Machine Learning Engineer Nanodegree

Arvato Financial Services, Customer Segmentation, Project

Dinesh Gopal
April 14th, 2020

I. Definition

Project Overview

Financial Services, Arvato is an IT company that has different products lines. We will be focusing on a use case that will help their financial group to identify potential customers through their mail order campaign.

Key Project Milestones:
• Customer Segmentation using Demographics Data
• Develop a Model to predict potential customer for the company through ad campaign
• Upload the results to Kaggle competition to share and score the performance and analysis

Problem Statement

In this project, we will analyze demographics data of customers of a mail-order sales company (Arvato) in Germany, comparing it against demographics information for the general population. We will use unsupervised learning techniques to perform customer segmentation and identifying parts of the population that best describe the core customer base of the company.

We'll then apply what we've learned on a dataset, with demographics information for targets of a marketing campaign for the company, and use a model to predict which individuals are most likely to convert into becoming customers for the company.

Metrics

We will use the real-life user data provided by Arvato for the project. Numpy and Pandas libraries will be used to perform data cleanup, EDA and pre-processing activities. PCA will be used to reduce the dimensionality of the data identifying as part of the feature engineering to identify ideal input features. K-Means algorithm will be applied on the processed data to identity different population clusters, while fine tuning the algorithm parameters.

The second part of the project involves creating a supervised model to predict potential customers for the company using the given dataset. Binary classification will be used to classify the outcome. Logistic Regression, XGBoost algorithms will be used. Confusion Matrix, Accuracy and ROC-AOC score will be used to validate the model performance.

II. Analysis

Data Exploration

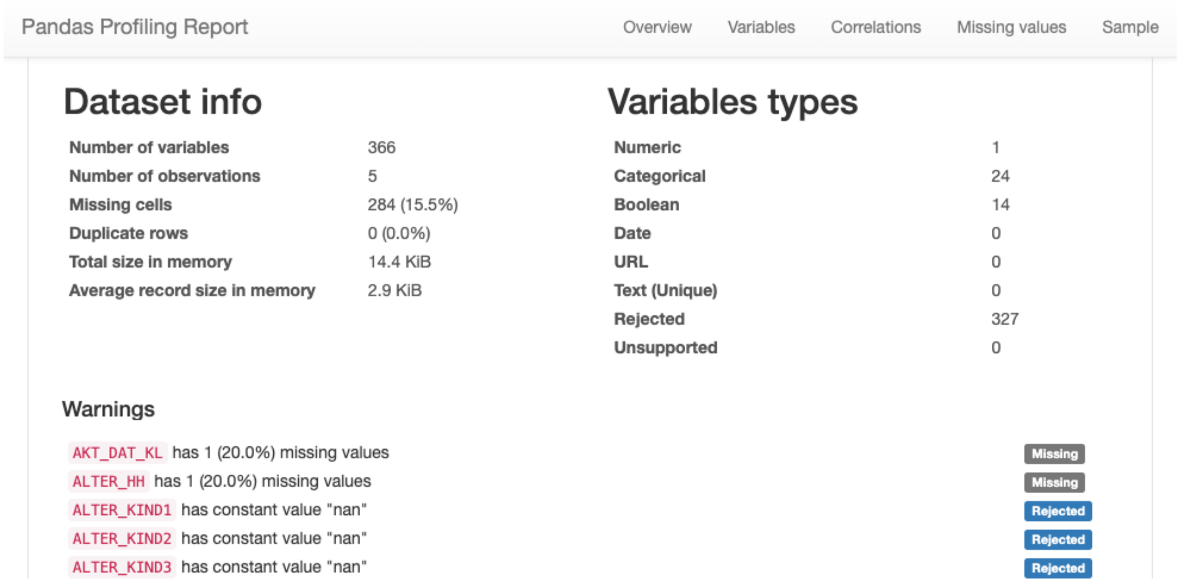
As with any data science or machine learning project, data cleanup and processing is the most crucial and time consuming aspect of the project. The data is what makes or breaks a model.

For our project, we have been give four real-life datasets of population, customer data for Germany for the mail order company and the user data of the mail order campaign.

- **AZDIAS**: Demographics data for the general population of Germany
- **CUSTOMERS**: Demographics data for customers of a mail-order company
- **MAILOUT_TRAIN** and **MAILOUT_TEST**: Demographics data for individuals, targets of a marketing campaign

First step in the project is the data analysis. We will be analyzing the data and process it which will help us identify clusters for our customer base and will eventually help us identify potential customers for the company.

Initial sneak peek at the demographics data AZDIAS which contains 891 211 persons (rows) x 366 features (columns). It has a combination of numeric and categorical data. The first part of the process is to identify all the NaNs in the dataset and handle it to make the dataset conducive for the unsupervised learning part of the project.



Dataset info

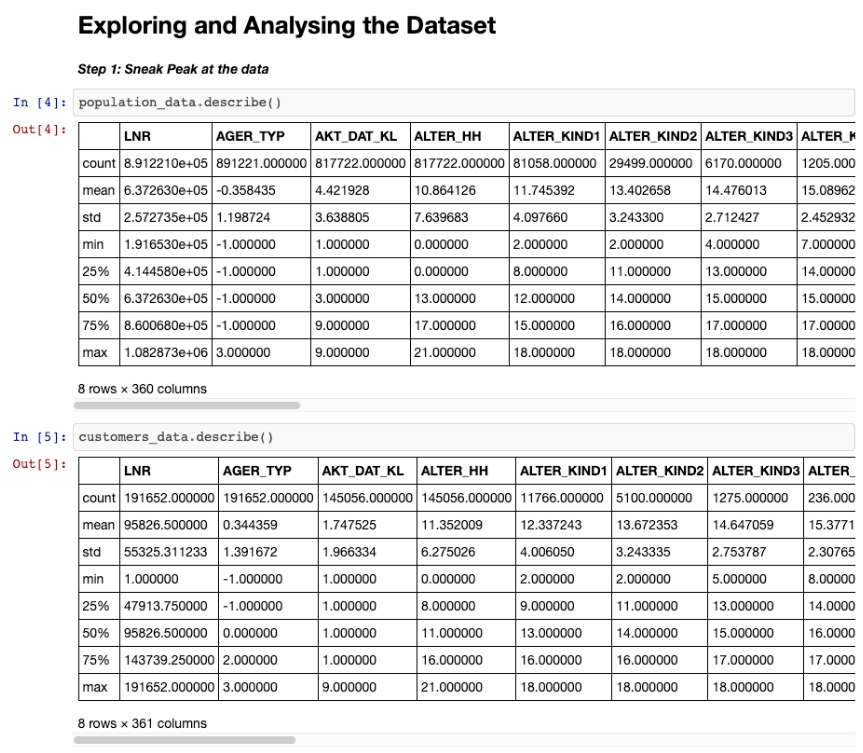
Number of variables	366
Number of observations	5
Missing cells	284 (15.5%)
Duplicate rows	0 (0.0%)
Total size in memory	14.4 KiB
Average record size in memory	2.9 KiB

Variables types

Numeric	1
Categorical	24
Boolean	14
Date	0
URL	0
Text (Unique)	0
Rejected	327
Unsupported	0

Warnings

AKT_DAT_KL has 1 (20.0%) missing values	Missing
ALTER_HH has 1 (20.0%) missing values	Missing
ALTER_KIND1 has constant value "nan"	Rejected
ALTER_KIND2 has constant value "nan"	Rejected
ALTER_KIND3 has constant value "nan"	Rejected



Exploring and Analysing the Dataset

Step 1: Sneak Peak at the data

```
In [4]: population_data.describe()
```

Out[4]:

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_K
count	8.912210e+05	891221.000000	817722.000000	817722.000000	81058.000000	29499.000000	6170.000000	1205.000
mean	6.372630e+05	-0.358435	4.421928	10.864126	11.745392	13.402658	14.476013	15.08962
std	2.572735e+05	1.198724	3.638805	7.639683	4.097660	3.243300	2.712427	2.452932
min	1.916530e+05	-1.000000	1.000000	0.000000	2.000000	2.000000	4.000000	7.000000
25%	4.144580e+05	-1.000000	1.000000	0.000000	8.000000	11.000000	13.000000	14.00000
50%	6.372630e+05	-1.000000	3.000000	13.000000	12.000000	14.000000	15.000000	15.00000
75%	8.600680e+05	-1.000000	9.000000	17.000000	15.000000	16.000000	17.000000	17.00000
max	1.082873e+06	3.000000	9.000000	21.000000	18.000000	18.000000	18.000000	18.00000

8 rows × 360 columns

```
In [5]: customers_data.describe()
```

Out[5]:

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_
count	191652.000000	191652.000000	145056.000000	145056.000000	11766.000000	5100.000000	1275.000000	236.000
mean	95826.500000	0.344359	1.747525	11.352009	12.337243	13.672353	14.647059	15.3771
std	55325.311233	1.391672	1.966334	6.275026	4.006050	3.243335	2.753787	2.30765
min	1.000000	-1.000000	1.000000	0.000000	2.000000	2.000000	5.000000	8.00000
25%	47913.750000	-1.000000	1.000000	8.000000	9.000000	11.000000	13.000000	14.0000
50%	95826.500000	0.000000	1.000000	11.000000	13.000000	14.000000	15.000000	16.0000
75%	143739.250000	2.000000	1.000000	16.000000	16.000000	16.000000	17.000000	17.0000
max	191652.000000	3.000000	9.000000	21.000000	18.000000	18.000000	18.000000	18.0000

8 rows × 361 columns

Project Design: We will follow the high-level process flow below for processing data, identifying population cluster and creating a supervised model.

High Level Process Flow:

- Collect Data for the use case
- Clean up and Pre-process the demographics (General & Customer) data
- Feature Engineering
- PCA, reduce input features to identify effective input features
- Split the dataset into test, train and validation
- Generate the shoulder curve to identify the cluster size
- Apply K-means algorithm to the processed dataset to identify clusters
- Identify potential customer base clusters by comparing general vs customer cluster
- Clean up and Pre-process the given train and test customer data
- Identify Input Features
- Create a model
- Train the model
- Test the model with test customer data
- Validate output data and model performance
- Infer insights from the final predicted output

The proposed solution will use the given dataset and potentially identify demographics of the customer base for the company and will use machine learning algorithms and other techniques to identify potential customers for the company from the mail order marketing campaign.

The following High Level Steps will be taken:

- Analyze and Clean-up data by handling NaNs, outliers and other anomalies
- Feature engineering to identify valid inputs
- Part 1: Unsupervised Learning Section
  - For customer segmentation, K-Means algorithm will be used to identify various customer base
  - Elbow and Silhouette methods will be used to identify and validate optimum clusters (K)
- Part 2: Supervised Learning Section
  - For Predicting potential customers, Logistic Regression and XGBoost algorithms will be used
- GridSearchCV APIs will be used for hyper parameter tuning
- Accuracy and ROC-AOC score metrics will be used to validate the performance of the algorithms

Exploratory Visualization

Arvato, the mail order company provided us with real life demographics and mail order camping dataset to work on the project use case. There are four data files provided for the use case.

- **Udacity_AZDIAS_052018.csv**: Demographics data for the general population of Germany; 891 211 persons (rows) x 366 features (columns)
- **Udacity_CUSTOMERS_052018.csv:** Demographics data for customers of a mail-order company; 191652 persons (rows) x 369 features (columns).
- **Udacity_MAILOUT_052018_TRAIN.csv**: Demographics data for individuals who were targets of a marketing campaign; 42 982 persons (rows) x 367 (columns).
- **Udacity_MAILOUT_052018_TEST.csv**: Demographics data for individuals who were targets of a marketing campaign; 42 833 persons (rows) x 366 (columns).

Based on the high-level analysis the dataset is not clean and has a good number of NaNs and missing values. We will use Pandas and Numpy APIs to fill in/remove NaN's by each column basis.

```
In [5]: customers_data.describe()
```

Out[5]:

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_
count	191652.000000	191652.000000	145056.000000	145056.000000	11766.000000	5100.000000	1275.000000	236.000
mean	95826.500000	0.344359	1.747525	11.352009	12.337243	13.672353	14.647059	15.3771
std	55325.311233	1.391672	1.966334	6.275026	4.006050	3.243335	2.753787	2.30765
min	1.000000	-1.000000	1.000000	0.000000	2.000000	2.000000	5.000000	8.00000
25%	47913.750000	-1.000000	1.000000	8.000000	9.000000	11.000000	13.000000	14.0000
50%	95826.500000	0.000000	1.000000	11.000000	13.000000	14.000000	15.000000	16.0000
75%	143739.250000	2.000000	1.000000	16.000000	16.000000	16.000000	17.000000	17.0000
max	191652.000000	3.000000	9.000000	21.000000	18.000000	18.000000	18.000000	18.0000

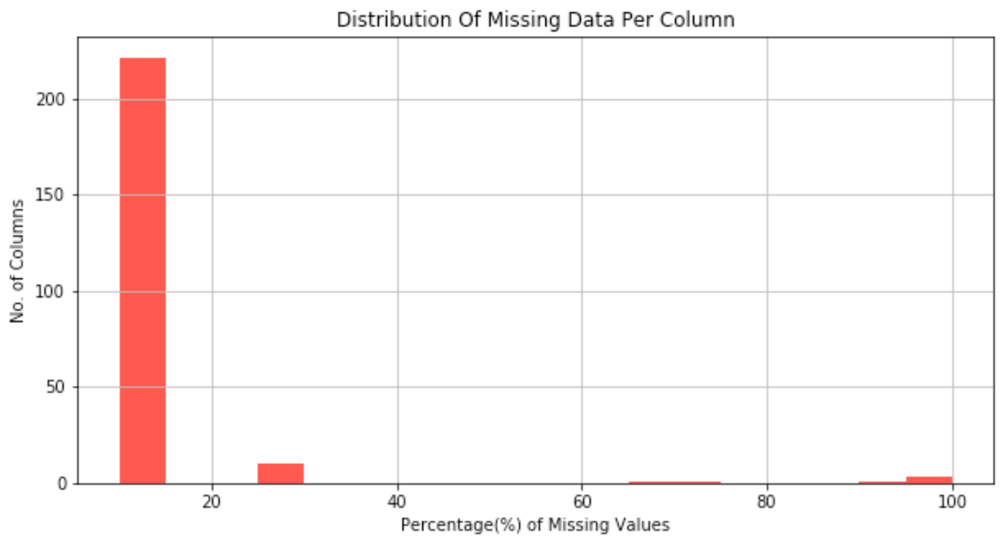
8 rows × 361 columns

Step 2: Identifying NaNs in the dataset Over Rows and Columns

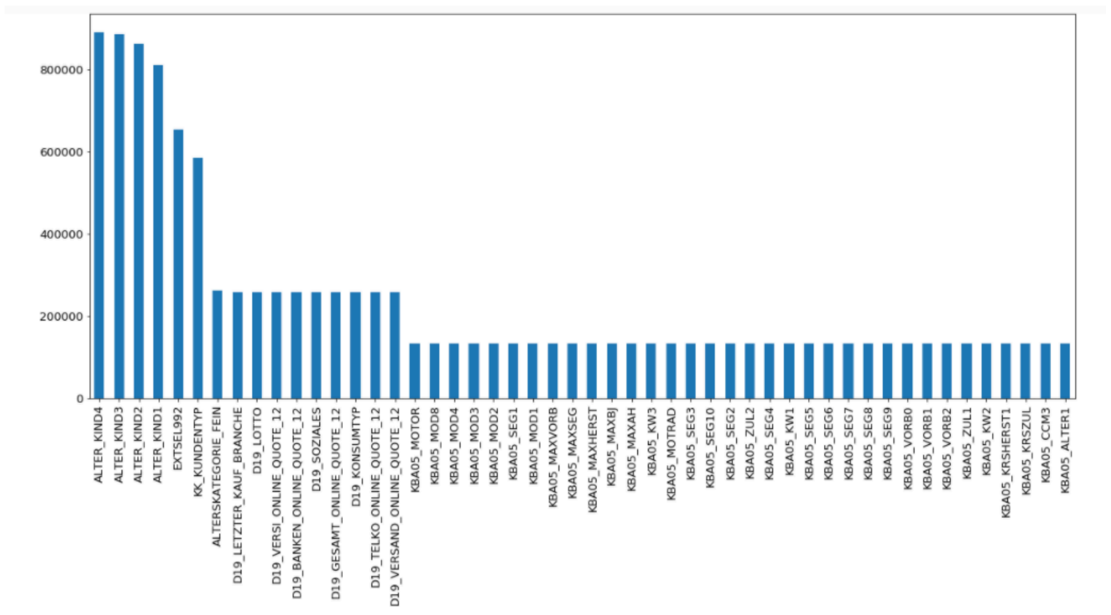
```
In [6]: #Identifying the null data (NaN) percentage in the dataset
        population_data_NaNs = population_data.isnull().sum()
        population_data_NaNs_Percent = population_data_NaNs / len(population_data) * 100
```

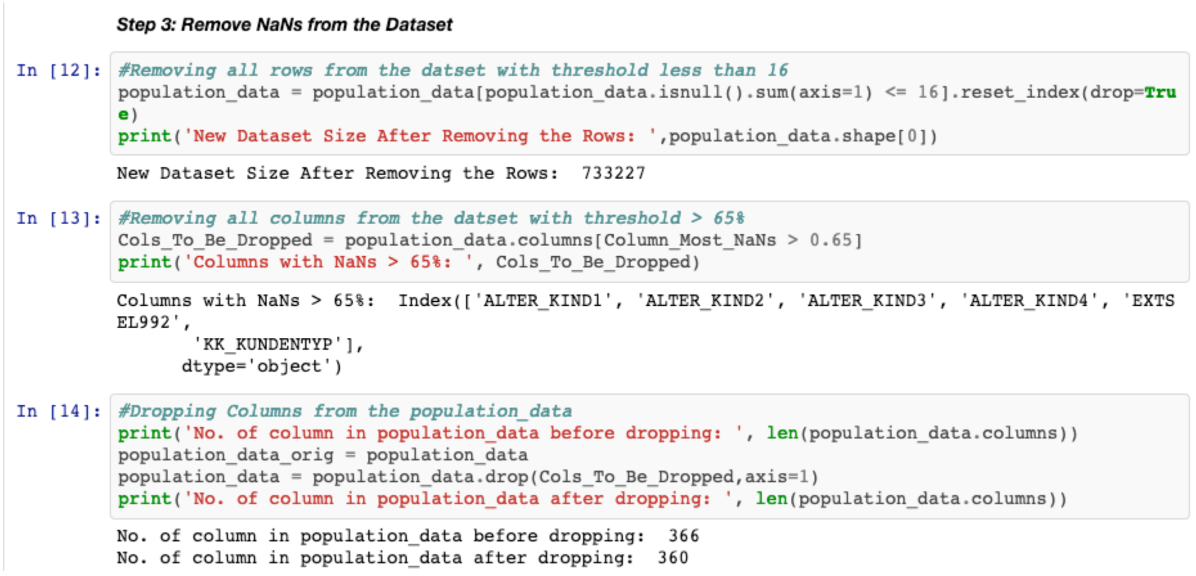
```
In [7]: #Top 10 Candidates
        population_data_NaNs_Percent.sort_values(ascending=False).head(10)
```

```
Out[7]: ALTER_KIND4                  99.864792
        ALTER_KIND3                  99.307691
        ALTER_KIND2                  96.690047
        ALTER_KIND1                  90.904837
        EXTSEL992                    73.399639
        KK_KUNDENTYP                 65.596749
        ALTERSKATEGORIE_FEIN         29.504130
        D19_LETZTER_KAUF_BRANCHE     28.849522
        D19_LOTTO                    28.849522
        D19_VERSI_ONLINE_QUOTE_12    28.849522
        dtype: float64
```



After identifying the ratio of columns and rows that were empty, we have determined a cut off ratio to remove the empty columns and rows. For columns, we identified a threshold of > 65% and for rows we have a threshold of < 16%.

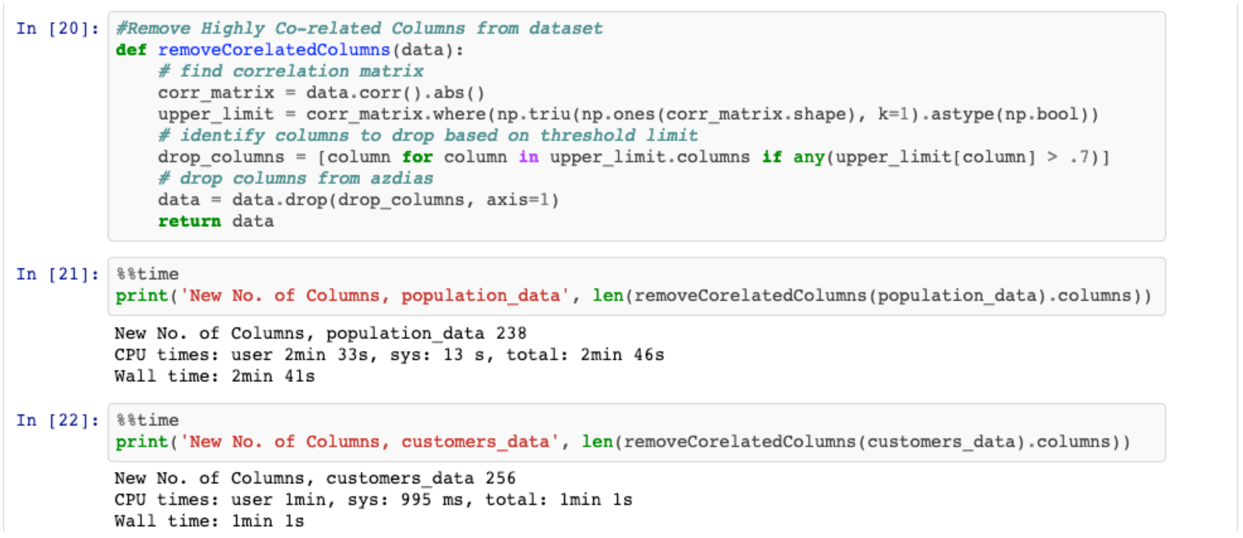




Step 3: Remove NaNs from the Dataset

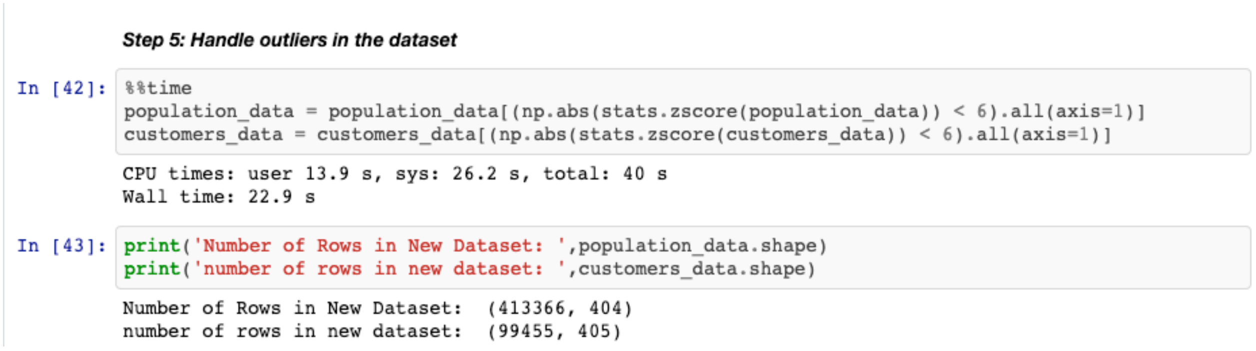
```
In [12]: #Removing all rows from the datset with threshold less than 16
         population_data = population_data[population_data.isnull().sum(axis=1) <= 16].reset_index(drop=Tru
         e)
         print('New Dataset Size After Removing the Rows: ',population_data.shape[0])

         New Dataset Size After Removing the Rows:  733227

In [13]: #Removing all columns from the datset with threshold > 65%
         Cols_To_Be_Dropped = population_data.columns[Column_Most_NaNs > 0.65]
         print('Columns with NaNs > 65%: ', Cols_To_Be_Dropped)

         Columns with NaNs > 65%:  Index(['ALTER_KIND1', 'ALTER_KIND2', 'ALTER_KIND3', 'ALTER_KIND4', 'EXTS
         EL992',
                'KK_KUNDENTYP'],
               dtype='object')

In [14]: #Dropping Columns from the population_data
         print('No. of column in population_data before dropping: ', len(population_data.columns))
         population_data_orig = population_data
         population_data = population_data.drop(Cols_To_Be_Dropped,axis=1)
         print('No. of column in population_data after dropping: ', len(population_data.columns))

         No. of column in population_data before dropping:  366
         No. of column in population_data after dropping:  360
```

Other techniques like removing highly co-related columns to reduce features from the dataset are used as part of the cleanup process.



```
In [20]: #Remove Highly Co-related Columns from dataset
         def removeCorelatedColumns(data):
             # find correlation matrix
             corr_matrix = data.corr().abs()
             upper_limit = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))
             # identify columns to drop based on threshold limit
             drop_columns = [column for column in upper_limit.columns if any(upper_limit[column] > .7)]
             # drop columns from azdias
             data = data.drop(drop_columns, axis=1)
             return data

In [21]: %%time
         print('New No. of Columns, population_data', len(removeCorelatedColumns(population_data).columns))

         New No. of Columns, population_data 238
         CPU times: user 2min 33s, sys: 13 s, total: 2min 46s
         Wall time: 2min 41s

In [22]: %%time
         print('New No. of Columns, customers_data', len(removeCorelatedColumns(customers_data).columns))

         New No. of Columns, customers_data 256
         CPU times: user 1min, sys: 995 ms, total: 1min 1s
         Wall time: 1min 1s
```

One-hot encoding and scaling are other techniques used to eliminate NaNs and standardize data. Z-scores were used to identify and remove outliers.



Step 5: Handle outliers in the dataset

```
In [42]: %%time
         population_data = population_data[(np.abs(stats.zscore(population_data)) < 6).all(axis=1)]
         customers_data = customers_data[(np.abs(stats.zscore(customers_data)) < 6).all(axis=1)]

         CPU times: user 13.9 s, sys: 26.2 s, total: 40 s
         Wall time: 22.9 s

In [43]: print('Number of Rows in New Dataset: ',population_data.shape)
         print('number of rows in new dataset: ',customers_data.shape)

         Number of Rows in New Dataset:  (413366, 404)
         number of rows in new dataset:  (99455, 405)
```

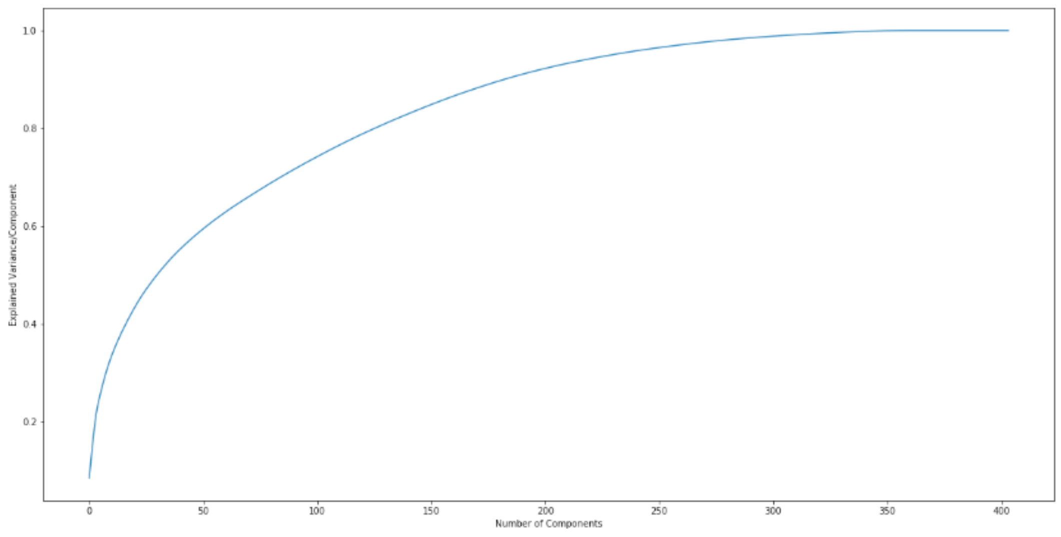
Algorithms and Techniques

Customer Segmentation:

In this section of the project we will be using unsupervised learning and dimension reduction techniques to reduce the number of features and will enable to find the customer cluster and attributes from the general population.

Reducing Dimensions:

Principal Component Analysis (PCA) is one of the most useful techniques in Exploratory Data Analysis to understand the data, reduce dimensions of data and for unsupervised learning in general. We will decide the number of input features to retain based on the cumulative variance.

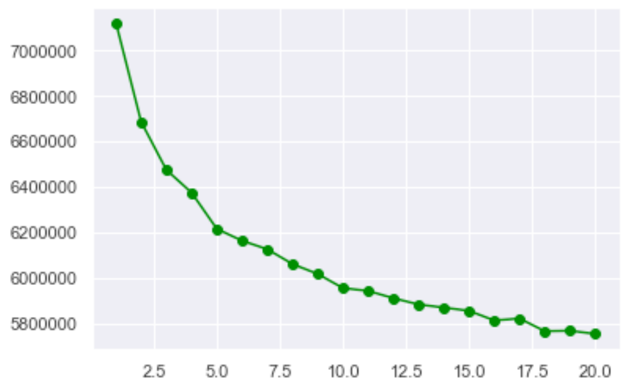


From the graph 260 is where the curve flattens and that's the number of principal components I have selected for the data with the cumulative variance around 95%. This the initial analysis, we can always come back and tune the numbers.

Creating Clusters:

The next step in the process is to use the reduced features to identify clusters from the dataset. I have used the K-means algorithm to identify the number of clusters in the dataset and have used the same technique with the customer dataset to identify potential customer base.

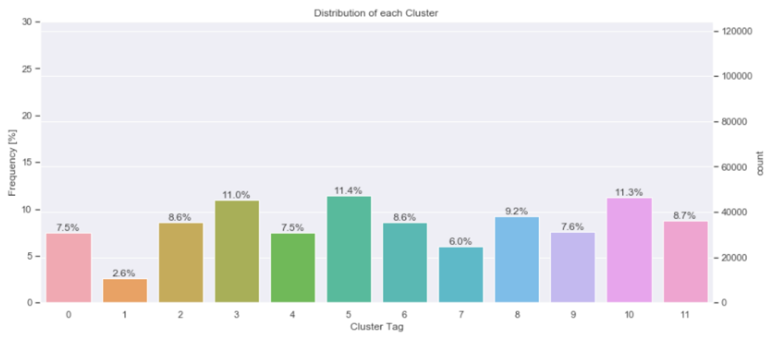
I have used the elbow method to identify optimal number of clusters available in the dataset.

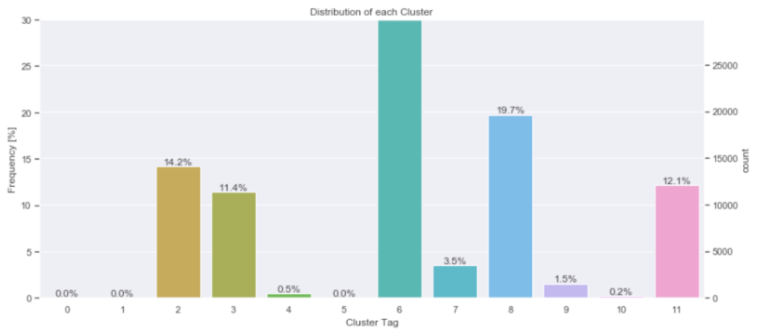


From the graph the elbow shape seems to form around 12 and after that the loss changes drastically. The ideal number of clusters will be 12. I used K-means and the number of clusters identified above to fit the customer data to identify the potential base.

Cluster Comparison:

In this section, I used the cluster data based on demographics of the general population of Germany and mapped the customer data from the mail-order sales company to those demographic clusters. The comparison of the two cluster distributions will enable us to identify the potential customer base.





Here we have two graphs that represents the distribution of population and customers of the company respectively. We can see the frequencies of cluster 6 and 8 in **Customers** data set are remarkably larger than the frequencies in **Population** (or general) dataset.

Considering the proportion of persons in each cluster for the general population and the customers the ratio should be fairly close but the mismatch of clusters between **Population** and **Customers** dataset indicates that there are only particular segments of the population that are interested in the company's products.

The huge difference of cluster 6/8 between two datasets suggests two things

- People in these two clusters could be potential target audience for the company
- Such large cluster difference could suggest the target base could have customers outside of the target demographics

Benchmark

The models built for the use case will use the demographics and customer data as a ground truth for evaluation and bench mark of the model and analysis. Given the dataset size, XGBoost regressor algorithm will be used as the benchmark model to evaluate and tweak the solution.

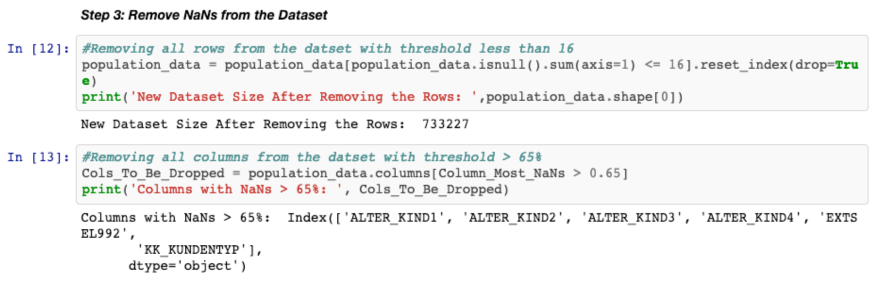
III. Methodology

Data Preprocessing

I followed a series of data cleaning steps to prepare the data for the next stages of the project. Some of them were like the following

- Identified NaNs or the unknowns from the dataset over rows and columns
- Removed columns with high correlation to reduce the feature set > 65%
- Removed rows with most unknowns or NaNs by identifying threshold < 16%
- Encoded the data for the remaining of the NaNs or unknowns in the dataset
- Handle the outliers in the dataset
- Scaled and standardized the dataset, used Mode as the filler value

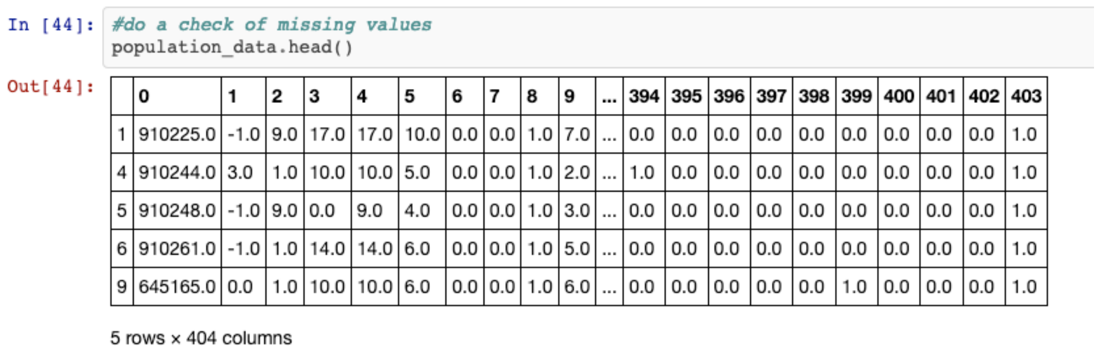
Implementation



Step 3: Remove NaNs from the Dataset

```
In [12]: #Removing all rows from the datset with threshold less than 16
         population_data = population_data[population_data.isnull().sum(axis=1) <= 16].reset_index(drop=Tru
         e)
         print('New Dataset Size After Removing the Rows: ',population_data.shape[0])

         New Dataset Size After Removing the Rows:  733227

In [13]: #Removing all columns from the datset with threshold > 65%
         Cols_To_Be_Dropped = population_data.columns[Column_Most_NaNs > 0.65]
         print('Columns with NaNs > 65%: ', Cols_To_Be_Dropped)

         Columns with NaNs > 65%:  Index(['ALTER_KIND1', 'ALTER_KIND2', 'ALTER_KIND3', 'ALTER_KIND4', 'EXTS
         EL992',
                'KK_KUNDENTYP'],
               dtype='object')
```

The dataset after all data preprocessing and encoding ended up looking like this, which will be used further.



```
In [44]: #do a check of missing values
         population_data.head()
```

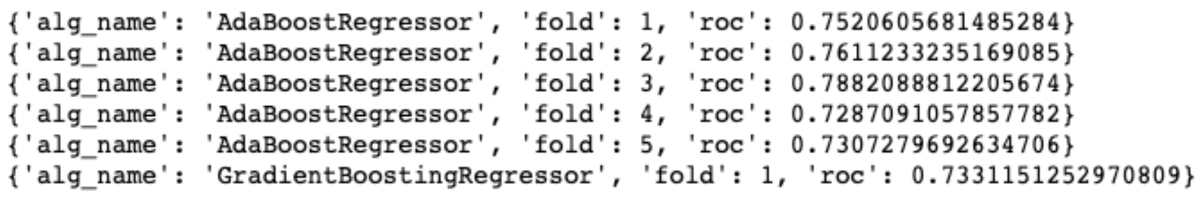
Out[44]:

	0	1	2	3	4	5	6	7	8	9	...	394	395	396	397	398	399	400	401	402	403
1	910225.0	-1.0	9.0	17.0	17.0	10.0	0.0	0.0	1.0	7.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
4	910244.0	3.0	1.0	10.0	10.0	5.0	0.0	0.0	1.0	2.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
5	910248.0	-1.0	9.0	0.0	9.0	4.0	0.0	0.0	1.0	3.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
6	910261.0	-1.0	1.0	14.0	14.0	6.0	0.0	0.0	1.0	5.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
9	645165.0	0.0	1.0	10.0	10.0	6.0	0.0	0.0	1.0	6.0	...	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0

5 rows × 404 columns

Refinement

As part of the model evaluation and performance, GridsearchCV techniques to evaluate multiple algorithms like AdaBoostRegressor, GradientBoostingRegressor to train the data and to calculate the corresponding roc_auc_score metircs.



```
{'alg_name': 'AdaBoostRegressor', 'fold': 1, 'roc': 0.7520605681485284}
{'alg_name': 'AdaBoostRegressor', 'fold': 2, 'roc': 0.7611233235169085}
{'alg_name': 'AdaBoostRegressor', 'fold': 3, 'roc': 0.7882088812205674}
{'alg_name': 'AdaBoostRegressor', 'fold': 4, 'roc': 0.7287091057857782}
{'alg_name': 'AdaBoostRegressor', 'fold': 5, 'roc': 0.7307279692634706}
{'alg_name': 'GradientBoostingRegressor', 'fold': 1, 'roc': 0.7331151252970809}
```

IV. Results

Model Evaluation and Validation

Supervised Learning and Output Predictions

In the last section of the project, we will apply supervised learning concepts to analyze the MAILOUT_TRAIN and MAILOUT_TEST dataset provided by the mail order company to predict probability of a person becoming a potential customer of the company.

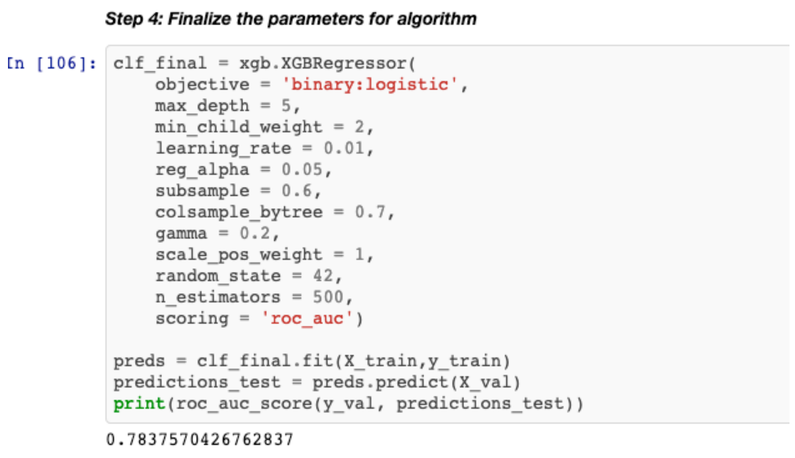
Analyze data and split the input features and label

I have split the dataset to identify the input features and label. The last column of the dataset is the label, RESPONSE we got from the potential customer.

Model Evaluation and Prediction:

There are really good algorithms like AdaBoostRegressor, GradientBoostingRegressor but I went with the XGBRegressor which works well for classification problems. I used 5-fold validation to train the data. I used techniques like GridSearchCV to identify the hyper parameters for the model.

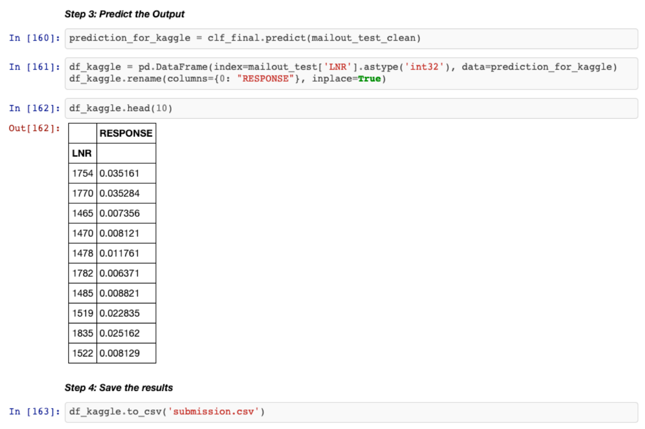
With the final set of parameters, I was able to achieve a roc_auc_score score of 0.78375.



Step 4: Finalize the parameters for algorithm

```
In [106]: clf_final = xgb.XGBRegressor(
              objective = 'binary:logistic',
              max_depth = 5,
              min_child_weight = 2,
              learning_rate = 0.01,
              reg_alpha = 0.05,
              subsample = 0.6,
              colsample_bytree = 0.7,
              gamma = 0.2,
              scale_pos_weight = 1,
              random_state = 42,
              n_estimators = 500,
              scoring = 'roc_auc')

          preds = clf_final.fit(X_train,y_train)
          predictions_test = preds.predict(X_val)
          print(roc_auc_score(y_val, predictions_test))

          0.7837570426762837
```

Using the same steps above, I have handled the Udacity_MAILOUT_052018_TEST.csv dataset to predict the output for the Kaggle competition and exported the output in a csv file for submission.



Step 3: Predict the Output

```
In [160]: prediction_for_kaggle = clf_final.predict(mailout_test_clean)

In [161]: df_kaggle = pd.DataFrame(index=mailout_test['LNR'].astype('int32'), data=prediction_for_kaggle)
          df_kaggle.rename(columns={0: "RESPONSE"}, inplace=True)

In [162]: df_kaggle.head(10)

Out[162]:
```

	RESPONSE
LNR	
1754	0.035161
1770	0.035284
1465	0.007356
1470	0.008121
1478	0.011761
1782	0.006371
1485	0.008821
1519	0.022835
1835	0.025162
1522	0.008129

Step 4: Save the results

```
In [163]: df_kaggle.to_csv('submission.csv')
```

Justification

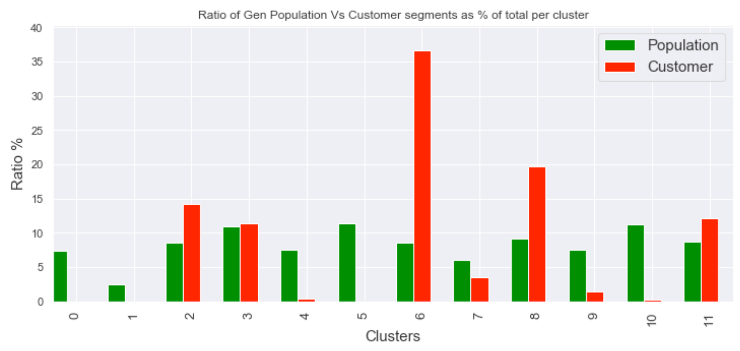
On quick analysis, we could find that the dataset is not balanced as for 43k ads only 532 people responded to it. Due to the imbalance nature of the dataset I have made for the following decisions to handle the same

- Use the data cleanup function created earlier to celan the data
- Use stratified K-Fold technique to split the data for testing and training
- Use XGBoostRegresssor algorithm as the problem is a classic classification problem
- Use roc_auc_socre as a metric for evaluating the model

V. Conclusion

As part of the project we have performed data processing and analysis techniques. First part of the project deals with clustering use base using unsupervised algorithms like K-means.

Free-Form Visualization

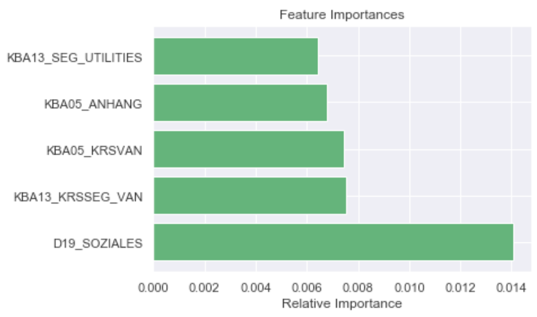


The huge difference of cluster 6/8 between two datasets suggests two things

- People in these two clusters could be potential target audience for the company
- Such large cluster difference could suggest the target base could have customers outside of the target demographics

Popular Features:

Popular Features identified from the dataset. D19_SOZIALES turned out to be the most important features of all in the dataset. D19 series in the dataset corresponds to the transactional level details.



Reflection

The overall project is a really good practical example of how to process real-life data and handle challenges while processing data and solve real life problems. The scope of the project ranges from data analysis to unsupervised learning to supervised learning along the way providing an ability to make decisions to achieve the end goal.

On a high level I have performed the following actions to solve the problem of identifying potential customers to the mail order company.

High Level Process Flow:

- Collect Data for the use case
- Clean up and Pre-process the demographics (General & Customer) data
- Feature Engineering
- PCA, reduce input features to identify effective input features
- Split the dataset into test, train and validation
- Generate the shoulder curve to identify the cluster size
- Apply K-means algorithm to the processed dataset to identify clusters
- Identify potential customer base clusters by comparing general vs customer cluster
- Clean up and Pre-process the given train and test customer data
- Identify Input Features
- Create a model
- Train the model
- Test the model with test customer data
- Validate output data and model performance
- Infer insights from the final predicted output

Improvement

- Implement a similar solution in Sagemaker by using its built-in tools and available algorithms
- Try a different way to present the outcome and the results in a different way
- Would like to try different algorithms and train them for longer iteration to test the data
- Would like to implement deep learning model for the business problem

References:

- https://jakevdp.github.io/PythonDataScienceHandbook/index.html
- https://stackoverflow.com/questions/29294983/how-to-calculate-correlation-between-all-columns-and-remove-highly-correlated-on
- https://stats.stackexchange.com/questions/143700/which-is-better-replacement-by-mean-and-replacement-by-median
- https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba
- https://stackoverflow.com/questions/23199796/detect-and-exclude-outliers-in-pandas-data-frame
- https://machinelearningmastery.com/rescaling-data-for-machine-learning-in-python-with-scikit-learn/