Machine Learning Engineer Nanodegree

Capstone Project

Ibrahim El Gohary

March 9th, 2020

I. Definition

Project Overview

In this project datasets from Arvato financials solutions company have been provided. The main aim of the project is to find what does the customers of the company have in common with the general population in Germany, since Arvato is a German company. To answer this question unsupersived learning models were applied to define the key features for the cluster of the population that are similar in features to the current customers of the company. The second part of the project is occupied with creating a supervised model that is able to predict based on provided features whether a person is possible customer if the financial solution of Arvato was presented to those people.

For the first part of the project to be able to study the main features of the general population group and the current customers for Arvato two datasets were provided and used. A dataset for the general population and a dataset for the current Arvato customers. The same set of features is present in both datasets.

In the second part the datasets provided are: A dataset for a previous group of people that were contact using a mailout campaign. Some of them became customers, others didn't reply or were not interested to use the company services. This was described in this dataset by a feature stating a response: a customer or not. The second dataset for this section is another group of people and their features and a prediction has to be made using the created machine learning model whether a person in this group of a bailout campaign will be a possible customer or not.

Problem Statement

The problem this project is trying to solve is instead of having to direct bailout campaign to a whole countries population to find new possible customers, which would drain the companies resources financially and would be sort of a tedious process, to be able to be targeted to highly probable people that may become customers for the company. So the people that the machine learning model predicts as likely people to become customers will be the first priority for the company to reach out.

Metrics

The f1 score was used to evaluate the supervised learning model score.

II. Analysis

Data Exploration

Datasets provided:

1. A dataset for the general population of Germany with a number of features
2. Dataset for the company's current customers
3. Training datasets with labels whether a person became a customers or not from a layout campaign
4. Test set for a bailout campaign which has to be used by the supervised learning model to predict whether a person is a high likely future customer or not

First of all working on each dataset a sample of it is pretend using the `head()` method. Then the shape of the data consisting of how many rows and columns using `shape` method. Finally a statistical description is provided using the `describe()` method, showing the total count of entries for every feature, mean of every column, standard deviation.

Then the `info()` method was used to give certain characteristics of the dataset, like how many datapoint for every datatype, number of datapoint per dataset.

Missing data were found, also outliers, some feature data range was large so data preprocessing had to take place. Also categorical features had to be converted into numerical values.

Exploratory Visualization

<pd.Dataframe>.head()

```
# Be sure to add in a lot more cells (both markdown and code) to document your
# approach and findings!

customers.head()
```

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKA
0	9626	2	1.0	10.0	NaN	NaN	NaN	NaN	10.0
1	9628	-1	9.0	11.0	NaN	NaN	NaN	NaN	NaN
2	143872	-1	1.0	6.0	NaN	NaN	NaN	NaN	0.0
3	143873	1	1.0	8.0	NaN	NaN	NaN	NaN	8.0
4	143874	-1	1.0	20.0	NaN	NaN	NaN	NaN	14.0

5 rows × 369 columns

<pd.DataFrame>.describe()

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

Out[13]:

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIN
count	5.000000e+04	50000.000000	45779.000000	45779.000000	4506.000000	1606.000000	349.000000	74.000000
mean	6.347478e+05	-0.354220	4.417091	10.832696	11.774745	13.555417	14.719198	15.445946
std	2.586020e+05	1.204067	3.641530	7.642482	4.051729	3.189602	2.497402	2.202999
min	1.916720e+05	-1.000000	1.000000	0.000000	2.000000	3.000000	7.000000	9.000000
25%	4.102668e+05	-1.000000	1.000000	0.000000	8.000000	11.000000	13.000000	14.000000
50%	6.337150e+05	-1.000000	3.000000	13.000000	12.000000	14.000000	15.000000	16.000000
75%	8.592248e+05	-1.000000	9.000000	17.000000	15.000000	16.000000	17.000000	17.000000
max	1.082872e+06	3.000000	9.000000	21.000000	18.000000	18.000000	18.000000	18.000000

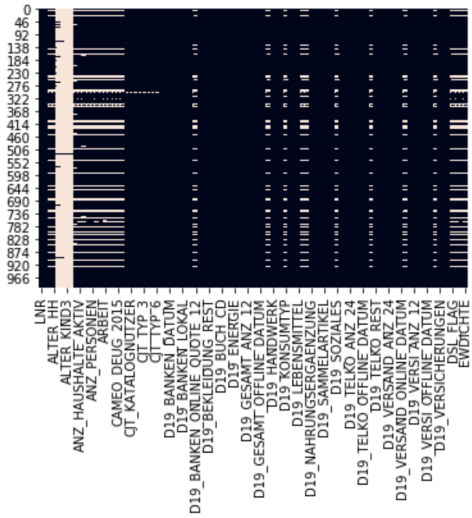
8 rows × 360 columns

<pd.DataFrame>.info()

```
: df_customers.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 50000 entries, 162743 to 158643
Columns: 369 entries, LNR to ALTERSKATEGORIE_GROB
dtypes: float64(267), int64(94), object(8)
memory usage: 142.4+ MB
```

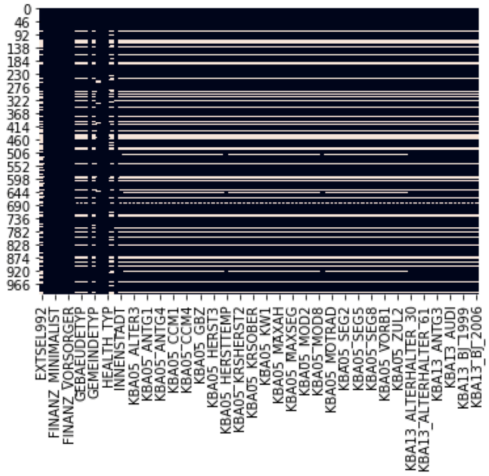
A heat map for the missing for the first columns and 1000 rows in one of the datasets:

```
[90]: sns.heatmap(df_mailout_train.iloc[:1000,:100].isnull(), cbar=False)
t[90]: <matplotlib.axes._subplots.AxesSubplot at 0x7f17debc2400>
```



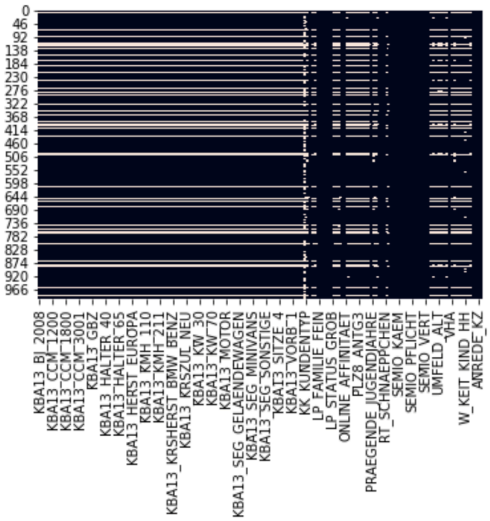
Missing values in columns 100 to 200

```
91]: sns.heatmap(df_mailout_train.iloc[:1000,100:200].isnull(), cbar=False)
91]: <matplotlib.axes._subplots.AxesSubplot at 0x7f17dea3d2e8>
```



Missing values in columns 200 to 300

```
[12]: sns.heatmap(df_mailout_train.iloc[:1000,200:].isnull(), cbar=False)
[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7f17dcf93828>
```



The colored dots or lines indicate a missing value and the black dots or areas indicate the the value is present or not null.

Algorithms and techniques

The techniques used in the project for the features' dimensionality reduction is the PCA principal component analysis to provide a smaller number of features that still cover 60% of the variance of the data to be able to use the data with supervised and unsupervised models.

For the Unsupervised part k-Means algorithm is being used to create clusters for the general population and the Arvato company customers and to be able to compare clusters of high counts of datapoint of both datasets.

In the supervised learning part an ensemble method Algorithm was used. The AdaBoost classifier which also included the DecisionTree classifier as a base estimator.

To be able to improve the quality of the Adaboost Classifier efficiency a grid search was applied to find better parameters for the classifier. Also class weight the

classifier was inserted so that to give more weight distribution and higher priority for the class predicting whether a person or a datapoint is a future possible customer.

Benchmark

The model that will be applied in this project relies on customer segmentation model to be able to target the suitable, prospective customers who would use the financial services provided by Arvato Finsncial solutions.

There are several benefits of implementing customer segmentation including informing marketing strategy, promotional strategy, product development, budget management, and delivering relevant content to your customers or prospective customers.

Example for customer segmentation model paper:

"Customer Segmentation Using Unsupervised Learning on Daily Energy Load Profiles" by J. du Toit, R. Davimes, A. Mohamed, K. Patel, and J. M. Nye http://www.-jait.us/uploadfile/2016/0505/20160505105403530.pdf

The cluster results were obtained using k-means

applied to the PCA scores with the Euclidean distance

metric.

Evaluation metrics used in this benchmark model is the the Silhouette index and Davies Bouldin metrics.

III. Methodology

Data Preprocessing

Handling missing data:

Missing data handling for integer or float values were replaced by the median value of the respective column.

First the feature columns of the integer of float column data types were stored in a variable using python list comprehension:

```
customers_int_float_columns = [key for key in
dict(df_customers_droppedna.dtypes) if
dict(df_customers_droppedna.dtypes)[key] in ['int64', 'float64']]
```

Then a method was created to replace the missing values with the mean of the feature column values:

```
# method to replace with median
def replace_Nan_with_median (df, column_list):

    for i in column_list:
        #print('i:::',column_name)
        median = df[i].median()
        df[i] =  df[i].fillna(median)
        #return new_df[column_name]
    print(df)
    return df
```

To this function the dataset and the variable for the integer and float data features names are passed to this method.

Handling categorical data

The for the categorical missing data had to be replaced with the column most frequent datapoint or value, the mean of the column.

```
customers_categorical_columns = [key for key in
dict(df_customers_droppedna.dtypes) if
dict(df_customers_droppedna.dtypes)[key] in ['object']]

for i in customers_categorical_columns:
    df_customers[i].fillna(df_customers[i].mode()[0],
inplace=True)
```

To convert the categorical values into integers the pandas method `get_dummies()` was used.

```
# create dummies variables for categorical variables
for i in customers_categorical_columns:
    df_customers = pd.get_dummies(df_customers, columns=[i])
```

The next section is removing outliers using the IQR interquartile range method.

The following finds the interquartile of the data the middle 50%.

```
Q1 = df_customers.quantile(0.25)
Q3 = df_customers.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

Next The the outliers for each datapoint in the dataset is shown as either false “not an outlier” pr true “an outlier”:

```
# outliers are the true values
print(df_customers < (Q1 - 1.5 * IQR)) |(df_customers > (Q3 + 1.5
* IQR))
```

Then we remove all outliers by replacing the outliers with nan values:

```
 # df_out is a df with outliers removed by including all values
that are "not" outliers
```

```
df_customers_out = df_customers[~(df_customers < (Q1 - 1.5 *
IQR)) |(df_customers > (Q3 + 1.5 * IQR))]
```

Then as described before, repeat the nan values with median replacement.

```
# replace Nan values with median
customers_int_float_columns = [key for key in
dict(df_customers_out.dtypes) if dict(df_customers_out.dtypes)
[key] in ['int64', 'float64', 'uint8']]


replace_Nan_with_median(df_customers_out,
customers_int_float_columns)
```

To make sure that no null values are still present:

```
df_customers_out.isnull().sum()
```

Feature scaling

For more efficient models we need the ranges for features not to be too broad or to small so we let the ranges be between 0 and 1. `MinMaxScaler()` class from sklearn was used.

```
from sklearn import preprocessing


x = df_customers.values # returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()

df_scaled = pd.DataFrame(min_max_scaler.fit_transform(x),
columns=df_customers.columns, index=df_customers.index)

# check: print unique values for a number of columns after
feature scaling
for i in range(1773,1774):
    u = pd.unique(df_scaled.iloc[i])
print (u)

df_scaled.iloc[:,90:99]
```

```
df_customers = df_scaled
```

```
[ 0.51998852  0.          0.16666667  0.25        0.00191205  0.14285714
  0.00266667  0.42857143  0.66666667  0.4         0.75        1.
  0.85714286  0.42857143  0.16666667  0.33333333  0.44444444  0.6         0.125
  0.2         0.4375      0.36363636  0.5         1.          1.
  0.28205128  0.63636364  0.22222222  0.28571429  0.11111111  0.375       0.8
  0.55555556  0.33333333  0.1503268   0.4         0.66666667  0.72727273
  0.7         0.61538462  0.46666667  0.57142857  0.83333333  0.1         0.625     ]
```

Notice for an example example feature all values range between 0 and 1.

Implementation

For all datasets a pca principal component analysis was carried out to create reduced dimensions of the datasets. The covered variance by the principal components had to be at least covering 60% of the variance of the data to be explanatory for the data. So I found the number of principal components to be used were 57 components.

```
from sklearn.decomposition import PCA
# import visuals as vs

# Use 57 principal component to cover 60% of the data variance
pca = PCA(n_components = 57)

pca.fit(df_customers)

print(pca.explained_variance_ratio_)

print("Total variance covered:
",pca.explained_variance_ratio_.sum())

customers_reduced_data = pca.transform(df_customers)



# in range function does not include the finishing element 58, so
it creates 57 columns
```

```
col_list = ['Dimension' + str(x) for x in range(1,58)]
col_list

customers_reduced_data = pd.DataFrame(data =
customers_reduced_data, columns = col_list)

customers_reduced_data
```

For the unsupervised learning part an addition had to be made to determine the most important features for each principal component for the general population dataset, also for the customers dataset.

```
from sklearn.decomposition import PCA

train_features = df_customers

model = PCA(n_components= 57).fit(train_features)
X_pc = model.transform(train_features)



# number of components
n_pcs= model.components_.shape[0]


# get the index of the most important feature on EACH component
# LIST COMPREHENSION HERE
most_important = [np.abs(model.components_[i]).argmax() for i in
range(n_pcs)]



initial_feature_names =df_customers.columns


# get the names
most_important_names = [initial_feature_names[most_important[i]]
for i in range(n_pcs)]


# LIST COMPREHENSION HERE AGAIN
```

```
dic = {'PC customers {}'.format(i+1): most_important_names[i] for
i in range(n_pcs)}

dic
```

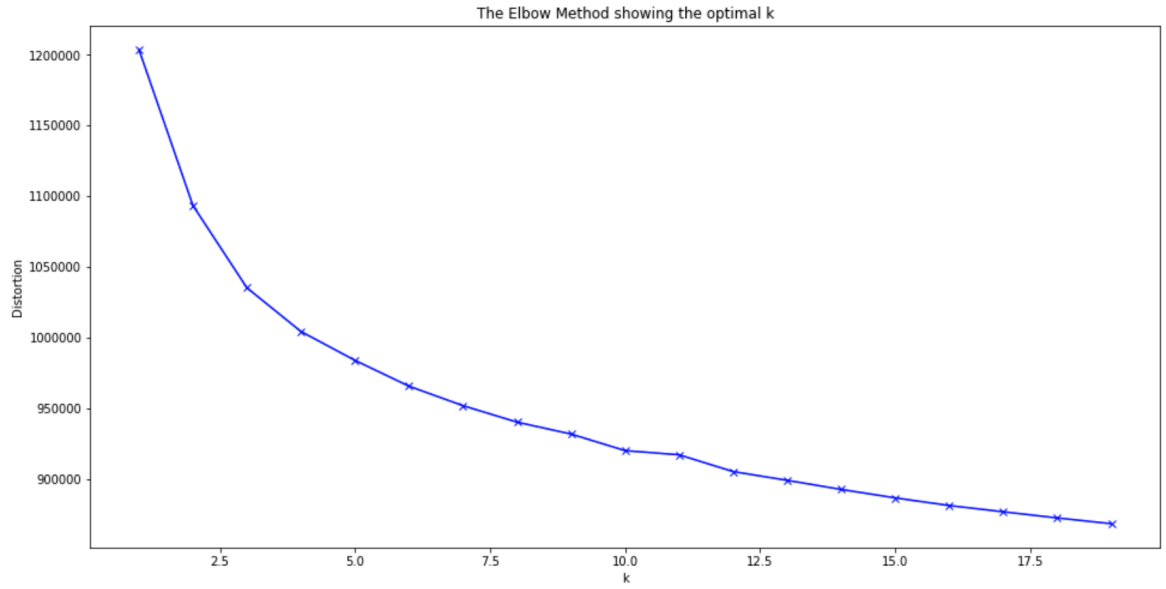
```
{'PC customers 1': 'D19_GESAMT_ONLINE_QUOTE_12',
 'PC customers 2': 'D19_GESAMT_ONLINE_QUOTE_12',
 'PC customers 3': 'GREEN_AVANTGARDE',
 'PC customers 4': 'ANREDE_KZ',
 'PC customers 5': 'ANREDE_KZ',
 'PC customers 6': 'PRODUCT_GROUP_COSMETIC_AND_FOOD',
 'PC customers 7': 'GREEN_AVANTGARDE',
 'PC customers 8': 'KBA13_KMH_140',
 'PC customers 9': 'D19_GESAMT_ONLINE_QUOTE_12',
 'PC customers 10': 'EINGEFUEGT_AM_1992-02-10 00:00:00',
 'PC customers 11': 'KBA13_ALTERHALTER_61',
 'PC customers 12': 'LP_FAMILIE_FEIN',
 'PC customers 13': 'VERS_TYP',
 'PC customers 14': 'VERS_TYP',
 'PC customers 15': 'VERS_TYP',
 'PC customers 16': 'VERS_TYP',
 'PC customers 17': 'PLZ8_HHZ',
 'PC customers 18': 'CAMEO_DEUG_2015_2',
 'PC customers 19': 'D19_LOTTO',
 'PC customers 20': 'KBA13_KRSSEG_VAN',
 'PC customers 21': 'PRODUCT_GROUP_COSMETIC',
 'PC customers 22': 'KBA13_KRSSEG_VAN',
 'PC customers 23': 'KBA13_KMH_0_140',
 'PC customers 24': 'PRODUCT_GROUP_COSMETIC',
```

Unsupervised learning model, k-Means

First I needed to know the reasonable number for clusters to be used, for this I used the elbow method which shows that 5 clusters is an efficient number for clusters to be created.

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.cluster import KMeans

#we are using
```

```
distortions = []
K = range(1,20)
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(df_final)
    distortions.append(kmeanModel.inertia_)

plt.figure(figsize=(16,8))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```



Then applying the k-Means model: setting the clusters to 5, fit the concatenated dataset of the general population and customers. I created a column that tells whether this datapoint originates from the customers dataset or the general population dataset.

```
# so the best number of clusters is 5
kmeanModel = KMeans(n_clusters=5)
kmeanModel.fit(df_final)
kmeanModel.predict(df_final)
```

```
kmeanModel.labels_.shape
df_final.shape

df_final["k_means"]= pd.Series(kmeanModel.labels_)

df_final
```

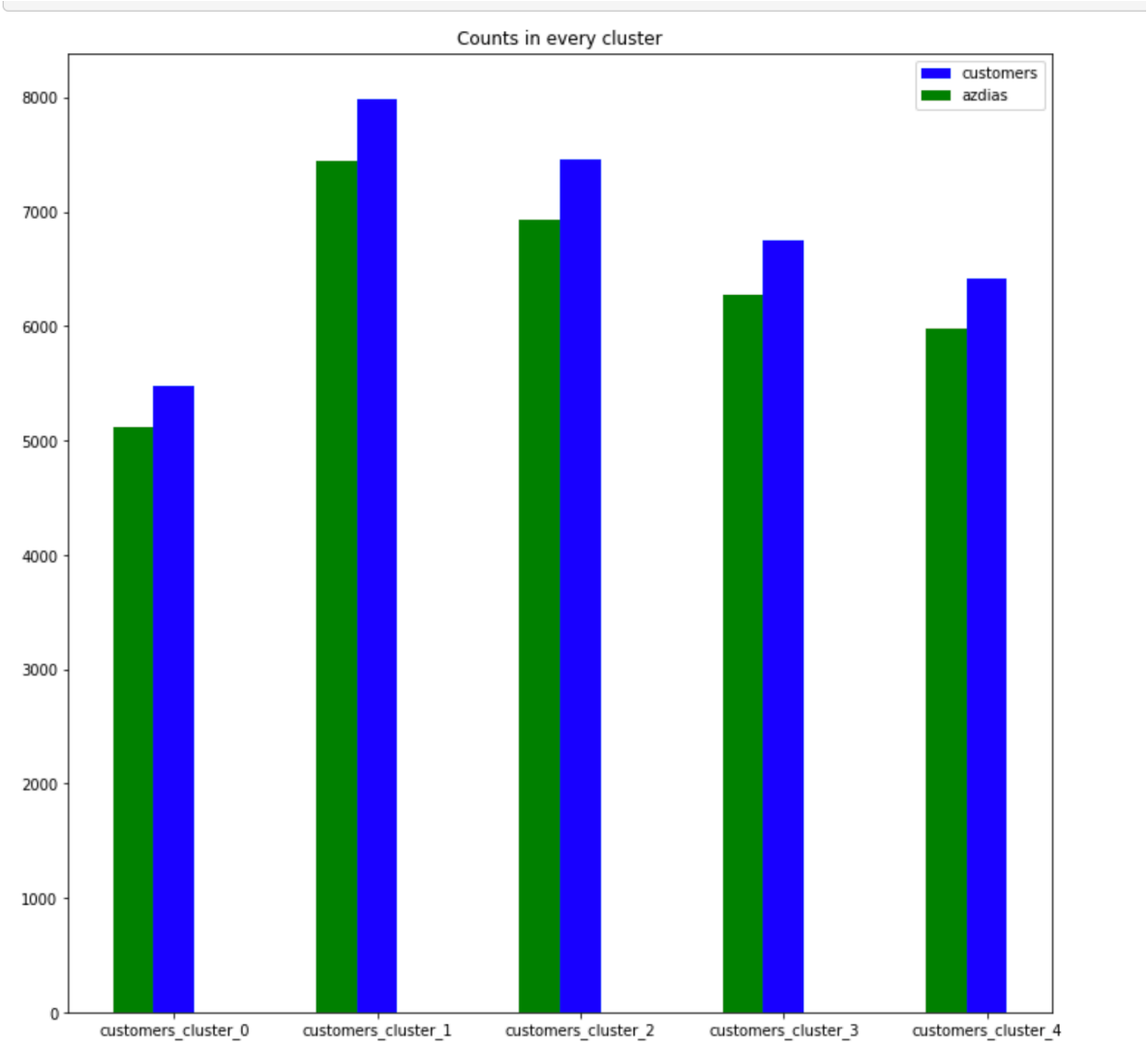
Then I calculated the number of datapoint in each cluster separately for the general population and the customers dataset:

```
customers_in_clusters = {}
for i in range(0,5):
    customers_in_clusters["customers_cluster_"+ str(i)] =
len(df_final[(df_final['customer_belong_column'] == 1) &
(df_final['k_means'] == i)])

print("customers_dict:",customers_in_clusters)

azdias_in_clusters = {}
for i in range(0,5):
    azdias_in_clusters["customers_cluster_"+ str(i)] =
len(df_final[(df_final['customer_belong_column'] == 0) &
(df_final['k_means'] == i)])

print("azdias_dict", azdias_in_clusters)

customers_in_clusters
```

```
customers_dict: {'customers_cluster_0': 5478, 'customers_cluster_1': 7990, 'customers_cluste
r_2': 7457, 'customers_cluster_3': 6757, 'customers_cluster_4': 6417}
azdias_dict {'customers_cluster_0': 5117, 'customers_cluster_1': 7443, 'customers_cluster_
2': 6937, 'customers_cluster_3': 6277, 'customers_cluster_4': 5981}
: {'customers_cluster_0': 5478,
 'customers_cluster_1': 7990,
 'customers_cluster_2': 7457,
 'customers_cluster_3': 6757,
 'customers_cluster_4': 6417}
```

A plot for counts of datapoint in each of the 5 clusters for both datasets: azdias and customers:

```
# plot counts of every cluster:
customers_in_clusters
azdias_in_clusters

X = np.arange(len(customers_in_clusters))
fig = plt.figure(figsize=(12,12))
ax = plt.subplot()
ax.bar(X, customers_in_clusters.values(), width=0.2, color='b',
align='center')
ax.bar(X-0.2, azdias_in_clusters.values(), width=0.2, color='g',
align='center')
ax.legend(('customers','azdias'))
plt.xticks(X, customers_in_clusters.keys())
plt.title("Counts in every cluster", fontsize=12)
plt.show()
```



The supervised learning model

I carried out pca analysis similar to that explained for the unsupervised learning model.

First I split the data into train and test sets:

```
from sklearn.model_selection import train_test_split

# Is the DataFrame produced from the PCA Analysis and y consists
of the labels, the response column
X= reduced_data
y= response_series

X_train, X_test, y_train, y_test = train_test_split(
```

```
    X, y, test_size=0.33, random_state=42)
```

Then I applied Adaboost supervised model with Gridsearch:

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier

param_grid = {'base_estimator__max_depth':[1,2],
              'base_estimator__min_samples_split':[3,4],
              'base_estimator__class_weight': [{0:1,1:5}]}



ABC = AdaBoostClassifier(base_estimator=
DecisionTreeClassifier())

ABC.get_params()

clf = GridSearchCV(ABC, param_grid)

clf.fit(X_train, y_train)

print(clf.best_params_)

y_true, y_pred = y_test, clf.predict(X_test)
print(classification_report(y_true, y_pred))
```

```
{'base_estimator__class_weight': {0: 1, 1: 5}, 'base_estimator__max_depth': 1, 'base_estimat
or__min_samples_split': 3}
             precision    recall  f1-score   support

        0.0       0.99      1.00      0.99     10741
        1.0       0.00      0.00      0.00       123

avg / total       0.98      0.98      0.98     10864
```

The above image shows the score and the classification report.

Finally using the created model to predict using the test set:

```
# The supervised learning model predictions for the test dataset:
clf.predict(df_test_reduced_data)

clf.score(X, y)
0.98459902794653709


y_pred = clf.predict(df_test_reduced_data)

import collections, numpy

collections.Counter(y_pred)
```

Output: `Counter({0.0: 32770, 1.0: 93})`

It predicted 93 future customers from the test set.

Refinement

To refine the supervised learning model I used the gridsearch to apply different parameters before going with certain parameters:

I tried to change the `max_depth`, `min_samples_split` and applied higher weight on the class 1 which is the class for the predicted future customers:

```
om sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier

param_grid = {'base_estimator__max_depth':[1,2],
              'base_estimator__min_samples_split':[3,4],
              'base_estimator__class_weight': [{0:1,1:5}]}



ABC = AdaBoostClassifier(base_estimator=
DecisionTreeClassifier())

ABC.get_params()
```

```
clf = GridSearchCV(ABC, param_grid)

clf.fit(X_train, y_train)

print(clf.best_params_)

y_true, y_pred = y_test, clf.predict(X_test)
print(classification_report(y_true, y_pred))
```

```
{'base_estimator__class_weight': {0: 1, 1: 5},
'base_estimator__max_depth': 1,
'base_estimator__min_samples_split': 3}
             precision    recall  f1-score   support

        0.0       0.99      1.00      0.99     10741
        1.0       0.00      0.00      0.00       123

avg / total       0.98      0.98      0.98     10864
```

Results

The Unsupervised model

The cluster 1 from the k-means model application was analyzed. I calculated the correlation for the dimensions of principal components with the highest t correlation and looked for the key features of the highest correlation.

Dimension3	Dimension4	0.079045
Dimension4	Dimension3	0.079045
Dimension3	Dimension11	0.080278
Dimension11	Dimension3	0.080278
Dimension37	Dimension36	0.082198
Dimension36	Dimension37	0.082198
Dimension3	Dimension15	0.087445
Dimension15	Dimension3	0.087445
Dimension1	Dimension25	0.103319
Dimension25	Dimension1	0.103319
Dimension3	Dimension2	0.112423
Dimension2	Dimension3	0.112423
Dimension55	Dimension57	0.116903
Dimension57	Dimension55	0.116903
Dimension1	Dimension1	1.000000
Dimension29	Dimension29	1.000000

Highestcorrelations for pca dimensions are:

Dimension 55, 57, 3, 2

In common they have:

'PC customers 2': 'D19*GESAMT*ONLINE*QUOTE*12',

Most important in 2nd principal component for both datasets

Azdias, general population most important features for the highest correlated principal components:

Dimension 55: SHOPPER*TYP*

Dimension 57: D19*KOSMETIK*

Dimension 3: FINANZ*ANLEGER*

Dimension 2: D19*GESAMT*ONLINE*QUOTE*12

Customer most important features for the highest correlated principal components:

Dimension 55: KK*KUNDENTYP*

Dimension 57: GEBAEUDETYP

Dimension 3: GREEN*AVANTGARDE*

Dimension 2: D19*GESAMT*ONLINE*QUOTE*12

D19*GESAMT*ONLINE*QUOTE*12 is the feature that is highly decisive of the person becomes future spending which has to do with the online spending of the person.

The Supervised model

I was evaluating the model by getting a high f1-score and a higher number of predictions from the test set, as at first I got only 4 predicted customers which needed to be improved. At last the predicted customers reached 93 from the test set predictions.

Conclusion

It is better to use machine learning models for making better decisions and being able to manage all kinds of resources better and more efficient. From the supervised model allow us to to know who of the population should be of high priority to reach to find new customers. The unsupervised model helped us to do some sort of customer segmentation and being able to study the customers of the company and the general population given the data.

I think a better solution can exist by providing more descriptive features and with more computational power that will let the computations take less time for testing the models and being able to study variations of the data handling and model creation more.