Machine Learning Engineer Nanodegree Program - Project Capstone

How I went through the project in the development stages of the project.

Phase 1: Selection of the project

Though I tried my initial analysis with an open dataset on Covid-19, I found many of the findings are repetitive and there was very less possibility for understanding clusters using unsupervised learnings.

Next I had to choose between the projects given by Udaciyt where I found the data of Arvato Financial services very interesting. It was also because I had a previous experience working with a company to understand customer segmentation.

Phase 2: Selection of Dataset

There following were the Datasets provided

1. DIAS Attributes - Values 2017.xlsx - The metadata
2. DIAS Information Levels - Attributes 2017.xlsx - Understanding the levels in the data
3. Udacity_AZDIAS_052018.csv: Demographics data for the general population of Germany; 891 211 persons (rows) x 366 features (columns).
4. Udacity_CUSTOMERS_052018.csv: Demographics data for customers of a mail-order company; 191 652 persons (rows) x 369 features (columns).
5. Udacity_MAILOUT_052018_TRAIN.csv: Demographics data for individuals who were targets of a marketing campaign; 42 982 persons (rows) x 367 (columns).
6. Udacity_MAILOUT_052018_TEST.csv: Demographics data for individuals who were targets of a marketing campaign; 42 833 persons (rows) x 366 (columns).

In this we are to perform Supervised learning using datasets 3 and 4 whereas with the same processing techniques apply unsupervised learning techniques using 5 and 6.

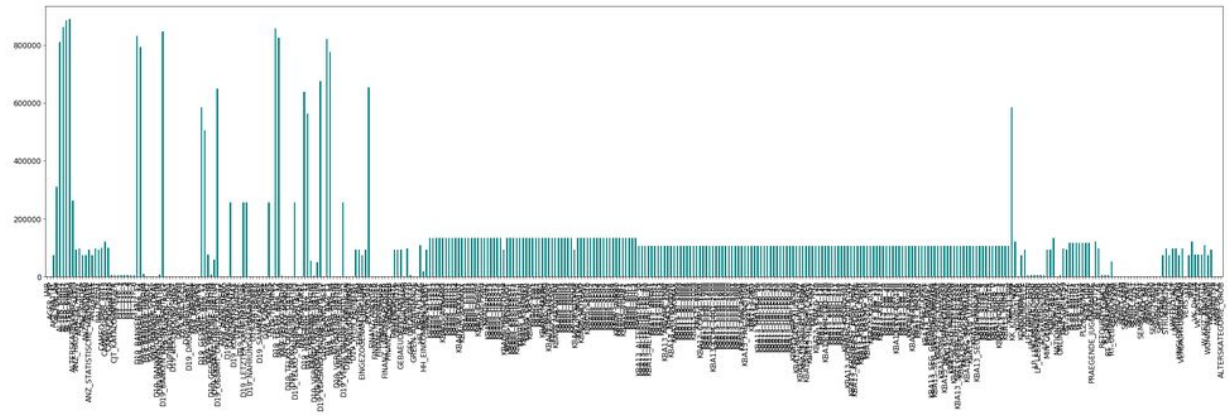
Phase 3: Understanding the data

Upon loading the data for General data with demographics of people of Germany and the customer data of Arivato financial services we can understand the presence of NULL values in many of the columns.

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_A
0	910215	-1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	910220	-1	9.0	0.0	NaN	NaN	NaN	NaN	21.0	
2	910225	-1	9.0	17.0	NaN	NaN	NaN	NaN	17.0	
3	910226	2	1.0	13.0	NaN	NaN	NaN	NaN	13.0	
4	910241	-1	1.0	20.0	NaN	NaN	NaN	NaN	14.0	

The image with head functions

Converting this into a visualization using Matplotlib library we get the following.



Plot showing the NULL/NaN values in the data

The NULL values can be classified into two categories.

- With columns more than 80% of NaN values that are irrelevant data
- With columns with lesser proportion of NaN values

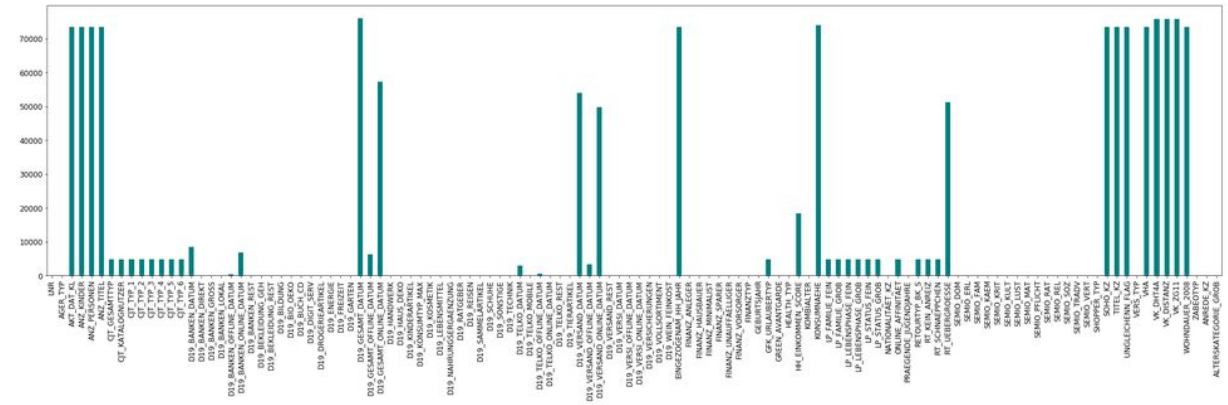
Both these categories are to be considered separately in the pre-processing steps.

Phase 4: Preprocessing of the data

Another step that calls for a pre-processing is understanding the variable types in the category variables and re-encoding them. For this manual looking is done into the ‘DIAS Information Levels - Attributes 2017.xlsx’ file and conclusions are made on the datatype for each column.

The third pre-processing done on the data is to fill with dummy variables and forward filling methods for NaN values.

After the pre-processing the data plot looks like as below

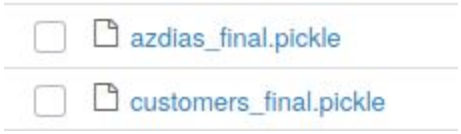


Plot made using Matplotlib - The ratio of columns with NULL values has been reduced

The same steps are applied on the ‘Customer’ data too. However, we need to handle the three extra columns that are present in this data, which classify the customer’s activity from the ‘General’ data. For this purpose the three extra columns are stored in a different dataframe and dropped from the original dataframe.

Phase 5: Utilizing the memory efficiently

On reaching till this point we find that the preprocessed data takes up a lot of memory that can affect the performance of the code and over usage of the memory of the machine. Hence we can store the data as pickle files. The below are some examples of the pickle files generated.



These files can be further unloaded using pickle commands for later use.

Phase 6: Bringing uniformity to the data

Applying feature scaling will be the next step.

Here feature scaling was done using StandardScaler. As we know there are other scalars such Quantile transforms or normal feature scaling. As we have data here with considerable missing value and where I have used the mean method to impute values standard scaling can maintain the accuracy of the data very well.

The StandardScaler from the sklearn library will be the choice to implement. And below are the general data and customer data after applying feature scaling.

```
In [55]: # View the first few lines of the azdias dataset after scaling
         azdias_clean.head()
```

Out[55]:

	LNR	AGER_TYP	AKT_DAT_KL	ANZ_KINDER	ANZ_PERSONEN	ANZ_TITEL	CJT_GESAMTTYP	CJT_KATALOGNUTZER	CJT_TYP_1	CJT_TYP_2	...
0	1.060942	-0.535207	-2.548188e-16	5.767669e-17	-6.016590e-16	1.315086e-17	-1.026509	1.117603	-1.735371	-1.570605	...
1	1.060961	-0.535207	1.313451e+00	-3.200530e-01	2.460008e-01	-6.309721e-02	0.859488	-1.567755	1.195893	1.291527	...
2	1.060981	-0.535207	1.313451e+00	-3.200530e-01	-6.572095e-01	-6.309721e-02	-0.397844	-0.896415	0.463077	0.575994	...
3	1.060984	1.967456	-9.817530e-01	-3.200530e-01	-1.560420e+00	-6.309721e-02	-1.026509	-0.225076	-1.002555	-0.855072	...
4	1.061043	-0.535207	-9.817530e-01	-3.200530e-01	2.052422e+00	-6.309721e-02	0.859488	-0.225076	-0.269739	-0.139539	...

5 rows × 118 columns

The General data we see the data to have an uniform range

```
In [56]: # View the first few lines of the customers dataset after scaling
         customers_clean.head()
```

Out[56]:

	LNR	AGER_TYP	AKT_DAT_KL	ANZ_KINDER	ANZ_PERSONEN	ANZ_TITEL	CJT_GESAMTTYP	CJT_KATALOGNUTZER	CJT_TYP_1	CJT_TYP_2	...
0	-2.439573	1.967456	-0.981753	-0.320053	0.246001	-0.063097	-2.283841	0.446263	-1.735371	-1.570605	...
1	-1.917770	-0.535207	-0.981753	-0.320053	-0.657210	-0.063097	-2.283841	1.117603	-1.002555	-0.855072	...
2	-1.917766	1.133235	-0.981753	-0.320053	-1.560420	-0.063097	-2.283841	1.117603	-1.735371	-1.570605	...
3	-1.917762	-0.535207	-0.981753	-0.320053	2.052422	-0.063097	-2.283841	0.446263	-0.269739	-0.139539	...
4	-1.917707	1.133235	-0.981753	-0.320053	0.246001	-0.063097	-2.283841	-0.225076	-1.735371	-1.570605	...

5 rows × 118 columns

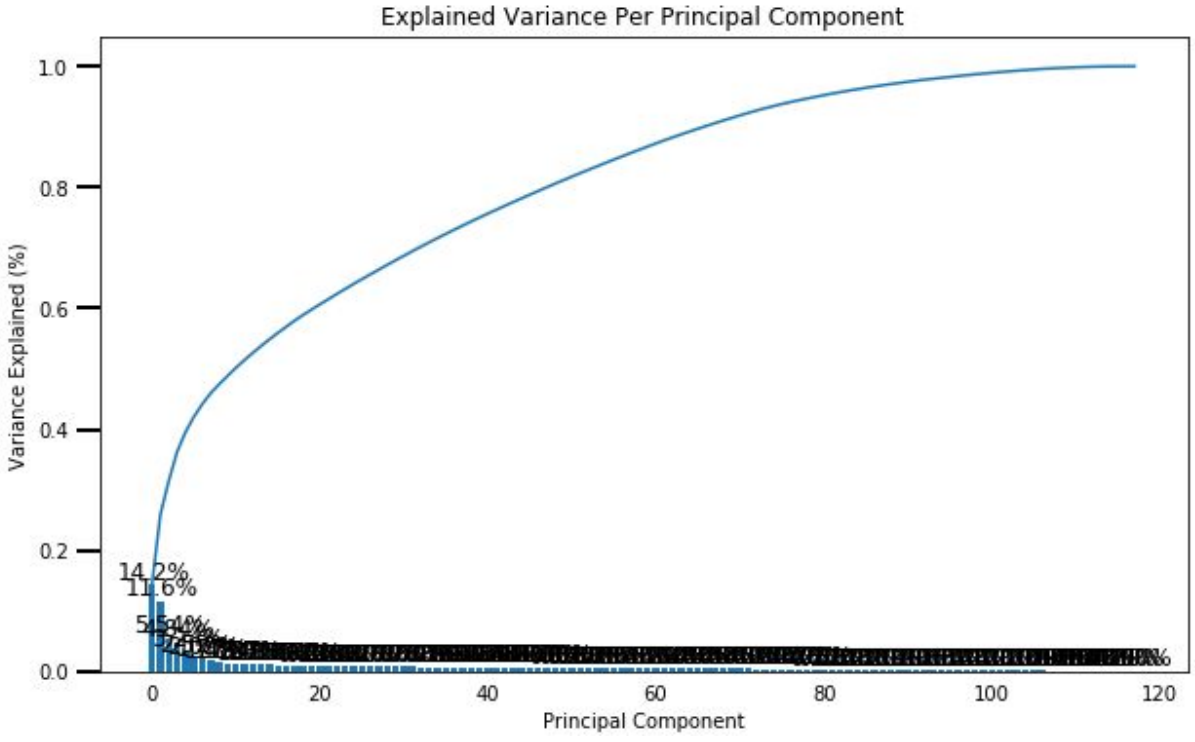
The Customer Data with uniformity brought to the values of various columns

Phase 7: Choosing the right variables using Principal Component Analysis

- Use sklearn's PCA class to apply Principle Component Analysis over the variables. We know we need to find variables that has maximum variance among themselves so that their impact is clearly seen over the predicted variables.

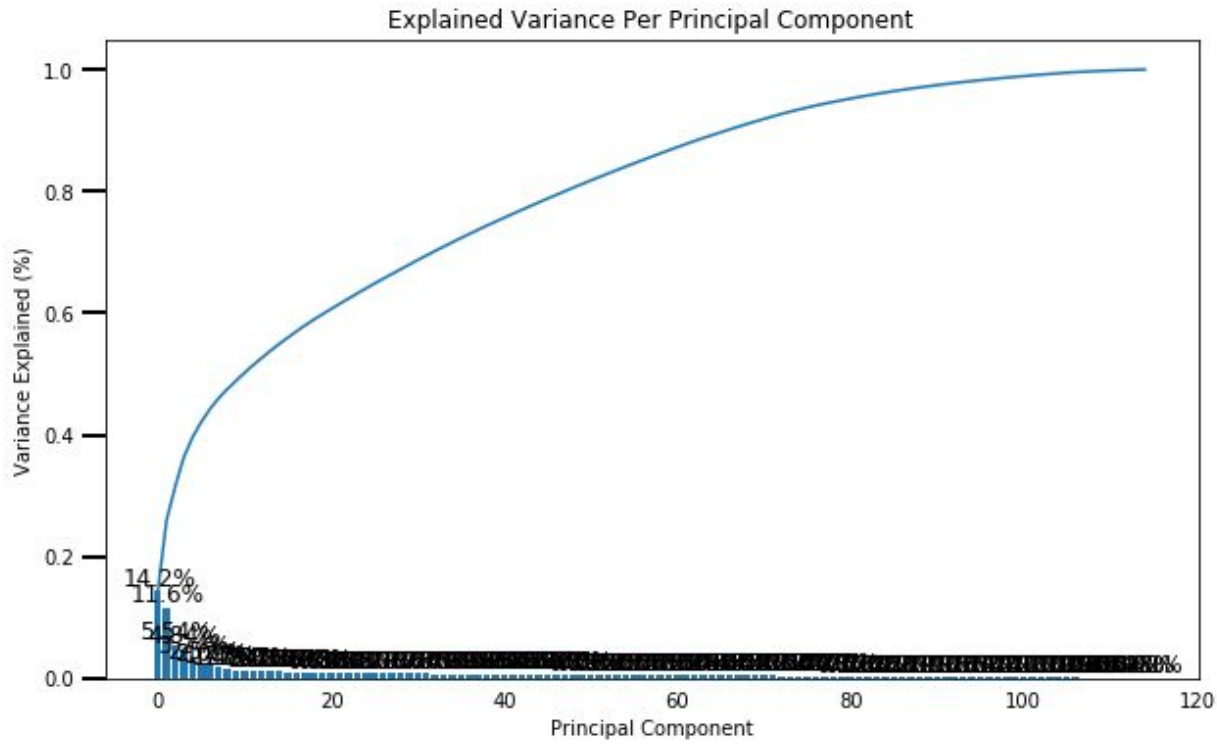
- We have to check out the ratio variance explained by each principle component as well as the cumulative variance explained.Try plotting the cumulative or sequential values using matplotlib's plot() function. Based on what you find, select a value for the number of transformed features you'll retain for the clustering part of the project.

- Once the choice for the number of components to keep is made, we can re-fit the transform to perform decide-on transformation.



Based on the graph we can choose either a value for PCA at the elbow or at 80% above variance. Hence I am choosing the value of 115.

Hence we can re-fit the data with 115 values for PCA.



The graph doesn't show much of a difference as the previous value near 120 and 115 are slightly closer.

Each principal component is a unit vector that points in the direction of highest variance (after accounting for the variance captured by earlier principal components). The further a weight is from zero, the more the principal component is in the direction of the corresponding feature. If two features have large weights of the same sign (both positive or both negative), then increases in one tend to be associated with increases in the other. To contrast, features with different signs can be expected to show a negative correlation: increases in one variable should result in a decrease in the other.

To investigate the features, you should map each weight to their corresponding feature name, then sort the features according to weight. The most interesting features for each principal component, then, will be those at the beginning and end of the sorted list. Use the data dictionary document to help you understand these most prominent features, their relationships, and what a positive or negative value on the principal component might indicate. You should investigate and interpret feature associations from the first three principal components in this

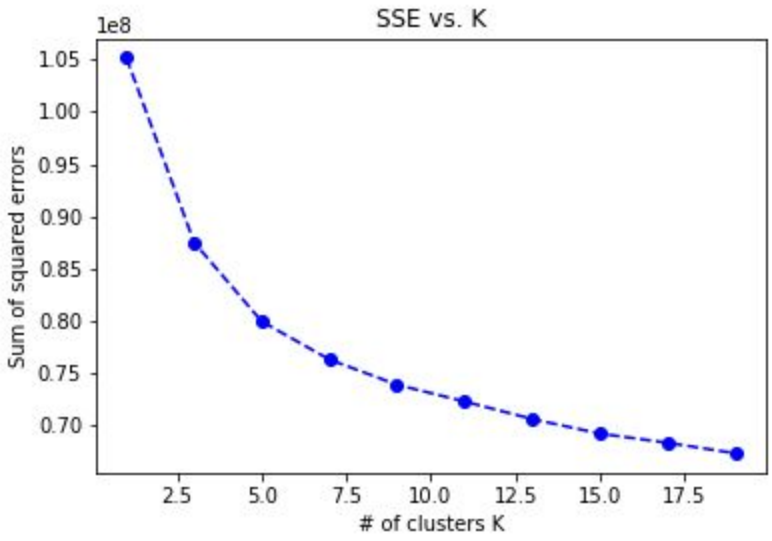
substep. To help facilitate this, you should write a function that you can call at any time to print the sorted list of feature weights, for the i-th principal component. This might come in handy in the next step of the project, when you interpret the tendencies of the discovered clusters.

Using the 115 value for PCA we can analyze each component.

Phase 8: Apply Unsupervised learning
For unsupervised learning I'm choosing K-Means clustering

- Use sklearn's KMeans class to perform k-means clustering on the PCA-transformed data.
- Then, compute the average difference from each point to its assigned cluster's center.
- Perform the above two steps for a number of different cluster counts. You can then see how the average distance decreases with an increasing number of clusters. However, each additional cluster provides a smaller net benefit. Use this fact to select a final number of clusters in which to group the data.
- Once you've selected a final number of clusters to use, re-fit a KMeans instance to perform the clustering operation

From the elbow method we can infer the K-value from the below diagram

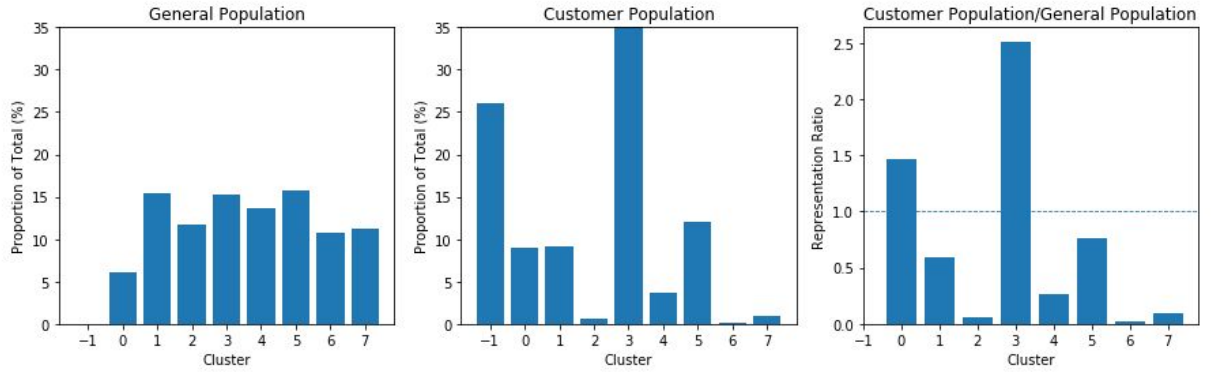


From the elbow method we can choose value 8 for the K-Means

With K-value as 8, we can fit the model and cluster the data, whose results are stored as an array.

```
array([[ -1.        ,  25.98929309],
       [  0.        ,   8.99599274],
       [  1.        ,   9.06330224],
       [  2.        ,   0.62404775],
       [  3.        ,  38.42067915],
       [  4.        ,   3.64045249],
       [  5.        ,  12.07657629],
       [  6.        ,   0.19097114],
       [  7.        ,   0.99868512]])
```

Using this array we can understand which of the clusters has maximum representation on customer data from the general data.



From the Graph we can understand the clusters, 1, 3,4, 5 and 6 are underrepresented in the Customers data while 2 and 7 are over represented.

Phase 9: Supervised Learning Model
For this we use the Training dataset and Test dataset.

First step is to impute the values as part of the pre-processing and perform all same except dropping rows for the test data cleaning as we need all the data for the test dataset to be maintained.

Next perform standard scaling.

An important element of the supervised learning is the evaluation metrics.
Here I am choosing the Receiver operating characteristic (ROC).

To select the best classifier we are using ROC as ROC is used for understanding trade -offs in Binary Classifier. ROC Curves summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds.

Since we aim for the best results we can try using different classifier algorithms and choose the best from them.

The results of each of the classifier algorithms are as in the below column.

Classifier	ROC Value	Time to run (s)
Logistic Regression	0.671	17
Random Forest Classifier	0.535	4.3
AdaBoost Classifier	0.692	27.02
Gradient Boost Classifier	0.704	71.9
Support Vector Machine	0.568	858.85

From the above table we understand the viable classifier is Gradient Boost Classifier even though it took 71.9 seconds to run on the training data, the accuracy is at highest.

Next we need to find the best hyperparameters selection for this model. From the parameters chosen by default for Gradient Boosting, I am providing a list of parameters for the following parameter grids.

```
param_grid = {'learning_rate': [0.01, 0.1, 1.0],
              'max_depth' : [2,3,4],
             'n_estimators': [20, 50, 100]}
```

GridSearch CV will choose between the best combination

For this we are using the GridSearch CV method from sklearn library.

After optimizing the model, we can fit the training data with the optimized model and predict on the test data using the same model.

For Kaggle submission we can combine the [‘LNR’] field and the response column to a CSV format as shown below.

	LNR	RESPONSE
0	1754	0.016196
1	1770	0.024638
2	1465	0.006509
3	1470	0.007304
4	1478	0.009218

A story about this project is also published on Medium and the following is the link

[Medium Link

The Github link is as below:
Github Link

Conclusion:
Through this project we went from the basic raw data to building a pipeline to preprocess the data to be able to be input to both unsupervised and supervised models. Due to the generality of data unsupervised learning could give us an overview of clusters only and not perfect prediction. However, with the train and untrained data with the more curation of data the predictions were more accurate. The prediction accuracy can be further improved by maintaining more columns that possess NaNs and understanding correlation between the variables. The actual customer data appears to be a little sparse with many of the general population's data not being represented. If further improvement is made in the data collection strategy like one to one basis media campaigns via social media or surveys we can improve the accuracy of the customer behaviour better.