1 [==============================] - 0s 75ms/step - loss: 18.0543 - mae:
18.0543
1/1 [==============================] - 0s 75ms/step - loss: 46.2318 - mae:
46.2318
1/1 [==============================] - 0s 85ms/step - loss: 34.5374 - mae:
34.5374
1/1 [==============================] - 0s 78ms/step - loss: 66.9124 - mae:
66.9124
1/1 [==============================] - 0s 74ms/step - loss: 88.1311 - mae:
88.1311
1/1 [==============================] - 0s 74ms/step - loss: 38.4034 - mae:
38.4034
1/1 [==============================] - 0s 75ms/step - loss: 56.7260 - mae:
```

```
56.7260
1/1 [==============================] - 0s 78ms/step - loss: 35.5874 - mae:
35.5874
1/1 [==============================] - 0s 74ms/step - loss: 49.6917 - mae:
49.6917
1/1 [==============================] - 0s 77ms/step - loss: 22.5575 - mae:
22.5575
1/1 [==============================] - 0s 74ms/step - loss: 19.0554 - mae:
19.0554
1/1 [==============================] - 0s 76ms/step - loss: 47.3739 - mae:
47.3739
1/1 [==============================] - 0s 84ms/step - loss: 34.9879 - mae:
34.9879
1/1 [==============================] - 0s 82ms/step - loss: 66.9560 - mae:
66.9560
1/1 [==============================] - 0s 83ms/step - loss: 88.0595 - mae:
88.0595
1/1 [==============================] - 0s 87ms/step - loss: 38.9138 - mae:
38.9138
1/1 [==============================] - 0s 77ms/step - loss: 56.6199 - mae:
56.6199
1/1 [==============================] - 0s 80ms/step - loss: 35.3416 - mae:
35.3416
1/1 [==============================] - 0s 80ms/step - loss: 49.3482 - mae:
49.3482
1/1 [==============================] - 0s 96ms/step - loss: 22.5779 - mae:
22.5779
1/1 [==============================] - 0s 83ms/step - loss: 19.2257 - mae:
19.2257
1/1 [==============================] - 0s 80ms/step - loss: 46.1910 - mae:
46.1910
1/1 [==============================] - 0s 83ms/step - loss: 34.5572 - mae:
34.5572
1/1 [==============================] - 0s 76ms/step - loss: 67.1804 - mae:
67.1804
1/1 [==============================] - 0s 77ms/step - loss: 88.3276 - mae:
88.3276
1/1 [==============================] - 0s 80ms/step - loss: 38.6579 - mae:
38.6579
1/1 [==============================] - 0s 76ms/step - loss: 56.4931 - mae:
56.4931
1/1 [==============================] - 0s 75ms/step - loss: 35.9724 - mae:
35.9724
1/1 [==============================] - 0s 78ms/step - loss: 49.5796 - mae:
49.5796
1/1 [==============================] - 0s 78ms/step - loss: 22.4829 - mae:
22.4829
1/1 [==============================] - 0s 77ms/step - loss: 18.4983 - mae:
```

```
18.4983
1/1 [==============================] - 0s 78ms/step - loss: 45.9632 - mae:
45.9632
1/1 [==============================] - 0s 83ms/step - loss: 34.7134 - mae:
34.7134
1/1 [==============================] - 0s 77ms/step - loss: 67.2033 - mae:
67.2033
1/1 [==============================] - 0s 75ms/step - loss: 87.6403 - mae:
87.6403
1/1 [==============================] - 0s 82ms/step - loss: 38.6456 - mae:
38.6456
1/1 [==============================] - 0s 78ms/step - loss: 56.4005 - mae:
56.4005
1/1 [==============================] - 0s 77ms/step - loss: 35.2253 - mae:
35.2253
1/1 [==============================] - 0s 75ms/step - loss: 49.6064 - mae:
49.6064
1/1 [==============================] - 0s 75ms/step - loss: 23.0798 - mae:
23.0798
1/1 [==============================] - 0s 76ms/step - loss: 19.0755 - mae:
19.0755
1/1 [==============================] - 0s 76ms/step - loss: 46.2253 - mae:
46.2253
1/1 [==============================] - 0s 96ms/step - loss: 34.8495 - mae:
34.8495
1/1 [==============================] - 0s 87ms/step - loss: 67.6291 - mae:
67.6291
1/1 [==============================] - 0s 87ms/step - loss: 88.0886 - mae:
88.0886
1/1 [==============================] - 0s 100ms/step - loss: 38.5369 - mae:
38.5369
1/1 [==============================] - 0s 88ms/step - loss: 56.7997 - mae:
56.7997
1/1 [==============================] - 0s 86ms/step - loss: 34.4713 - mae:
34.4713
1/1 [==============================] - 0s 78ms/step - loss: 48.4577 - mae:
48.4577
1/1 [==============================] - 0s 76ms/step - loss: 22.6045 - mae:
22.6045
1/1 [==============================] - 0s 78ms/step - loss: 19.2455 - mae:
19.2455
1/1 [==============================] - 0s 78ms/step - loss: 46.2621 - mae:
46.2621
1/1 [==============================] - 0s 90ms/step - loss: 34.8602 - mae:
34.8602
1/1 [==============================] - 0s 90ms/step - loss: 66.5912 - mae:
66.5912
1/1 [==============================] - 0s 89ms/step - loss: 88.1830 - mae:
```

```
88.1830
1/1 [==============================] - 0s 87ms/step - loss: 37.9746 - mae:
37.9746
1/1 [==============================] - 0s 97ms/step - loss: 56.5824 - mae:
56.5824
1/1 [==============================] - 0s 91ms/step - loss: 34.7584 - mae:
34.7584
1/1 [==============================] - 0s 79ms/step - loss: 49.0341 - mae:
49.0341
1/1 [==============================] - 0s 76ms/step - loss: 22.5258 - mae:
22.5258
1/1 [==============================] - 0s 75ms/step - loss: 19.0214 - mae:
19.0214
1/1 [==============================] - 0s 77ms/step - loss: 46.2461 - mae:
46.2461
1/1 [==============================] - 0s 90ms/step - loss: 34.9502 - mae:
34.9502
1/1 [==============================] - 0s 90ms/step - loss: 66.7355 - mae:
66.7355
1/1 [==============================] - 0s 86ms/step - loss: 88.0255 - mae:
88.0255
1/1 [==============================] - 0s 87ms/step - loss: 36.4557 - mae:
36.4557
1/1 [==============================] - 0s 95ms/step - loss: 56.7251 - mae:
56.7251
1/1 [==============================] - 0s 87ms/step - loss: 34.6996 - mae:
34.6996
1/1 [==============================] - 0s 75ms/step - loss: 49.6846 - mae:
49.6846
1/1 [==============================] - 0s 82ms/step - loss: 22.7642 - mae:
22.7642
1/1 [==============================] - 0s 77ms/step - loss: 18.8216 - mae:
18.8216
1/1 [==============================] - 0s 81ms/step - loss: 46.1217 - mae:
46.1217
1/1 [==============================] - 0s 86ms/step - loss: 34.8597 - mae:
34.8597
1/1 [==============================] - 0s 86ms/step - loss: 67.1105 - mae:
67.1105
1/1 [==============================] - 0s 88ms/step - loss: 87.8551 - mae:
87.8551
1/1 [==============================] - 0s 90ms/step - loss: 39.6312 - mae:
39.6312
1/1 [==============================] - 0s 87ms/step - loss: 56.5049 - mae:
56.5049
1/1 [==============================] - 0s 90ms/step - loss: 34.5635 - mae:
34.5635
1/1 [==============================] - 0s 76ms/step - loss: 50.2150 - mae:
```

```
50.2150
1/1 [==============================] - 0s 87ms/step - loss: 22.5123 - mae:
22.5123
1/1 [==============================] - 0s 78ms/step - loss: 18.5688 - mae:
18.5688
1/1 [==============================] - 0s 76ms/step - loss: 46.2216 - mae:
46.2216
1/1 [==============================] - 0s 96ms/step - loss: 34.9546 - mae:
34.9546
1/1 [==============================] - 0s 89ms/step - loss: 66.5849 - mae:
66.5849
1/1 [==============================] - 0s 86ms/step - loss: 88.1375 - mae:
88.1375
1/1 [==============================] - 0s 86ms/step - loss: 38.3772 - mae:
38.3772
1/1 [==============================] - 0s 89ms/step - loss: 56.1169 - mae:
56.1169
1/1 [==============================] - 0s 87ms/step - loss: 34.1358 - mae:
34.1358
1/1 [==============================] - 0s 74ms/step - loss: 48.8332 - mae:
48.8332
1/1 [==============================] - 0s 75ms/step - loss: 22.6294 - mae:
22.6294
1/1 [==============================] - 0s 76ms/step - loss: 18.6585 - mae:
18.6585
1/1 [==============================] - 0s 83ms/step - loss: 46.2141 - mae:
46.2141
1/1 [==============================] - 0s 93ms/step - loss: 34.8253 - mae:
34.8253
1/1 [==============================] - 0s 90ms/step - loss: 67.1039 - mae:
67.1039
1/1 [==============================] - 0s 89ms/step - loss: 88.1191 - mae:
88.1191
1/1 [==============================] - 0s 88ms/step - loss: 38.3087 - mae:
38.3087
1/1 [==============================] - 0s 91ms/step - loss: 56.3362 - mae:
56.3362
1/1 [==============================] - 0s 94ms/step - loss: 34.9705 - mae:
34.9705
1/1 [==============================] - 0s 73ms/step - loss: 48.6663 - mae:
48.6663
1/1 [==============================] - 0s 76ms/step - loss: 22.6112 - mae:
22.6112
1/1 [==============================] - 0s 81ms/step - loss: 18.3549 - mae:
18.3549
1/1 [==============================] - 0s 74ms/step - loss: 46.3202 - mae:
46.3202
1/1 [==============================] - 0s 88ms/step - loss: 35.5259 - mae:
```

```
35.5259
1/1 [==============================] - 0s 91ms/step - loss: 66.9913 - mae:
66.9913
1/1 [==============================] - 0s 89ms/step - loss: 87.9710 - mae:
87.9710
1/1 [==============================] - 0s 85ms/step - loss: 37.5806 - mae:
37.5806
1/1 [==============================] - 0s 86ms/step - loss: 57.0066 - mae:
57.0066
1/1 [==============================] - 0s 87ms/step - loss: 35.1389 - mae:
35.1389
1/1 [==============================] - 0s 76ms/step - loss: 49.1122 - mae:
49.1122
1/1 [==============================] - 0s 77ms/step - loss: 22.7602 - mae:
22.7602
1/1 [==============================] - 0s 86ms/step - loss: 19.0299 - mae:
19.0299
1/1 [==============================] - 0s 77ms/step - loss: 46.4903 - mae:
46.4903
1/1 [==============================] - 0s 95ms/step - loss: 35.0863 - mae:
35.0863
1/1 [==============================] - 0s 101ms/step - loss: 66.9302 - mae:
66.9302
1/1 [==============================] - 0s 98ms/step - loss: 88.2090 - mae:
88.2090
1/1 [==============================] - 0s 93ms/step - loss: 38.8914 - mae:
38.8914
1/1 [==============================] - 0s 94ms/step - loss: 57.2310 - mae:
57.2310
1/1 [==============================] - 0s 96ms/step - loss: 34.6236 - mae:
34.6236
1/1 [==============================] - 0s 80ms/step - loss: 49.8190 - mae:
49.8190
1/1 [==============================] - 0s 82ms/step - loss: 22.6568 - mae:
22.6568
1/1 [==============================] - 0s 79ms/step - loss: 18.8661 - mae:
18.8661
1/1 [==============================] - 0s 76ms/step - loss: 45.9454 - mae:
45.9454
1/1 [==============================] - 0s 89ms/step - loss: 34.8293 - mae:
34.8293
1/1 [==============================] - 0s 90ms/step - loss: 67.1056 - mae:
67.1056
1/1 [==============================] - 0s 88ms/step - loss: 88.4018 - mae:
88.4018
1/1 [==============================] - 0s 99ms/step - loss: 38.3328 - mae:
38.3328
1/1 [==============================] - 0s 96ms/step - loss: 56.6867 - mae:
```

```
56.6867
1/1 [==============================] - 0s 85ms/step - loss: 35.0608 - mae:
35.0608
1/1 [==============================] - 0s 80ms/step - loss: 49.5959 - mae:
49.5959
1/1 [==============================] - 0s 76ms/step - loss: 22.5533 - mae:
22.5533
1/1 [==============================] - 0s 76ms/step - loss: 18.8762 - mae:
18.8762
1/1 [==============================] - 0s 76ms/step - loss: 45.9046 - mae:
45.9046
1/1 [==============================] - 0s 87ms/step - loss: 34.6751 - mae:
34.6751
1/1 [==============================] - 0s 85ms/step - loss: 67.1877 - mae:
67.1877
1/1 [==============================] - 0s 88ms/step - loss: 88.0485 - mae:
88.0485
1/1 [==============================] - 0s 87ms/step - loss: 38.5138 - mae:
38.5138
1/1 [==============================] - 0s 96ms/step - loss: 56.1840 - mae:
56.1840
1/1 [==============================] - 0s 90ms/step - loss: 34.8090 - mae:
34.8090
1/1 [==============================] - 0s 78ms/step - loss: 49.5267 - mae:
49.5267
1/1 [==============================] - 0s 79ms/step - loss: 22.2738 - mae:
22.2738
1/1 [==============================] - 0s 73ms/step - loss: 18.6622 - mae:
18.6622
1/1 [==============================] - 0s 78ms/step - loss: 46.4267 - mae:
46.4267
1/1 [==============================] - 0s 89ms/step - loss: 34.6574 - mae:
34.6574
1/1 [==============================] - 0s 94ms/step - loss: 66.9795 - mae:
66.9795
1/1 [==============================] - 0s 86ms/step - loss: 88.1558 - mae:
88.1558
1/1 [==============================] - 0s 86ms/step - loss: 38.6222 - mae:
38.6222
1/1 [==============================] - 0s 91ms/step - loss: 55.8183 - mae:
55.8183
1/1 [==============================] - 0s 91ms/step - loss: 35.6696 - mae:
35.6696
1/1 [==============================] - 0s 74ms/step - loss: 49.6543 - mae:
49.6543
1/1 [==============================] - 0s 85ms/step - loss: 22.6213 - mae:
22.6213
1/1 [==============================] - 0s 79ms/step - loss: 17.8565 - mae:
```

```
17.8565
1/1 [==============================] - 0s 77ms/step - loss: 46.1856 - mae:
46.1856
1/1 [==============================] - 0s 93ms/step - loss: 34.9302 - mae:
34.9302
1/1 [==============================] - 0s 92ms/step - loss: 67.0507 - mae:
67.0507
1/1 [==============================] - 0s 89ms/step - loss: 88.2833 - mae:
88.2833
1/1 [==============================] - 0s 91ms/step - loss: 38.2319 - mae:
38.2319
1/1 [==============================] - 0s 87ms/step - loss: 56.5580 - mae:
56.5580
1/1 [==============================] - 0s 90ms/step - loss: 35.1141 - mae:
35.1141
1/1 [==============================] - 0s 77ms/step - loss: 49.7550 - mae:
49.7550
1/1 [==============================] - 0s 77ms/step - loss: 22.5691 - mae:
22.5691
1/1 [==============================] - 0s 81ms/step - loss: 18.6774 - mae:
18.6774
1/1 [==============================] - 0s 74ms/step - loss: 47.2736 - mae:
47.2736
1/1 [==============================] - 0s 89ms/step - loss: 35.0407 - mae:
35.0407
1/1 [==============================] - 0s 88ms/step - loss: 66.9671 - mae:
66.9671
1/1 [==============================] - 0s 90ms/step - loss: 88.3016 - mae:
88.3016
1/1 [==============================] - 0s 85ms/step - loss: 39.8552 - mae:
39.8552
1/1 [==============================] - 0s 89ms/step - loss: 56.7037 - mae:
56.7037
1/1 [==============================] - 0s 91ms/step - loss: 34.8166 - mae:
34.8166
1/1 [==============================] - 0s 77ms/step - loss: 49.7667 - mae:
49.7667
1/1 [==============================] - 0s 79ms/step - loss: 22.3748 - mae:
22.3748
1/1 [==============================] - 0s 79ms/step - loss: 18.6093 - mae:
18.6093
1/1 [==============================] - 0s 76ms/step - loss: 46.2962 - mae:
46.2962
1/1 [==============================] - 0s 92ms/step - loss: 34.7128 - mae:
34.7128
1/1 [==============================] - 0s 92ms/step - loss: 67.1190 - mae:
67.1190
1/1 [==============================] - 0s 86ms/step - loss: 88.1014 - mae:
```

```
88.1014
1/1 [==============================] - 0s 87ms/step - loss: 39.0779 - mae:
39.0779
1/1 [==============================] - 0s 92ms/step - loss: 56.8198 - mae:
56.8198
1/1 [==============================] - 0s 88ms/step - loss: 35.4246 - mae:
35.4246
1/1 [==============================] - 0s 79ms/step - loss: 49.8725 - mae:
49.8725
1/1 [==============================] - 0s 76ms/step - loss: 22.3957 - mae:
22.3957
1/1 [==============================] - 0s 81ms/step - loss: 18.7235 - mae:
18.7235
1/1 [==============================] - 0s 74ms/step - loss: 46.4383 - mae:
46.4383
1/1 [==============================] - 0s 92ms/step - loss: 34.8224 - mae:
34.8224
1/1 [==============================] - 0s 92ms/step - loss: 66.7892 - mae:
66.7892
1/1 [==============================] - 0s 89ms/step - loss: 88.2230 - mae:
88.2230
1/1 [==============================] - 0s 92ms/step - loss: 38.5802 - mae:
38.5802
1/1 [==============================] - 0s 93ms/step - loss: 57.0950 - mae:
57.0950
1/1 [==============================] - 0s 86ms/step - loss: 34.7234 - mae:
34.7234
1/1 [==============================] - 0s 76ms/step - loss: 50.1280 - mae:
50.1280
1/1 [==============================] - 0s 74ms/step - loss: 22.6608 - mae:
22.6608
1/1 [==============================] - 0s 74ms/step - loss: 18.9569 - mae:
18.9569
1/1 [==============================] - 0s 77ms/step - loss: 45.8820 - mae:
45.8820
1/1 [==============================] - 0s 88ms/step - loss: 34.8336 - mae:
34.8336
1/1 [==============================] - 0s 90ms/step - loss: 66.5215 - mae:
66.5215
1/1 [==============================] - 0s 93ms/step - loss: 88.4938 - mae:
88.4938
1/1 [==============================] - 0s 92ms/step - loss: 37.9964 - mae:
37.9964
1/1 [==============================] - 0s 91ms/step - loss: 55.4774 - mae:
55.4774
1/1 [==============================] - 0s 89ms/step - loss: 34.2545 - mae:
34.2545
1/1 [==============================] - 0s 76ms/step - loss: 50.3114 - mae:
```

```
50.3114
1/1 [==============================] - 0s 76ms/step - loss: 22.8644 - mae:
22.8644
1/1 [==============================] - 0s 74ms/step - loss: 16.7087 - mae:
16.7087
1/1 [==============================] - 0s 86ms/step - loss: 45.9456 - mae:
45.9456
1/1 [==============================] - 0s 88ms/step - loss: 34.9617 - mae:
34.9617
1/1 [==============================] - 0s 88ms/step - loss: 67.2022 - mae:
67.2022
1/1 [==============================] - 0s 93ms/step - loss: 88.2442 - mae:
88.2442
1/1 [==============================] - 0s 86ms/step - loss: 38.5459 - mae:
38.5459
1/1 [==============================] - 0s 89ms/step - loss: 56.6138 - mae:
56.6138
1/1 [==============================] - 0s 96ms/step - loss: 35.1766 - mae:
35.1766
1/1 [==============================] - 0s 76ms/step - loss: 49.0326 - mae:
49.0326
1/1 [==============================] - 0s 76ms/step - loss: 22.6556 - mae:
22.6556
1/1 [==============================] - 0s 77ms/step - loss: 18.5638 - mae:
18.5638
1/1 [==============================] - 0s 76ms/step - loss: 46.0021 - mae:
46.0021
1/1 [==============================] - 0s 86ms/step - loss: 34.9734 - mae:
34.9734
1/1 [==============================] - 0s 90ms/step - loss: 66.7854 - mae:
66.7854
1/1 [==============================] - 0s 99ms/step - loss: 88.3297 - mae:
88.3297
1/1 [==============================] - 0s 87ms/step - loss: 38.6703 - mae:
38.6703
1/1 [==============================] - 0s 90ms/step - loss: 56.6635 - mae:
56.6635
1/1 [==============================] - 0s 93ms/step - loss: 35.1400 - mae:
35.1400
1/1 [==============================] - 0s 78ms/step - loss: 49.4287 - mae:
49.4287
1/1 [==============================] - 0s 74ms/step - loss: 23.2690 - mae:
23.2690
1/1 [==============================] - 0s 91ms/step - loss: 18.4927 - mae:
18.4927
1/1 [==============================] - 0s 75ms/step - loss: 46.2270 - mae:
46.2270
1/1 [==============================] - 0s 88ms/step - loss: 34.7604 - mae:
```

```
34.7604
1/1 [==============================] - 0s 95ms/step - loss: 67.1415 - mae:
67.1415
1/1 [==============================] - 0s 92ms/step - loss: 88.7022 - mae:
88.7022
1/1 [==============================] - 0s 93ms/step - loss: 38.5521 - mae:
38.5521
1/1 [==============================] - 0s 89ms/step - loss: 57.1458 - mae:
57.1458
1/1 [==============================] - 0s 87ms/step - loss: 33.5285 - mae:
33.5285
1/1 [==============================] - 0s 78ms/step - loss: 49.9499 - mae:
49.9499
1/1 [==============================] - 0s 74ms/step - loss: 23.4801 - mae:
23.4801
1/1 [==============================] - 0s 77ms/step - loss: 18.8431 - mae:
18.8431
1/1 [==============================] - 0s 77ms/step - loss: 46.2672 - mae:
46.2672
1/1 [==============================] - 0s 96ms/step - loss: 34.9480 - mae:
34.9480
1/1 [==============================] - 0s 96ms/step - loss: 66.9427 - mae:
66.9427
1/1 [==============================] - 0s 88ms/step - loss: 88.1000 - mae:
88.1000
1/1 [==============================] - 0s 98ms/step - loss: 38.4836 - mae:
38.4836
1/1 [==============================] - 0s 87ms/step - loss: 56.7887 - mae:
56.7887
1/1 [==============================] - 0s 94ms/step - loss: 35.0481 - mae:
35.0481
1/1 [==============================] - 0s 83ms/step - loss: 49.8678 - mae:
49.8678
1/1 [==============================] - 0s 80ms/step - loss: 22.0411 - mae:
22.0411
1/1 [==============================] - 0s 90ms/step - loss: 18.7920 - mae:
18.7920
1/1 [==============================] - 0s 74ms/step - loss: 46.1231 - mae:
46.1231
1/1 [==============================] - 0s 89ms/step - loss: 35.2631 - mae:
35.2631
1/1 [==============================] - 0s 86ms/step - loss: 67.1973 - mae:
67.1973
1/1 [==============================] - 0s 99ms/step - loss: 88.5977 - mae:
88.5977
1/1 [==============================] - 0s 87ms/step - loss: 37.7770 - mae:
37.7770
1/1 [==============================] - 0s 86ms/step - loss: 56.6053 - mae:
```

```
56.6053
1/1 [==============================] - 0s 95ms/step - loss: 35.1492 - mae:
35.1492
1/1 [==============================] - 0s 79ms/step - loss: 49.6628 - mae:
49.6628
1/1 [==============================] - 0s 81ms/step - loss: 22.5670 - mae:
22.5670
1/1 [==============================] - 0s 80ms/step - loss: 18.4252 - mae:
18.4252
1/1 [==============================] - 0s 76ms/step - loss: 45.9403 - mae:
45.9403
1/1 [==============================] - 0s 90ms/step - loss: 35.2810 - mae:
35.2810
1/1 [==============================] - 0s 90ms/step - loss: 67.1086 - mae:
67.1086
1/1 [==============================] - 0s 89ms/step - loss: 88.3554 - mae:
88.3554
1/1 [==============================] - 0s 85ms/step - loss: 38.3973 - mae:
38.3973
1/1 [==============================] - 0s 89ms/step - loss: 56.4208 - mae:
56.4208
1/1 [==============================] - 0s 92ms/step - loss: 34.9893 - mae:
34.9893
1/1 [==============================] - 0s 80ms/step - loss: 49.0940 - mae:
49.0940
1/1 [==============================] - 0s 117ms/step - loss: 22.4523 - mae:
22.4523
1/1 [==============================] - 0s 76ms/step - loss: 18.5609 - mae:
18.5609
1/1 [==============================] - 0s 76ms/step - loss: 46.3521 - mae:
46.3521
1/1 [==============================] - 0s 87ms/step - loss: 34.8383 - mae:
34.8383
1/1 [==============================] - 0s 86ms/step - loss: 66.9300 - mae:
66.9300
1/1 [==============================] - 0s 86ms/step - loss: 88.2994 - mae:
88.2994
1/1 [==============================] - 0s 93ms/step - loss: 38.7403 - mae:
38.7403
1/1 [==============================] - 0s 91ms/step - loss: 56.7223 - mae:
56.7223
1/1 [==============================] - 0s 94ms/step - loss: 35.6657 - mae:
35.6657
1/1 [==============================] - 0s 80ms/step - loss: 49.5377 - mae:
49.5377
1/1 [==============================] - 0s 76ms/step - loss: 22.5647 - mae:
22.5647
1/1 [==============================] - 0s 73ms/step - loss: 18.9723 - mae:
```

```
18.9723
1/1 [==============================] - 0s 78ms/step - loss: 46.2419 - mae:
46.2419
1/1 [==============================] - 0s 86ms/step - loss: 34.8540 - mae:
34.8540
1/1 [==============================] - 0s 89ms/step - loss: 67.1590 - mae:
67.1590
1/1 [==============================] - 0s 86ms/step - loss: 88.3969 - mae:
88.3969
1/1 [==============================] - 0s 87ms/step - loss: 38.1851 - mae:
38.1851
1/1 [==============================] - 0s 85ms/step - loss: 56.4670 - mae:
56.4670
1/1 [==============================] - 0s 92ms/step - loss: 34.7343 - mae:
34.7343
1/1 [==============================] - 0s 76ms/step - loss: 49.3271 - mae:
49.3271
1/1 [==============================] - 0s 104ms/step - loss: 22.3573 - mae:
22.3573
1/1 [==============================] - 0s 77ms/step - loss: 18.2759 - mae:
18.2759
1/1 [==============================] - 0s 75ms/step - loss: 46.2198 - mae:
46.2198
1/1 [==============================] - 0s 74ms/step - loss: 35.0400 - mae:
35.0400
1/1 [==============================] - 0s 75ms/step - loss: 67.5758 - mae:
67.5758
1/1 [==============================] - 0s 77ms/step - loss: 89.0080 - mae:
89.0080
1/1 [==============================] - 0s 82ms/step - loss: 38.7360 - mae:
38.7360
1/1 [==============================] - 0s 81ms/step - loss: 56.1560 - mae:
56.1560
1/1 [==============================] - 0s 74ms/step - loss: 35.9568 - mae:
35.9568
1/1 [==============================] - 0s 74ms/step - loss: 50.4647 - mae:
50.4647
1/1 [==============================] - 0s 83ms/step - loss: 22.9917 - mae:
22.9917
1/1 [==============================] - 0s 79ms/step - loss: 17.5685 - mae:
17.5685
1/1 [==============================] - 0s 79ms/step - loss: 46.3004 - mae:
46.3004
1/1 [==============================] - 0s 75ms/step - loss: 34.6270 - mae:
34.6270
1/1 [==============================] - 0s 77ms/step - loss: 67.0428 - mae:
67.0428
1/1 [==============================] - 0s 75ms/step - loss: 87.8871 - mae:
```

```
87.8871
1/1 [==============================] - 0s 82ms/step - loss: 39.0041 - mae:
39.0041
1/1 [==============================] - 0s 81ms/step - loss: 56.6565 - mae:
56.6565
1/1 [==============================] - 0s 75ms/step - loss: 35.6857 - mae:
35.6857
1/1 [==============================] - 0s 81ms/step - loss: 49.4129 - mae:
49.4129
1/1 [==============================] - 0s 77ms/step - loss: 22.8579 - mae:
22.8579
1/1 [==============================] - 0s 80ms/step - loss: 19.2741 - mae:
19.2741
1/1 [==============================] - 0s 74ms/step - loss: 45.9546 - mae:
45.9546
1/1 [==============================] - 0s 77ms/step - loss: 34.9881 - mae:
34.9881
1/1 [==============================] - 0s 76ms/step - loss: 67.0815 - mae:
67.0815
1/1 [==============================] - 0s 79ms/step - loss: 88.2175 - mae:
88.2175
1/1 [==============================] - 0s 77ms/step - loss: 38.8763 - mae:
38.8763
1/1 [==============================] - 0s 76ms/step - loss: 56.7324 - mae:
56.7324
1/1 [==============================] - 0s 77ms/step - loss: 35.0720 - mae:
35.0720
1/1 [==============================] - 0s 75ms/step - loss: 49.6972 - mae:
49.6972
1/1 [==============================] - 0s 76ms/step - loss: 22.2592 - mae:
22.2592
1/1 [==============================] - 0s 74ms/step - loss: 18.9196 - mae:
18.9196
1/1 [==============================] - 0s 76ms/step - loss: 46.3735 - mae:
46.3735
1/1 [==============================] - 0s 79ms/step - loss: 34.8210 - mae:
34.8210
1/1 [==============================] - 0s 78ms/step - loss: 67.0598 - mae:
67.0598
1/1 [==============================] - 0s 79ms/step - loss: 88.0814 - mae:
88.0814
1/1 [==============================] - 0s 78ms/step - loss: 38.1593 - mae:
38.1593
1/1 [==============================] - 0s 74ms/step - loss: 56.4500 - mae:
56.4500
1/1 [==============================] - 0s 74ms/step - loss: 34.6898 - mae:
34.6898
1/1 [==============================] - 0s 78ms/step - loss: 49.9030 - mae:
```

```
49.9030
1/1 [==============================] - 0s 78ms/step - loss: 22.7053 - mae:
22.7053
1/1 [==============================] - 0s 81ms/step - loss: 18.9385 - mae:
18.9385
1/1 [==============================] - 0s 73ms/step - loss: 46.4824 - mae:
46.4824
1/1 [==============================] - 0s 82ms/step - loss: 34.9776 - mae:
34.9776
1/1 [==============================] - 0s 75ms/step - loss: 67.0533 - mae:
67.0533
1/1 [==============================] - 0s 76ms/step - loss: 87.6197 - mae:
87.6197
1/1 [==============================] - 0s 76ms/step - loss: 38.5647 - mae:
38.5647
1/1 [==============================] - 0s 75ms/step - loss: 56.4234 - mae:
56.4234
1/1 [==============================] - 0s 84ms/step - loss: 35.1668 - mae:
35.1668
1/1 [==============================] - 1s 563ms/step - loss: 49.8984 - mae:
49.8984
1/1 [==============================] - 0s 83ms/step - loss: 22.3943 - mae:
22.3943
1/1 [==============================] - 0s 76ms/step - loss: 18.4913 - mae:
18.4913
1/1 [==============================] - 0s 80ms/step - loss: 46.1765 - mae:
46.1765
1/1 [==============================] - 0s 86ms/step - loss: 35.0569 - mae:
35.0569
1/1 [==============================] - 0s 81ms/step - loss: 67.2417 - mae:
67.2417
1/1 [==============================] - 0s 80ms/step - loss: 88.1796 - mae:
88.1796
1/1 [==============================] - 0s 77ms/step - loss: 38.1912 - mae:
38.1912
1/1 [==============================] - 0s 78ms/step - loss: 56.8435 - mae:
56.8435
1/1 [==============================] - 0s 82ms/step - loss: 35.2090 - mae:
35.2090
1/1 [==============================] - 0s 75ms/step - loss: 49.7270 - mae:
49.7270
1/1 [==============================] - 0s 75ms/step - loss: 22.6245 - mae:
22.6245
1/1 [==============================] - 0s 75ms/step - loss: 18.9734 - mae:
18.9734
1/1 [==============================] - 0s 76ms/step - loss: 46.1322 - mae:
46.1322
1/1 [==============================] - 0s 88ms/step - loss: 34.8839 - mae:
```

```
34.8839
1/1 [==============================] - 0s 83ms/step - loss: 66.6495 - mae:
66.6495
1/1 [==============================] - 0s 81ms/step - loss: 87.9275 - mae:
87.9275
1/1 [==============================] - 0s 85ms/step - loss: 38.6981 - mae:
38.6981
1/1 [==============================] - 0s 82ms/step - loss: 56.7375 - mae:
56.7375
1/1 [==============================] - 0s 78ms/step - loss: 34.6390 - mae:
34.6390
1/1 [==============================] - 0s 77ms/step - loss: 49.3037 - mae:
49.3037
1/1 [==============================] - 0s 80ms/step - loss: 21.9444 - mae:
21.9444
1/1 [==============================] - 0s 76ms/step - loss: 19.0896 - mae:
19.0896
1/1 [==============================] - 0s 75ms/step - loss: 46.1298 - mae:
46.1298
1/1 [==============================] - 0s 76ms/step - loss: 34.3400 - mae:
34.3400
1/1 [==============================] - 0s 75ms/step - loss: 66.9500 - mae:
66.9500
1/1 [==============================] - 0s 75ms/step - loss: 88.1064 - mae:
88.1064
1/1 [==============================] - 0s 88ms/step - loss: 38.3242 - mae:
38.3242
1/1 [==============================] - 0s 80ms/step - loss: 56.8456 - mae:
56.8456
1/1 [==============================] - 0s 75ms/step - loss: 34.7460 - mae:
34.7460
1/1 [==============================] - 0s 82ms/step - loss: 49.7829 - mae:
49.7829
1/1 [==============================] - 0s 77ms/step - loss: 22.3604 - mae:
22.3604
1/1 [==============================] - 0s 80ms/step - loss: 19.1929 - mae:
19.1929
1/1 [==============================] - 0s 78ms/step - loss: 46.6205 - mae:
46.6205
1/1 [==============================] - 0s 78ms/step - loss: 35.0596 - mae:
35.0596
1/1 [==============================] - 0s 76ms/step - loss: 67.2562 - mae:
67.2562
1/1 [==============================] - 0s 78ms/step - loss: 88.1293 - mae:
88.1293
1/1 [==============================] - 0s 74ms/step - loss: 38.1404 - mae:
38.1404
1/1 [==============================] - 0s 79ms/step - loss: 56.4033 - mae:
```

```
56.4033
1/1 [==============================] - 0s 76ms/step - loss: 35.7738 - mae:
35.7738
1/1 [==============================] - 0s 76ms/step - loss: 49.8150 - mae:
49.8150
1/1 [==============================] - 0s 77ms/step - loss: 22.7053 - mae:
22.7053
1/1 [==============================] - 0s 75ms/step - loss: 18.8134 - mae:
18.8134
1/1 [==============================] - 0s 84ms/step - loss: 45.5147 - mae:
45.5147
1/1 [==============================] - 0s 81ms/step - loss: 34.8094 - mae:
34.8094
1/1 [==============================] - 0s 76ms/step - loss: 66.9724 - mae:
66.9724
1/1 [==============================] - 0s 79ms/step - loss: 88.1167 - mae:
88.1167
1/1 [==============================] - 0s 80ms/step - loss: 38.9134 - mae:
38.9134
1/1 [==============================] - 0s 81ms/step - loss: 57.4340 - mae:
57.4340
1/1 [==============================] - 0s 83ms/step - loss: 34.6283 - mae:
34.6283
1/1 [==============================] - 0s 82ms/step - loss: 49.4599 - mae:
49.4599
1/1 [==============================] - 0s 78ms/step - loss: 22.1800 - mae:
22.1800
1/1 [==============================] - 0s 80ms/step - loss: 19.0761 - mae:
19.0761
1/1 [==============================] - 0s 76ms/step - loss: 46.4149 - mae:
46.4149
1/1 [==============================] - 0s 78ms/step - loss: 34.9377 - mae:
34.9377
1/1 [==============================] - 0s 80ms/step - loss: 67.1411 - mae:
67.1411
1/1 [==============================] - 0s 84ms/step - loss: 88.1874 - mae:
88.1874
1/1 [==============================] - 0s 79ms/step - loss: 37.8503 - mae:
37.8503
1/1 [==============================] - 0s 77ms/step - loss: 56.4509 - mae:
56.4509
1/1 [==============================] - 0s 82ms/step - loss: 34.7569 - mae:
34.7569
1/1 [==============================] - 0s 79ms/step - loss: 49.8147 - mae:
49.8147
1/1 [==============================] - 0s 80ms/step - loss: 22.2852 - mae:
22.2852
1/1 [==============================] - 0s 77ms/step - loss: 18.3340 - mae:
```

```
18.3340
1/1 [==============================] - 0s 83ms/step - loss: 46.2818 - mae:
46.2818
1/1 [==============================] - 0s 82ms/step - loss: 34.7314 - mae:
34.7314
1/1 [==============================] - 0s 77ms/step - loss: 66.9466 - mae:
66.9466
1/1 [==============================] - 0s 78ms/step - loss: 88.1109 - mae:
88.1109
1/1 [==============================] - 0s 76ms/step - loss: 38.1915 - mae:
38.1915
1/1 [==============================] - 0s 82ms/step - loss: 56.4787 - mae:
56.4787
1/1 [==============================] - 0s 76ms/step - loss: 34.4067 - mae:
34.4067
1/1 [==============================] - 0s 78ms/step - loss: 50.1211 - mae:
50.1211
1/1 [==============================] - 0s 77ms/step - loss: 22.5559 - mae:
22.5559
1/1 [==============================] - 0s 77ms/step - loss: 18.8602 - mae:
18.8602
1/1 [==============================] - 0s 78ms/step - loss: 46.2116 - mae:
46.2116
1/1 [==============================] - 0s 75ms/step - loss: 34.7787 - mae:
34.7787
1/1 [==============================] - 0s 76ms/step - loss: 66.9547 - mae:
66.9547
1/1 [==============================] - 0s 83ms/step - loss: 88.2806 - mae:
88.2806
1/1 [==============================] - 0s 80ms/step - loss: 37.8442 - mae:
37.8442
1/1 [==============================] - 0s 76ms/step - loss: 56.7391 - mae:
56.7391
1/1 [==============================] - 0s 82ms/step - loss: 35.0427 - mae:
35.0427
1/1 [==============================] - 0s 81ms/step - loss: 49.7293 - mae:
49.7293
1/1 [==============================] - 1s 563ms/step - loss: 22.5147 - mae:
22.5147
1/1 [==============================] - 0s 79ms/step - loss: 18.7021 - mae:
18.7021
1/1 [==============================] - 0s 88ms/step - loss: 46.2304 - mae:
46.2304
1/1 [==============================] - 0s 81ms/step - loss: 34.8466 - mae:
34.8466
1/1 [==============================] - 0s 82ms/step - loss: 67.1916 - mae:
67.1916
1/1 [==============================] - 0s 81ms/step - loss: 88.3648 - mae:
```

```
88.3648
1/1 [==============================] - 0s 75ms/step - loss: 38.0927 - mae:
38.0927
1/1 [==============================] - 0s 79ms/step - loss: 56.6449 - mae:
56.6449
1/1 [==============================] - 0s 86ms/step - loss: 35.3549 - mae:
35.3549
1/1 [==============================] - 0s 81ms/step - loss: 49.2022 - mae:
49.2022
1/1 [==============================] - 0s 79ms/step - loss: 22.1139 - mae:
22.1139
1/1 [==============================] - 0s 84ms/step - loss: 18.8106 - mae:
18.8106
1/1 [==============================] - 0s 76ms/step - loss: 46.0293 - mae:
46.0293
1/1 [==============================] - 0s 80ms/step - loss: 35.0010 - mae:
35.0010
1/1 [==============================] - 0s 81ms/step - loss: 67.5955 - mae:
67.5955
1/1 [==============================] - 0s 83ms/step - loss: 88.5224 - mae:
88.5224
1/1 [==============================] - 0s 84ms/step - loss: 38.6455 - mae:
38.6455
1/1 [==============================] - 0s 93ms/step - loss: 56.5819 - mae:
56.5819
1/1 [==============================] - 0s 85ms/step - loss: 33.9599 - mae:
33.9599
1/1 [==============================] - 0s 86ms/step - loss: 48.7827 - mae:
48.7827
1/1 [==============================] - 0s 82ms/step - loss: 22.8912 - mae:
22.8912
1/1 [==============================] - 0s 77ms/step - loss: 19.1955 - mae:
19.1955
1/1 [==============================] - 0s 80ms/step - loss: 45.9786 - mae:
45.9786
1/1 [==============================] - 0s 87ms/step - loss: 34.9977 - mae:
34.9977
1/1 [==============================] - 0s 78ms/step - loss: 67.0870 - mae:
67.0870
1/1 [==============================] - 0s 83ms/step - loss: 88.2025 - mae:
88.2025
1/1 [==============================] - 0s 85ms/step - loss: 38.8166 - mae:
38.8166
1/1 [==============================] - 0s 80ms/step - loss: 56.6338 - mae:
56.6338
1/1 [==============================] - 0s 81ms/step - loss: 35.0436 - mae:
35.0436
1/1 [==============================] - 0s 81ms/step - loss: 49.9774 - mae:
```

```
49.9774
1/1 [==============================] - 0s 76ms/step - loss: 22.8368 - mae:
22.8368
1/1 [==============================] - 0s 81ms/step - loss: 19.0821 - mae:
19.0821
1/1 [==============================] - 0s 80ms/step - loss: 46.2743 - mae:
46.2743
1/1 [==============================] - 0s 76ms/step - loss: 34.9863 - mae:
34.9863
1/1 [==============================] - 0s 86ms/step - loss: 67.1616 - mae:
67.1616
1/1 [==============================] - 0s 81ms/step - loss: 88.3359 - mae:
88.3359
1/1 [==============================] - 0s 77ms/step - loss: 37.9192 - mae:
37.9192
1/1 [==============================] - 0s 81ms/step - loss: 56.3068 - mae:
56.3068
1/1 [==============================] - 0s 76ms/step - loss: 35.3036 - mae:
35.3036
1/1 [==============================] - 0s 74ms/step - loss: 50.0749 - mae:
50.0749
1/1 [==============================] - 0s 81ms/step - loss: 22.6029 - mae:
22.6029
1/1 [==============================] - 0s 74ms/step - loss: 18.6100 - mae:
18.6100
1/1 [==============================] - 0s 75ms/step - loss: 46.1609 - mae:
46.1609
1/1 [==============================] - 0s 80ms/step - loss: 34.7106 - mae:
34.7106
1/1 [==============================] - 0s 77ms/step - loss: 66.9368 - mae:
66.9368
1/1 [==============================] - 0s 83ms/step - loss: 88.3810 - mae:
88.3810
1/1 [==============================] - 0s 78ms/step - loss: 37.9061 - mae:
37.9061
1/1 [==============================] - 0s 77ms/step - loss: 56.9635 - mae:
56.9635
1/1 [==============================] - 0s 86ms/step - loss: 34.3143 - mae:
34.3143
1/1 [==============================] - 0s 82ms/step - loss: 50.0919 - mae:
50.0919
1/1 [==============================] - 0s 84ms/step - loss: 22.3308 - mae:
22.3308
1/1 [==============================] - 0s 83ms/step - loss: 19.5073 - mae:
19.5073
1/1 [==============================] - 0s 86ms/step - loss: 46.1036 - mae:
46.1036
1/1 [==============================] - 0s 80ms/step - loss: 34.9211 - mae:
```

```
34.9211
1/1 [==============================] - 0s 79ms/step - loss: 67.1400 - mae:
67.1400
1/1 [==============================] - 0s 95ms/step - loss: 88.2803 - mae:
88.2803
1/1 [==============================] - 0s 76ms/step - loss: 38.1356 - mae:
38.1356
1/1 [==============================] - 0s 82ms/step - loss: 56.5062 - mae:
56.5062
1/1 [==============================] - 0s 88ms/step - loss: 34.7887 - mae:
34.7887
1/1 [==============================] - 0s 80ms/step - loss: 50.1889 - mae:
50.1889
1/1 [==============================] - 0s 83ms/step - loss: 23.7011 - mae:
23.7011
1/1 [==============================] - 0s 76ms/step - loss: 18.8806 - mae:
18.8806
1/1 [==============================] - 0s 90ms/step - loss: 46.2989 - mae:
46.2989
1/1 [==============================] - 0s 77ms/step - loss: 35.0405 - mae:
35.0405
1/1 [==============================] - 0s 79ms/step - loss: 66.7967 - mae:
66.7967
1/1 [==============================] - 0s 85ms/step - loss: 88.1821 - mae:
88.1821
1/1 [==============================] - 0s 82ms/step - loss: 38.7617 - mae:
38.7617
1/1 [==============================] - 0s 80ms/step - loss: 56.6420 - mae:
56.6420
1/1 [==============================] - 0s 78ms/step - loss: 34.9426 - mae:
34.9426
1/1 [==============================] - 0s 77ms/step - loss: 49.5608 - mae:
49.5608
1/1 [==============================] - 0s 82ms/step - loss: 22.1249 - mae:
22.1249
1/1 [==============================] - 0s 80ms/step - loss: 19.1402 - mae:
19.1402
1/1 [==============================] - 0s 80ms/step - loss: 46.3646 - mae:
46.3646
1/1 [==============================] - 0s 80ms/step - loss: 34.9986 - mae:
34.9986
1/1 [==============================] - 0s 74ms/step - loss: 66.8995 - mae:
66.8995
1/1 [==============================] - 0s 81ms/step - loss: 88.0724 - mae:
88.0724
1/1 [==============================] - 0s 80ms/step - loss: 38.7455 - mae:
38.7455
1/1 [==============================] - 0s 82ms/step - loss: 56.1730 - mae:
```

```
56.1730
1/1 [==============================] - 0s 83ms/step - loss: 35.7016 - mae:
35.7016
1/1 [==============================] - 0s 77ms/step - loss: 49.3479 - mae:
49.3479
1/1 [==============================] - 0s 77ms/step - loss: 22.2428 - mae:
22.2428
1/1 [==============================] - 1s 541ms/step - loss: 18.7199 - mae:
18.7199
1/1 [==============================] - 0s 85ms/step - loss: 46.4359 - mae:
46.4359
1/1 [==============================] - 0s 77ms/step - loss: 34.7793 - mae:
34.7793
1/1 [==============================] - 0s 75ms/step - loss: 66.9501 - mae:
66.9501
1/1 [==============================] - 0s 84ms/step - loss: 88.1812 - mae:
88.1812
1/1 [==============================] - 0s 75ms/step - loss: 38.3485 - mae:
38.3485
1/1 [==============================] - 0s 75ms/step - loss: 56.8412 - mae:
56.8412
1/1 [==============================] - 0s 78ms/step - loss: 34.3577 - mae:
34.3577
1/1 [==============================] - 0s 75ms/step - loss: 49.7002 - mae:
49.7002
1/1 [==============================] - 0s 75ms/step - loss: 22.2581 - mae:
22.2581
1/1 [==============================] - 0s 76ms/step - loss: 19.0629 - mae:
19.0629
1/1 [==============================] - 0s 75ms/step - loss: 46.0285 - mae:
46.0285
1/1 [==============================] - 0s 75ms/step - loss: 34.9062 - mae:
34.9062
1/1 [==============================] - 0s 74ms/step - loss: 66.6894 - mae:
66.6894
1/1 [==============================] - 0s 75ms/step - loss: 88.2339 - mae:
88.2339
1/1 [==============================] - 0s 80ms/step - loss: 38.3801 - mae:
38.3801
1/1 [==============================] - 0s 74ms/step - loss: 56.3906 - mae:
56.3906
1/1 [==============================] - 0s 76ms/step - loss: 35.2102 - mae:
35.2102
1/1 [==============================] - 0s 84ms/step - loss: 50.0804 - mae:
50.0804
1/1 [==============================] - 0s 77ms/step - loss: 22.4437 - mae:
22.4437
1/1 [==============================] - 0s 76ms/step - loss: 18.7385 - mae:
```

```
18.7385
1/1 [==============================] - 0s 78ms/step - loss: 46.0132 - mae:
46.0132
1/1 [==============================] - 0s 79ms/step - loss: 34.6559 - mae:
34.6559
1/1 [==============================] - 0s 79ms/step - loss: 66.3858 - mae:
66.3858
1/1 [==============================] - 0s 80ms/step - loss: 88.3430 - mae:
88.3430
1/1 [==============================] - 0s 78ms/step - loss: 38.4157 - mae:
38.4157
1/1 [==============================] - 0s 81ms/step - loss: 56.9542 - mae:
56.9542
1/1 [==============================] - 0s 85ms/step - loss: 35.3647 - mae:
35.3647
```

We get the best model and make the predictions using the test set.

[216]:
```
NN_model = gs.best_estimator_
results = NN_model.predict(scaled_test)
```

We gave the test errors for the model.

[217]:
```
print('NN MAE = ',mean_absolute_error(results,y_test))
print('NN MSE = ', mean_squared_error(results,y_test))
print('NN RMSE = ', np.sqrt(mean_squared_error(results,y_test)))
```

```
NN MAE =  38.853211799066386
NN MSE =  4975.467496051275
NN RMSE =  70.53699381212155
```

1.5 5. Evaluation

We evaluate the last 4 models with out of sample sets (test sets for our first selected client).

[218]:
```
print('ARIMA MAE = ', mean_absolute_error(arima_pred,test_ts0))
print('ARIMA MSE = ', mean_squared_error(arima_pred,test_ts0))
print('ARIMA RMSE = ', np.sqrt(mean_squared_error(arima_pred,test_ts0)))

print('NN MAE = ',mean_absolute_error(results,y_test))
print('NN MSE = ', mean_squared_error(results,y_test))
print('NN RMSE = ', np.sqrt(mean_squared_error(results,y_test)))

print('Prophet MAE = ',mean_absolute_error(prophet_future,test_results))
print('Prophet MSE = ', mean_squared_error(prophet_future,test_results))
print('Prophet RMSE = ', np.
 ↪sqrt(mean_squared_error(prophet_future,test_results)))
```

```
print('XGBOOST MAE = ',mean_absolute_error(XGBOOST_prediction,y_test))
print('XGBOOST MSE = ', mean_squared_error(XGBOOST_prediction,y_test))
print('XGBOOST RMSE = ', np.sqrt(mean_squared_error(XGBOOST_prediction,y_test)))
```

```
ARIMA MAE =  41.199633403080526
ARIMA MSE =  4206.301874970926
ARIMA RMSE =  64.856008780767
NN MAE =  38.853211799066386
NN MSE =  4975.467496051275
NN RMSE =  70.53699381212155
Prophet MAE =  38.14150563463835
Prophet MSE =  3961.0575290471243
Prophet RMSE =  62.9369329491605
XGBOOST MAE =  31.367442750502175
XGBOOST MSE =  3659.5421719196574
XGBOOST RMSE =  60.494149898313786
```

According to the test regression metrics the best model to predict the amount a client will spend, taking into account the year 2014, is the XGBOOST with MAE = 31.367442750502175, MSE = 3659.5421719196574 and RMSE = 60.494149898313786.

1.6 Conclusions

We answered the following questions:

- Which are the credit card customers' profiles for the year 2014?
- Which are the all around purchase card transactions' profiles for the year 2014?
- How much would a client spend in the next period taking into account the year 2014?

For the first one we profiled the clients by clustering the dataset using three different techniques, K-means, Hierarchical Clustering and DBSCAN.

For the second one, we profiled all the transactions by clustering the dataset with k-modes, since the dataset had mainly categorical features.

We used ARIMA, Prophet, XGBoost and Neural Networks to answer the last question. In this case, we found out that for the data we used XGBoost presented the best results.