Capstone Project

Customer Segmentation Report for Arvato Financial Services

Problem Statement:

In this project we will be using available datasets on customers of Arvato to target new client base with the help of machine learning techniques.

Which individual's company should target who are most likely to convert into becoming customers of the company?

We will solve this problem in sub-parts:

1) Unsupervised learning techniques to perform customer segmentation
2) Supervised learning approach to target customers for the marketing campaign

Data Exploration and Visualization:

To perform customer segmentation, we need to use the following datasets Udacity_AZDIAS_052018.csv and Udacity_CUSTOMERS_052018.csv. Both datasets have a lot of same features however, the customers datasets contain three extra columns ('CUSTOMER_GROUP', 'ONLINE_PURCHASE', and 'PRODUCT_GROUP') which provide broad information about customers. These extra columns were dropped from customers dataset to make the azdias and customer dataset similar.

```
# remove extra columns from customers data
customers_extra_columns=set(customers.columns.tolist())-set(azdias.columns.tolist())
print(customers_extra_columns)
customers.drop(customers_extra_columns,axis=1, inplace=True)
```

```
{'PRODUCT_GROUP', 'ONLINE_PURCHASE', 'CUSTOMER_GROUP'}
```

```
print(azdias.shape)
print(customers.shape)
```

```
(891221, 366)
(191652, 366)
```

Missing Data Analysis:

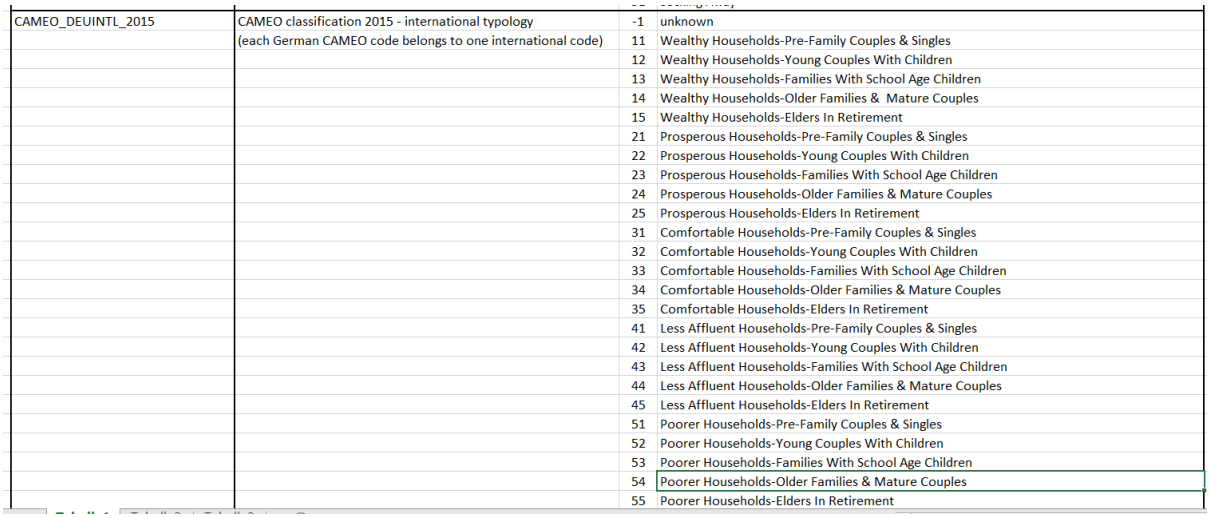
After, I explored and understood the data that I was working with. I calculated the percentage of missing values for each feature. Most of the features have less than 20% missing values hence the features were dropped with more than 20% missing values.

After that, all the rows were deleted which have missing data for more than 10 columns.

Feature Engineering:

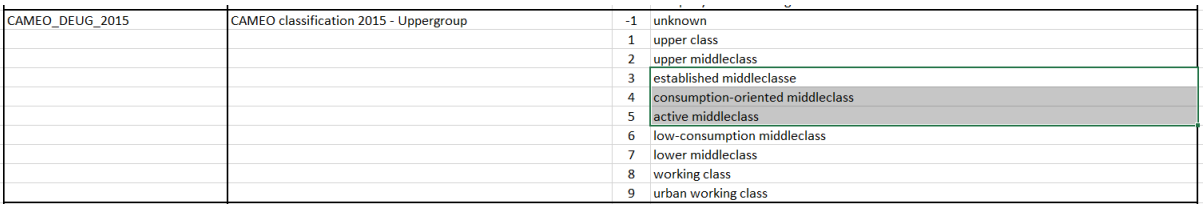
Supervised and unsupervised learning techniques only work on data that is encoded numerically. Most of the features in our datasets are in numerical form which do not need additional processing, some features are in object form which need additional processing. To do the additional processing I created clean_data function to convert all the features in numerical form.

1) **OST_WEST_KZ:** is a binary feature that had the values (W, O) in the original dataset which I mapped to (1,0) respectively.
2) **VERS_TYP and ANREDE_KZ:** Converted these features into {0,1} scale from {1,2} scale.
3) **CAMEO_INTL_2015:** I noticed that the first digit represents the wealth of the household and second digit represents life stage of the household hence I created two separate features to represent wealth and life stage -> CAMEO_INTL_2015_WEALTH and CAMEO_INTL_2015_LIFESTAGE features.



CAMEO_DEUINTL_2015	CAMEO classification 2015 - international typology	-1	unknown
	(each German CAMEO code belongs to one international code)	11	Wealthy Households-Pre-Family Couples & Singles
		12	Wealthy Households-Young Couples With Children
		13	Wealthy Households-Families With School Age Children
		14	Wealthy Households-Older Families & Mature Couples
		15	Wealthy Households-Elders In Retirement
		21	Prosperous Households-Pre-Family Couples & Singles
		22	Prosperous Households-Young Couples With Children
		23	Prosperous Households-Families With School Age Children
		24	Prosperous Households-Older Families & Mature Couples
		25	Prosperous Households-Elders In Retirement
		31	Comfortable Households-Pre-Family Couples & Singles
		32	Comfortable Households-Young Couples With Children
		33	Comfortable Households-Families With School Age Children
		34	Comfortable Households-Older Families & Mature Couples
		35	Comfortable Households-Elders In Retirement
		41	Less Affluent Households-Pre-Family Couples & Singles
		42	Less Affluent Households-Young Couples With Children
		43	Less Affluent Households-Families With School Age Children
		44	Less Affluent Households-Older Families & Mature Couples
		45	Less Affluent Households-Elders In Retirement
		51	Poorer Households-Pre-Family Couples & Singles
		52	Poorer Households-Young Couples With Children
		53	Poorer Households-Families With School Age Children
		54	Poorer Households-Older Families & Mature Couples
		55	Poorer Households-Elders In Retirement

4) **CAMEO_DEUG_2015:** Combined same classes into one like {upper class, upper middleclass} were treated as 1, {established middleclasse, consumption-oriented middleclass and active middleclass} were treated as 2 so on.



CAMEO_DEUG_2015	CAMEO classification 2015 - Uppergroup	-1	unknown
		1	upper class
		2	upper middleclass
		3	established middleclasse
		4	consumption-oriented middleclass
		5	active middleclass
		6	low-consumption middleclass
		7	lower middleclass
		8	working class
		9	urban working class

5) **PRAEGENDE_JUGENDJAHRE:** Created a decade feature combining same decades into one group. Like {40ies - war years (Mainstream, O+W) and 40ies - reconstruction years (Avantgarde, O+W)} were treated as 1.

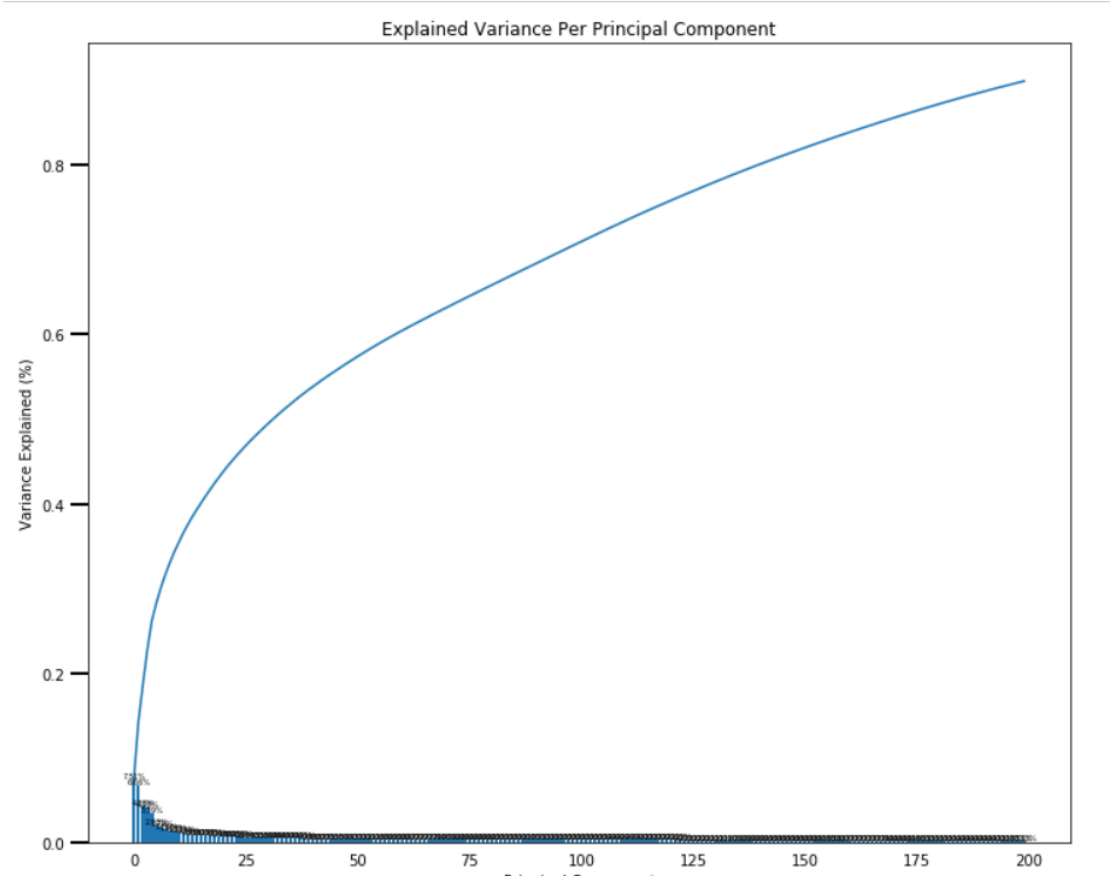
6) **CAMEO_DEU_2015:** is a categorical feature with all values represented by strings hence I used “get_dummies” function to one-hot-encoding this feature.

7) Removed the EINGEFUEGT_AM (was not defined in the attributes sheet and also was in the object format) and LNR (identification code – no need to keep it in the dataset) features from the dataset as they did not look very important features.

With the above processing, a data cleaning function was created to clean the azdias and customers data. Prior to PCA, additional processing methods are required to transform the dataframe to all number-based matrix. The first step is imputing all 'nans' to value where we used "Imputer" function and the second step is normalizing all data where we used "StandardScaler" function.

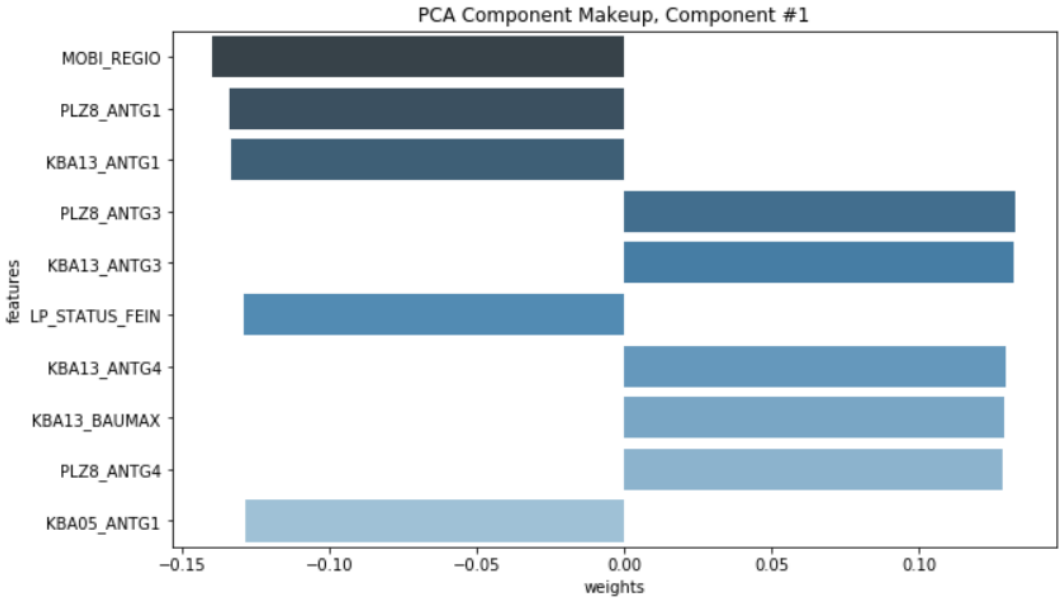
Dimensionality Reduction:

On the scaled data, our data is now ready to apply dimensional reduction techniques. PCA is applied for the dimension reduction. To decide how many components, I need to keep that accounted for over 70% of the variance in the results I used a scree plot.



Based on the above plot, we can see 70% information is retained with only 110 components.

Now we can examine the makeup of each PCA component based on the weightings of the original features that are included in the component. Please find below the PCA component makeup for component #1.



Clustering:

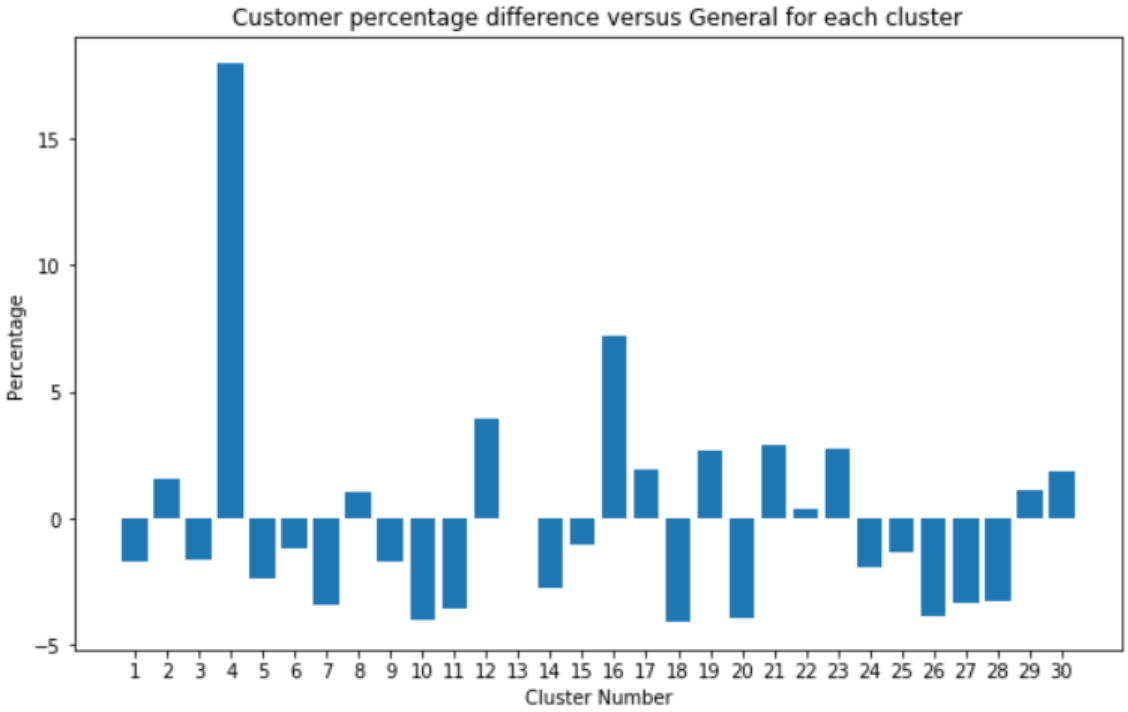
On the transformed data, we will apply K-means clustering to the dataset. The same clustering model was also performed on customers data to see how market segments differ between the general population and mail order sales company.

The average centroid distance between cluster points and a centroid is plotted with a different number of centroids in the k-mean clustering method. Elbow method is used to determine the optimum number of clusters.

From the plot, 30 clusters look like a reasonable value. Hence, we can apply the same transformation to customer datasets.

Customer Segmentation:

Now two cluster distributions were generated to see where the strongest customer base for the company is.



The above figures show the percentage difference between the customer dataset and general dataset. We can see Cluster 4 is the most over-represented cluster by the customer dataset, while Custer 18 is the most under-represented cluster. The features that correlated to the clusters are further investigated:

First, we counted the data points of each cluster for general population and company's customer database. We found out that cluster 4 has 17.94% higher customer percentage compared to general population and cluster 18 is under-presented with a difference of -4.1%.

For Cluster 4, we find that VK_ZG11, HH_EINKOMMEN_SCORE and VK_DISTANZ are the highest impacted feature. HH_EINKOMMEN_SCORE shows the income of a household which impacts the customer distribution, VK_ZG11 and VK_DISTANZ are not described in attributes file. Negative correlated features are LP_STATUS_GROB (Social status rough), FINANZ_MINIMALIST (Low Financial Interest) and LP_STATUS_FEIN (Social Status Fine) which shows better financial backed people are more likely to be a potential customer.

For cluster 18, highest impacted features are FINANZ_MINIMALIST (Low Financial Interest), CJT_TYP_6 (Not defined) and CJT_TYP_3 (Not defined). It shows that we should not target those people who have low financial interest. Negative correlated features are SEMIO_RAT, CJT_TYP_1 (Not defined) and CJT_TYP_2 (Not defined) which shows rational personality people are less likely to be the customer of the company.

Supervised Learning Method:

Now we have found out that which parts of the population are more likely to be customer of the company. Its time to build supervised learning model.

Each of the rows in the "MAILOUT" data files represents an individual that was targeted for a mailout campaign. The "MAILOUT" data has been split into two approximately equal parts, each with almost 43 000 data rows.

Data Cleaning:

In train model, we have one extra column named “RESPONSE” that states whether or not a person became a customer of the company following the campaign. Hence we create “response” dataframe to represent the train data and dropped the “RESPONSE” column from the train dataset.

Now we can check the distribution of the output label and surprisingly only 532 out of 42,962 data is positive. The dataset is highly imbalanced because of the disproportionate number of customers and non-customers as positive responses are only 1.23%.

Like unsupervised learning model, we need to clean the data by dropping the high percentage of missing value features. However here we cannot drop the row data as data is highly imbalanced there is a possibility, we can drop positive responses also.

After cleanup, additional processing methods are required to transform the dataframe to all number-based matrix. The first step is imputing all ‘nans’ to value where we used “Imputer” function and the second step is normalizing all data where we used “StandardScaler” function.

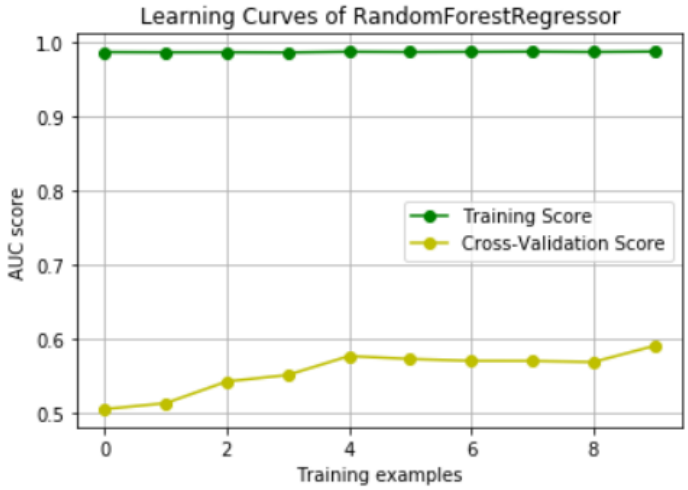
Model Selection/Evaluation:

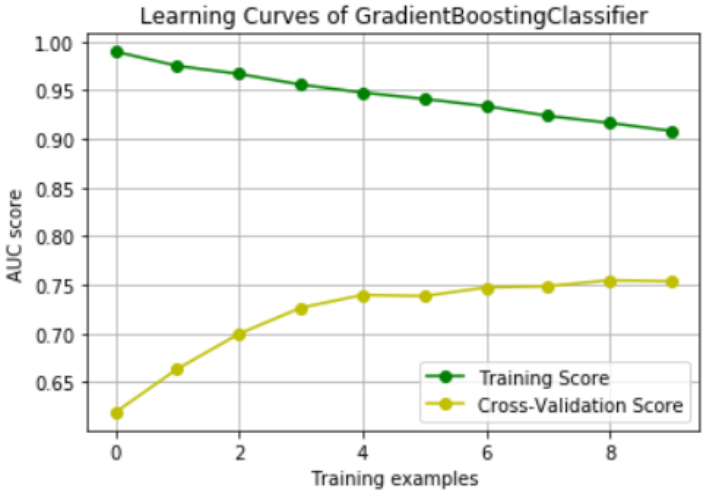
Since there is a large imbalance in the dataset, predicting individual classes and using accuracy does not seem to be an appropriate performance evaluation method. Instead, we used ROC-AUC to evaluate performance. There are number of approaches to deal with class imbalance. We will use the following models to train the data:

1) XGBRegressor

2) RandomForestRegressor

3) GradientBoostingClassifier





Based on model performance it was quite clear that GradientBoostingClassifier was winner out of these 3 models, training score keeps decreasing and cross-validation score keeps increasing. while

for the other two models training score remains constant and the cross-validation score is low compared to GradientBoostingClassifier model. Therefore, GradientBoostingClassifier is adopted for optimization and training.

Improving Results:

The best model had a mean score of 76.8% on the training sets which was achieved by tuning the following parameters (number of estimators, Learning Rate, max_depth, min_samples_split and min_samples_leaf). GridSearchCV was used to tune each parameter with atleast 3-4 values. Grid search scored using the AUC. The improvements was only marginal despite number of parameters tuned. It was difficult to get a big leap in performance by just tuning the parameters. A significant jump may be achieved by feature engineering, but it requires specific domain knowledge on the dataset.

```
clf = GradientBoostingClassifier(random_state=1)

# use gridsearch to find best fitted model

parameters = {
            'n_estimators': [15],
            'learning_rate':[0.1],
            'max_depth': [5],
            'min_samples_split':[220],
            'min_samples_leaf':range(30,71,10)
             }


best_predicter = grid_search(clf, parameters, X_train, y_train)
```

```
Best score:  0.7634379084047392
Best params:  {'learning_rate': 0.1, 'max_depth': 5, 'min_samples_leaf': 30, 'min_samples_split': 220, 'n_estimators': 15}
```

```
best_predicter
```

```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=5,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=30, min_samples_split=220,
                           min_weight_fraction_leaf=0.0, n_estimators=15,
                           n_iter_no_change=None, presort='auto',
                           random_state=1, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

Kaggle Competition:

Now that we have created a model to predict which individuals are most likely to respond to a mailout campaign, it's time to test that model in competition through Kaggle.

The finely tuned model was used to fit the complete training dataset to predict the response variable of the testing dataset. Using the optimized GradientBoostingClassifier test dataset is predicted and saved the results in result2.csv file.

```
output.head(5)
```

	LNR	RESPONSE
0	1754	0.968665
1	1770	0.975377
2	1465	0.995235
3	1470	0.995211
4	1478	0.978337

Conclusion:

At the moment the most challenging part is testing and optimization which requires a lot of time to do the optimization. As cloud-based training machine learning model is valuable hence its's not possible to do too many iterations.

This project was incredibly challenging. I am excited to see how much I have learned during this project. I think there are few aspects of the implementation which could be improved.

1) Changing the missing value threshold which we used to drop the columns and similar for rows.
2) Sampling techniques to balance classes
3) Aside from the machine learning algorithms, the score may be improved by feature engineering. Experts have mentioned that usually domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering would require looking each feature individually and decide whether you want to keep it or discard it. This again would require specific domain knowledge.