Customer Segmentation: Arvato Financial Services

Jaweria Shabbir

December 8, 2020

Machine Learning Engineer Nanodegree

Project Report

Project Definition

Project Overview:

Arvato-Bertelsmann financial solutions company is headquartered in Germany; it provides customer support services to numerous companies in different domains. Arvato-Bertelsmann uses latest technology to provide services to their customers. It focuses on using cutting edge technology combined with predictive analytics to aid businesses make important decisions based on large datasets and analysis.

Customer Segmentation has become a very popular practice with today's market. It helps sellers, identify their customers and perform targeted advertisement for them. This usually helps save money, and gives greater return on the investment. This application can be altered and used in many other applications aside from selling products or services.

Customer Segmentation divides populations and predicts future customers based on features and characteristics. This is crucial for business growth, and effective marketing strategy. Customer segmentation allows companies to prioritize target advertising to customers who are more likely to become to their customers, this also reduces marketing costs. Predictive analytics help build long term relationship between business and customers, the continuous relations is significant to a business success (James M. Curran, Sajeev Varki, Deborah E. Rosen (2010)).

Arvato-Bertelsmann financial solutions company works with a mail-order company to sell their organic products. The data provided by Arvato for this customer segmentation-project is based on the customers of mail-order company and general population Germany.

Problem Statement:

Based on the data given, what demographics in Germany are most likely to become customers for a mail-order company that sells organic products? Identify the population of people that are more likely to become future customers for the mail-order company among the general population?

Analysis:

Data Exploration:

The data has been provided by Arvato-Bertelsmann Financial Services. The data provided is based on general population of Germany and the customers of a mail-order sales company. The data provided can predict the future customers based on thorough analysis and predictive model. The first two files [Azdias and Customers] will be used to see similarities and differences between customers vs. population at large. The analysis from those to files will be used to make prediction on the other two files [MAILOUT] and predict which individuals are more likely to become customers. The 'customers' data includes three extra columns ('CUSTOMER_GROUP',

'ONLINE_PURCHASE', and 'PRODUCT_GROUP') which provide further information about the customers. The MAILOUT_TRAIN file contains an additional column "RESPONSE", that indicates whether or not the individual became a customer of the mail-order sales company.

The four separate datasets are listed below with brief descriptions:

- **Udacity_AZDIAS_052018.csv:** contains demographics data for the general population of Germany. Each row represents an individual and the columns represent features [891,211 rows and 366 features]
- **Udacity_CUSTOMERS_052018.csv**: contains demographics data for the existing customers of mail-order Company [191,652 rows (individuals) with 369 columns (features)].
- **Udacity_MAILOUT_052018_TEST.csv**: Demographics data for individuals who were targets of a marketing campaign; 42 982 persons (rows) x 367 (columns).
- **Udacity_MAILOUT_052018_TRAIN.csv**: Demographics data for individuals who were targets of a marketing campaign; 42 833 persons (rows) x 366 (columns).

There are two other excel sheets which provide additional attributes to the datasets.

- **DIAS Information Levels - Attributes 2017.xlsx**: top-level list of attributes and descriptions, organized by informational category.
- **DIAS Attributes - Values 2017.xlsx**: detailed mapping of data values for each feature in alphabetical order.

All four datasets are downloaded and analyzed for its shape and size. Numbers of rows and columns were not uniformed throughout the four datasets.

```
"Loaded AZDIAS data with size: (891221, 366) Loaded Customers data with
size : (191652, 369 Loaded Training data with size : (42962, 367) Loaded
Testing data with size : (42833, 366)")
```

The data was explored; to detect nan values, non-numeric values, and missing values. These values cause lots of noise in the dataset. Therefore I created a map to keep track of each column. Non-numeric columns will be transformed into one-hot-encoded; missing values are replaced with mean of the column. Columns with insignificant data [for this purpose] are dropped; for example column name: OST_WEST_KZ. For the customers data, columns: PRODUCT_GROUP, 'CUSTOMER_GROUP, and ONLINE_PURCHASE are dropped, because these columns are not present in general population data. This makes the two dataset have the same shape. I created a helper function to transform the columns with noisy data; it is included in the file.

It is also important to determine whether the data is balanced, this helps evaluate how well are given dataset is imbalanced because it has a lot more negative responses than positive responses. In the normalized dataset, there are 42,430 negative responses, while there are only 532 positive

responses from the customers. I will keep this in mind as I choose evaluation metrics and algorithm techniques. The metric scores performing and which metric is best suitable for the dataset provided. The

Visualization:

Missigno library is used to visualize the missing data in graphs and further decide which columns or to drop and understand the correlations between data. Some of the graphs are included below for general population data.

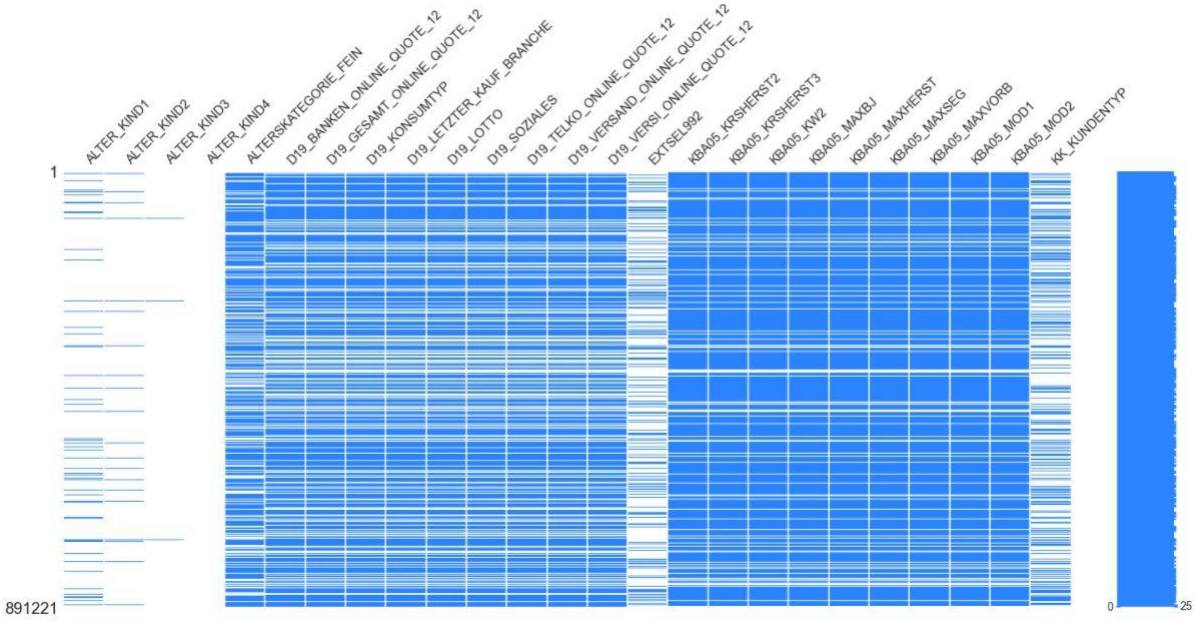

Figure 1: Matrix graph representing missing data with white space.

Missingno library matrix graph above renders visualization for the missing data in the datasets. White lines indicate the missing values. It makes it easy to visualize which columns or rows are not providing valuable data, and if they should be dropped.

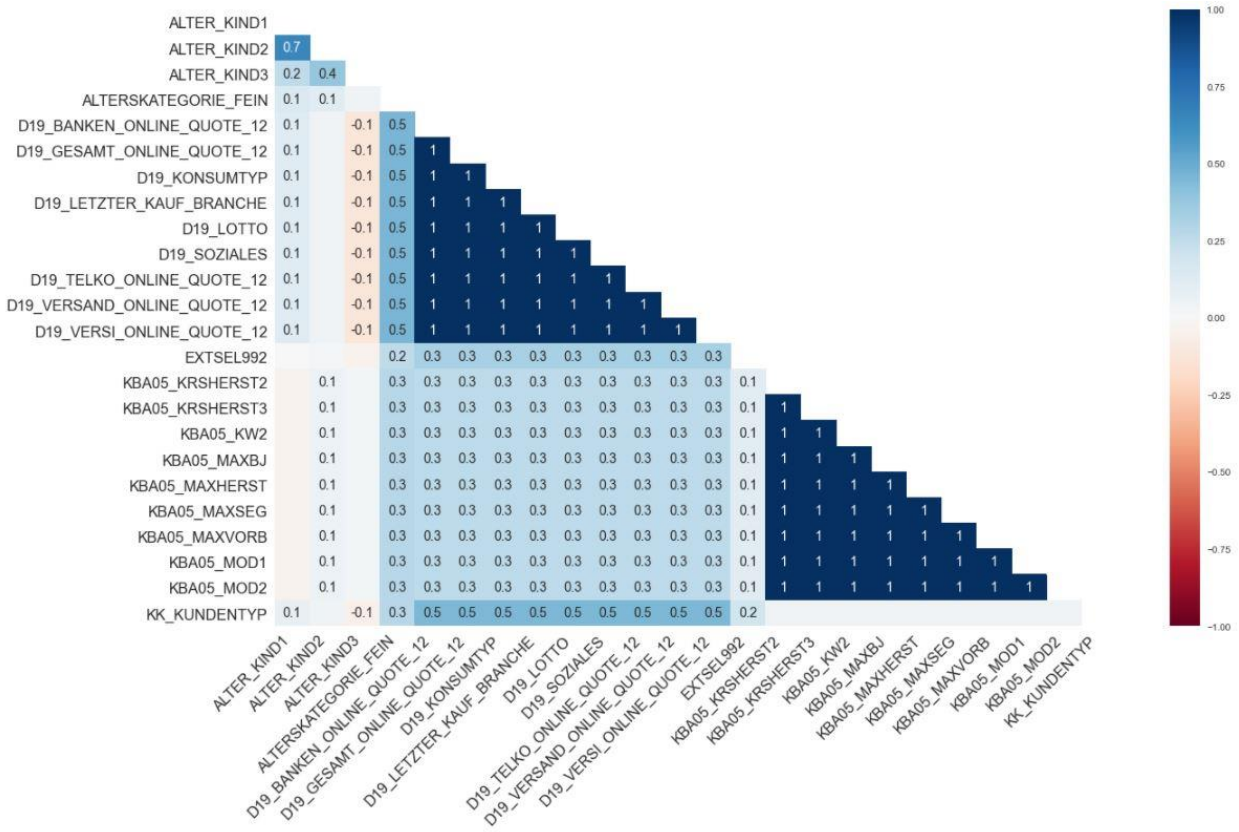

Figure 2: Heatmap shows correlation in the datasets among different features

Heatmap above is used to visualize the correlation between different columns. Positive correlation is indicated by darker color. Correlation map helped me determined how each column is affected by another, and what effect do the missing values have on the rest of the data. It helped determine which columns should be dropped.

Methodology:

Data Preprocessing:

Clean data function is created to map different abnormalities in columns and transform them accordingly. Non-numeric data is one hot encoded, missing data is filled with mean values and some columns with insignificant are dropped.

Analyze the amount of nan values for a column - divide by the number of total rows to get the ratio. The NAN is filled with the string "NAN" so it is easier to see the value counts.247 columns are found with more than 10.0% NAN values. Columns with more than 10% NAN values are detected and dropped. Data with too many NAN values decreases the quality of the data. I included assertion based testing to make sure my data is uniform, and the data is cleaned as intended.

Implementation:

Unsupervised Learning:

After the data is cleaned to some extent, clustering is applied. I chose to use unsupervised clustering because I am most familiar with clustering algorithm and it is very simple to apply and interpret. This would be a great start and it will give me more insight to the dataset. However, feature selection, or dimensionality reduction is not applied instantly. Naïve clustering is used as a benchmark. Clustering is applied to every column with the exception of label column. The result of the first clustering did not convey any useful information. Clusters of existing customers and clusters of general population overlapped. This doesn't help much with analysis; we need to process the data further for it to be more meaningful.

While scaling on one-hot encoded features can be performed mathematically, it provides nothing useful practically. Scaling one-hot encoded features only adds noise to the data. Additionally, PCA and other dimensionality techniques expect continuous values, so using them on non-continuous categorical features do not throw an error but it means nothing.

Instead of scaling everything naïvely, only the continuous value columns are scaled. Feature Selection is performed on the scaled & categorical features, followed by dimensionality reduction.

Graph below shows feature scores from the dataset. One attribute "LP_LEBENSPHASE_FEIN" far outweighs the other. Top 50 features are chosen and the other 147 columns are dropped. It is important to note that there is no one right way of doing this. Features selected can be switched around, the ones selected are based on the graph.

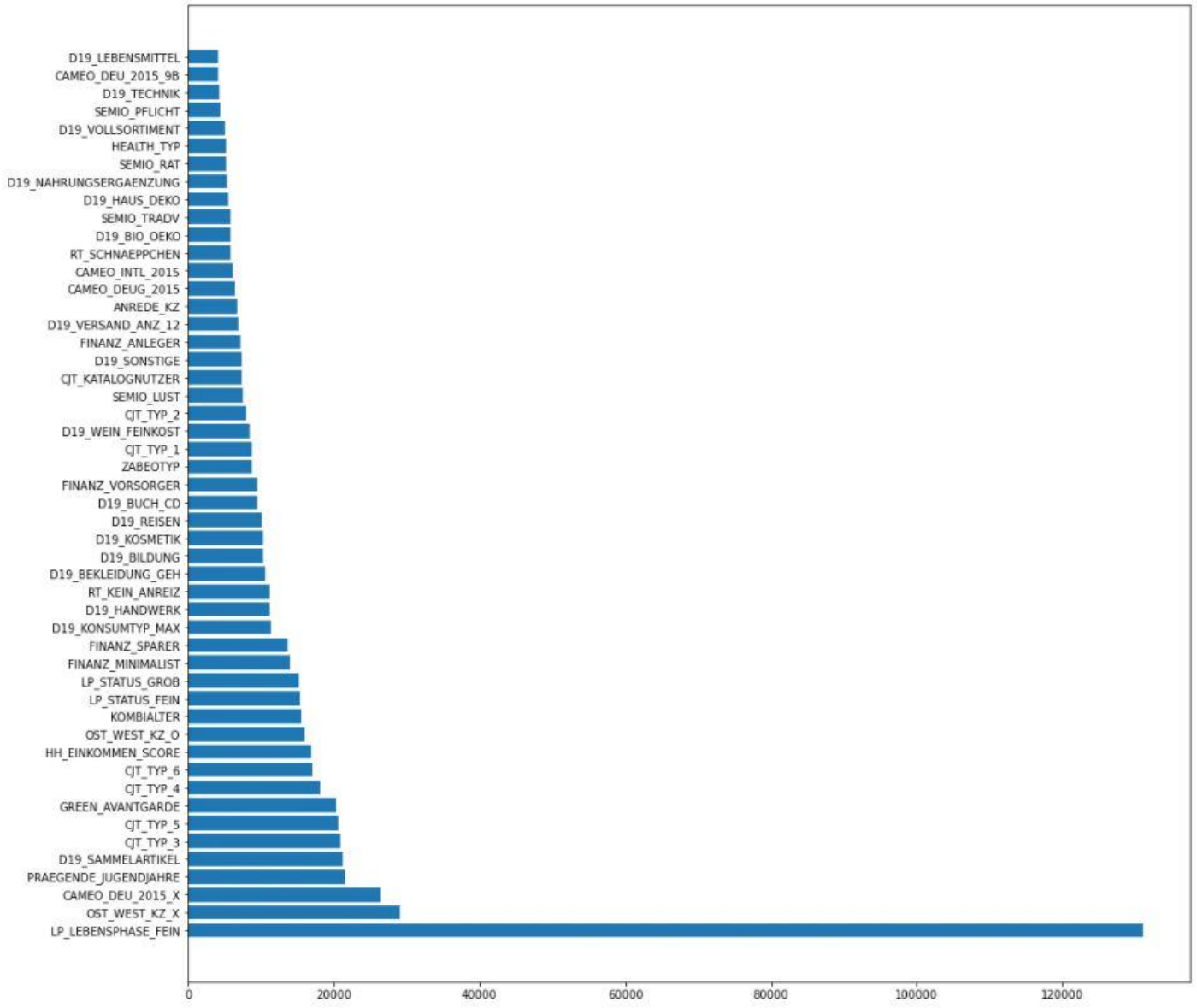

Figure 3: Bar graph shows feature scores from the dataset, LP_LEBENSPHASE_FEIN with the highest score of well above 120000.

The number of components onto which to project the data heavily influences the reconstruction error. High reconstruction error indicates information in the original data was lost. The benefit of having less components is it can lead to better generalization and training times.

The way to determine a good number of components is to look at the reconstruction error. The error is calculated as the normalized root mean square error between the original and data reconstructed from the projected data. The bar plot below shows the reconstruction errors. Based on the bar plot, if a 0.02 reconstruction error is deemed acceptable, than it looks like we need 40 components to achieve dimensionality reduction without introducing too much noise.

The threshold used to determine number of components to use is based on heuristics. There are only approximately right answers here.

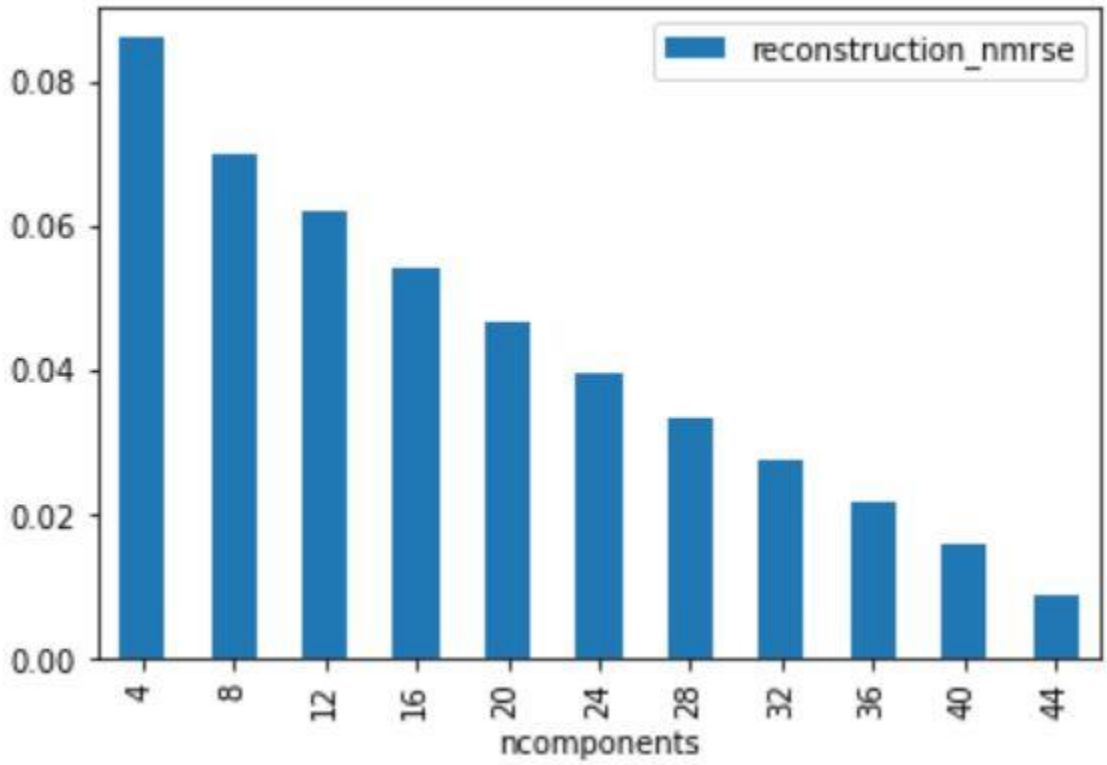

Figure 4: Bar plot shows reconstruction error with different number of components chosen.

PCA is applied with 40 components (as determined using the bar plot above). The bar plot below visualizes the explained variance ratio. As shown in the graph and data below {explained variance ratio}, principal component 1 holds most of the data and most of the data is retained.

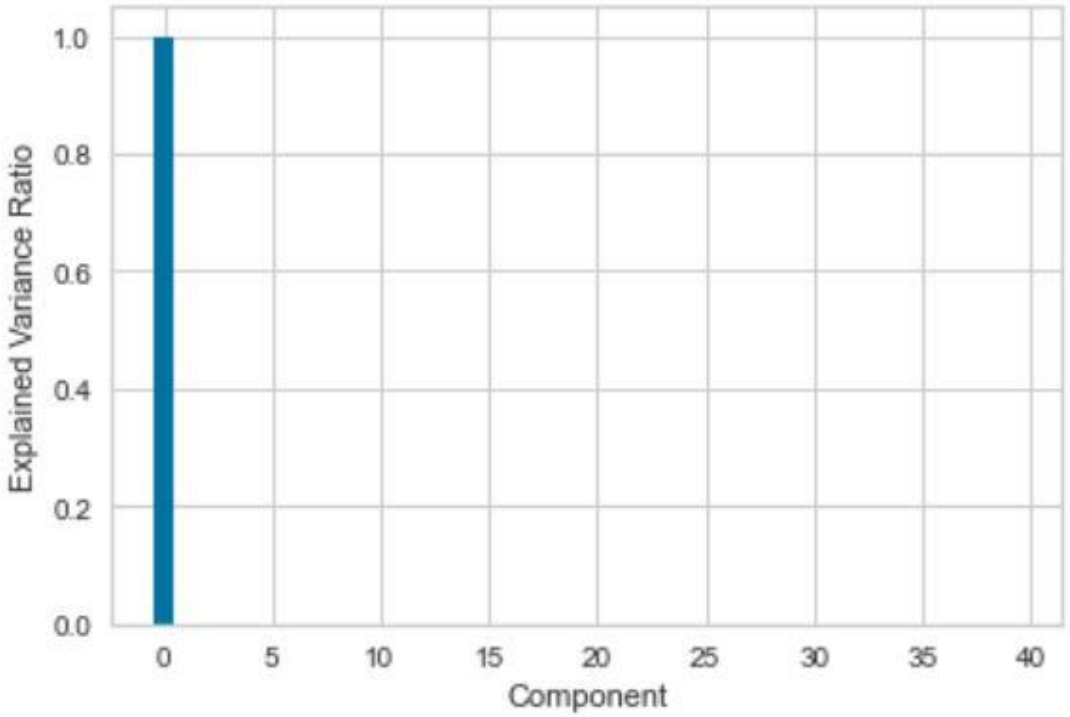

Figure 5: explained variance ratio with one principal component holding most of the data

Elbow graph was then used to visualize clusters and figure out the optimum number of clusters. Yellobrick package is used for visualization. Distortion Score Elbow for KMeans Clustering is gra phed below. Elbow at k=6 so we choose 6 clusters as optimal then evaluate the cluster metric. We can use the clusters into which a lot of customers fall as the clusters that contain "customer or potential customers."

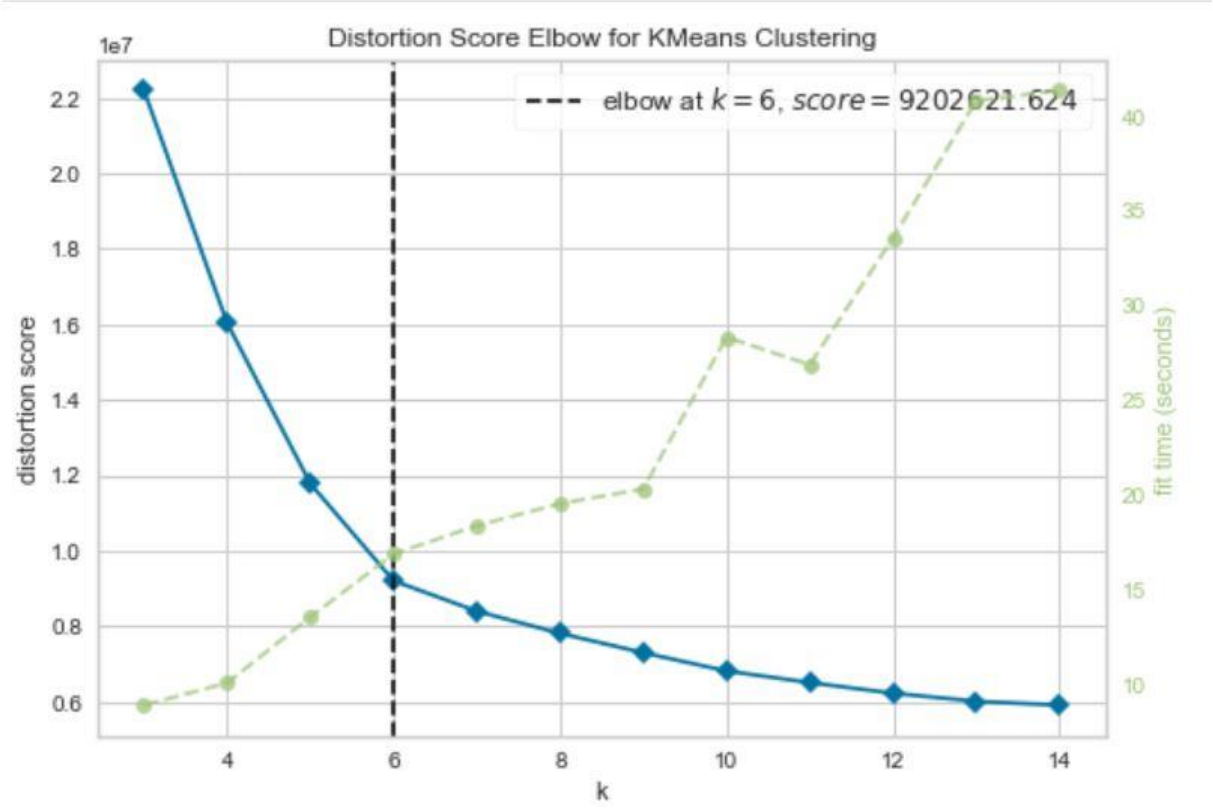

Figure 6: Elbow graph to determine the optimum number of clusters to use for clustering.

Clustering is applied again after feature selection and PCA, with optimum number of clusters. Table below shows the results. According to the results Clusters 0 and 1 hold more than 50% of the customers. People from the general population that fall into clusters 0 and 1 are more likely to transition to becoming customers. Within the customers, clusters 0 and 1 contain almost equal sample counts, meaning the customers fall into two segments.

Cluster #	Customers Percentage
1	0.260096
0	0.257310
3	0.138475
2	0.137812
4	0.105801
5	0.100505

Supervised Learning:

Now that we have meaningful data from clustering, we can move on to the next set of datasets: Udacity_MAILOUT_052018_TEST.csv and Udacity_MAILOUT_052018_TRAIN.csv. Supervised learning algorithm will be applied on the training data. Columns that are only related to ID number are excluded; "LNR", "AGER_TYP." At first, I decided to give several supervised learning algorithms a chance. This included: Logistic Regression, Decision Tree Classifier, Random Forest Classifier, XGBoost, and KNeighbors Classifier. However, as I dived further into the problem, I decided to focus on only three algorithms: Logistic regression, random forest classifier and XGBoost. I chose Logistic regression because it is very simple and easy to interpret. It is a binary classifier with 0 and 1, just like the responses in the dataset; and it was very easy to implement. It does not require much tuning or scaling, which made it a good start. It also scored consistently well on all the performance metrics including roc_auc. For the second model, I chose Random Forest classifier because unlike logistic regression, it is not linear. RFC is an ensemble of decision trees, and it works well for imbalanced data (we have imbalanced dataset). I included the XGBoost classifier as the third model, because it is known to perform well in various scenarios and especially on Kaggle competitions. XGBoost was also ideal because it requires least amount of feature engineering (normalizing, scaling) and it works well for large datasets with least impact from outliers.

At first I used all five metrics a try to understand each of my algorithms better. Performance metric used to evaluate each of the algorithms are: accuracy, f1, precision, recall and roc_auc. Graphs for recall and roc_auc performance results are shown below. The graphs for other metrics were also plotted to understand the models better but are not included in the report because they were not suitable for this dataset.

ROC AUC curve score also known as receiver operating characteristic curve (area under the curve) is an ideal metric to plot performance of binary classification model. The area under the curve provides a single score for a classifier model, which helps compare the binary classifications model directly. Since the Arvato dataset is imbalanced, with a lot more people with a negative response than a positive one, roc_auc is an ideal metric to evaluate our models. The higher score of auc_auc curve represents a better performing model. The Kaggle competition also uses roc_auc metric for evaluation, therefore, that is the metric I will use to evaluate the performance of the classification models I use.

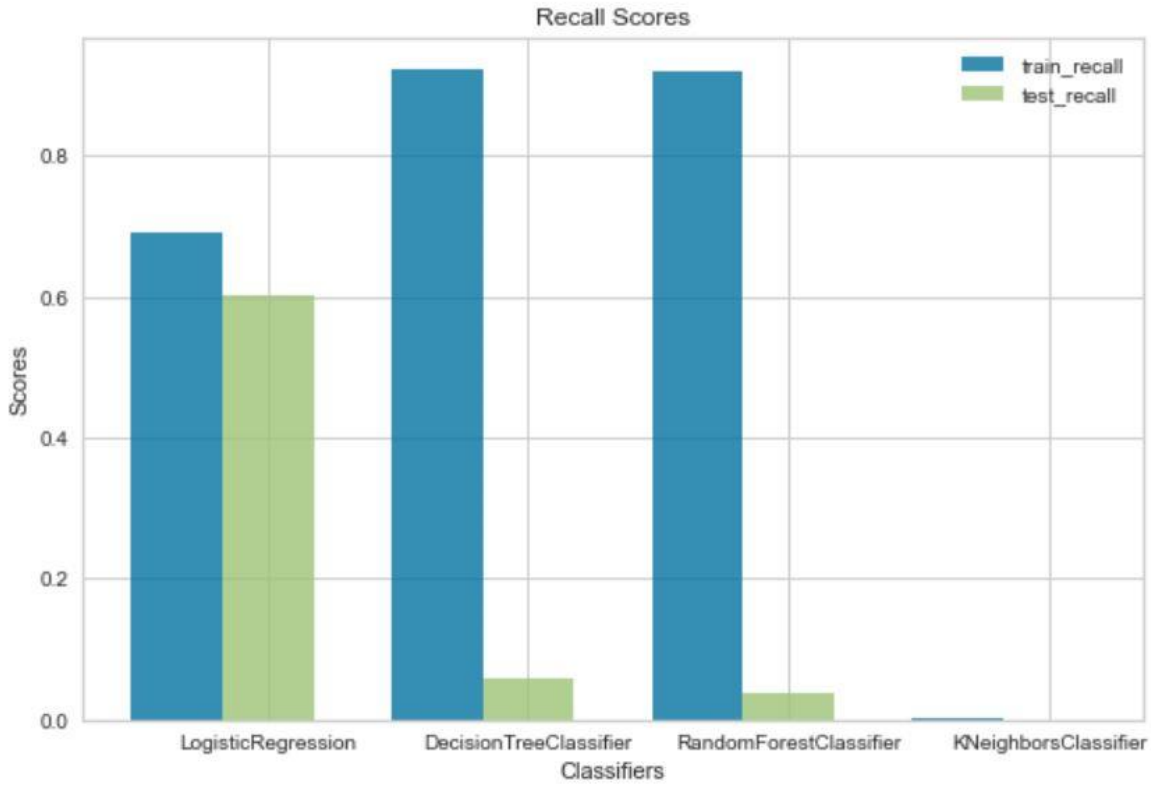

Figure 7: The bar plot shows the performance of recall evaluation metric on different algorithms.

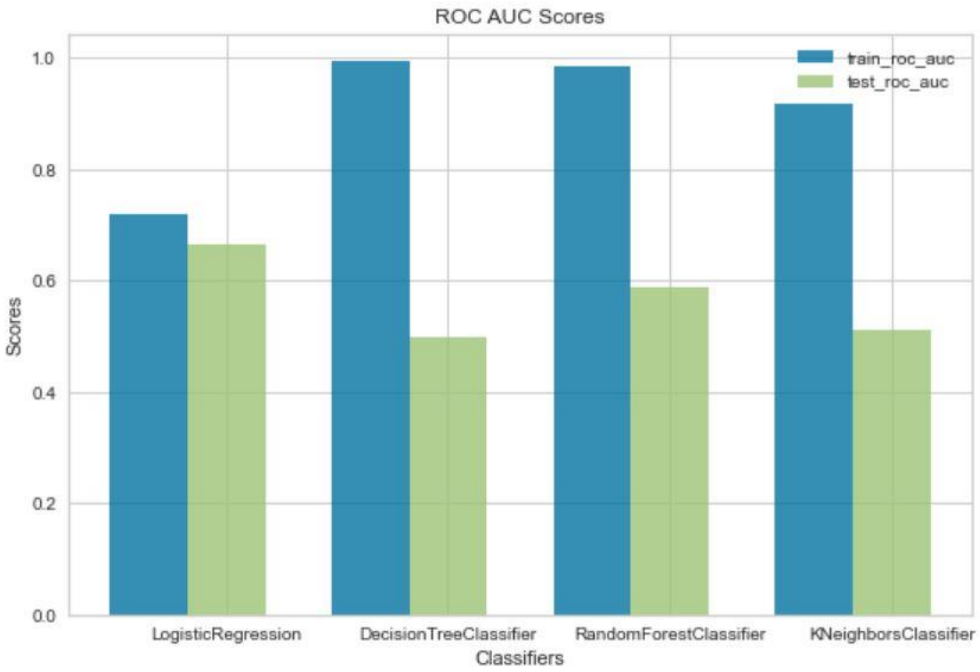

Figure 8: The bar plot shows the performance of ROC AUC evaluation metric on different algorithms.

The following process is like the ones done previously for unsupervised learning. At first, supervised learning is applied without any feature scaling or selections, this will be a benchmark. This is saved as BENCHMARK.csv. Next, feature selection is applied on the scaled data using statistical method. Graph below shows feature scores from the dataset. One attribute "D19_KONSUMTYP_MAX" far outweighs the other. Top 50 features are chosen and the other 147 columns are dropped. It is important to note that there is no one right way of choosing these features. Features selection is estimation; the ones selected are based on the graph.

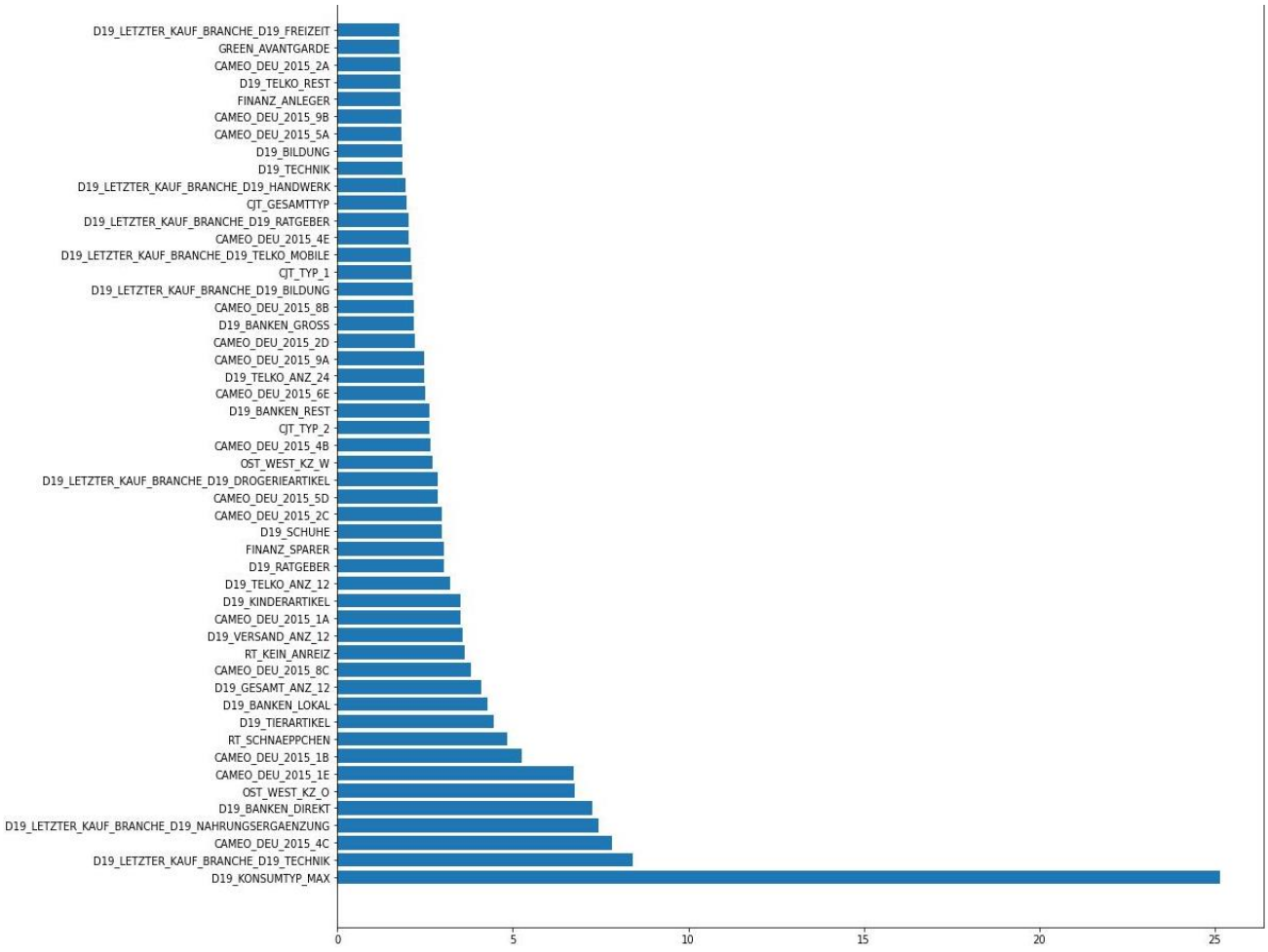

Figure 9: Bar plot shows feature scores from the dataset with D19_KONSUMTYP_MAX outweighing all the other features.

Dimensionality Reduction is performed next; graph below shows the results of reconstruction error. Based on the bar plot below, if a 0.02 reconstruction error is deemed acceptable, then it looks like we need 40 components to achieve dimensionality reduction without introducing too much noise. The threshold used to determine number of components to use is based on heuristics. There are only approximately right answers here. Classifications are applied afterwards.

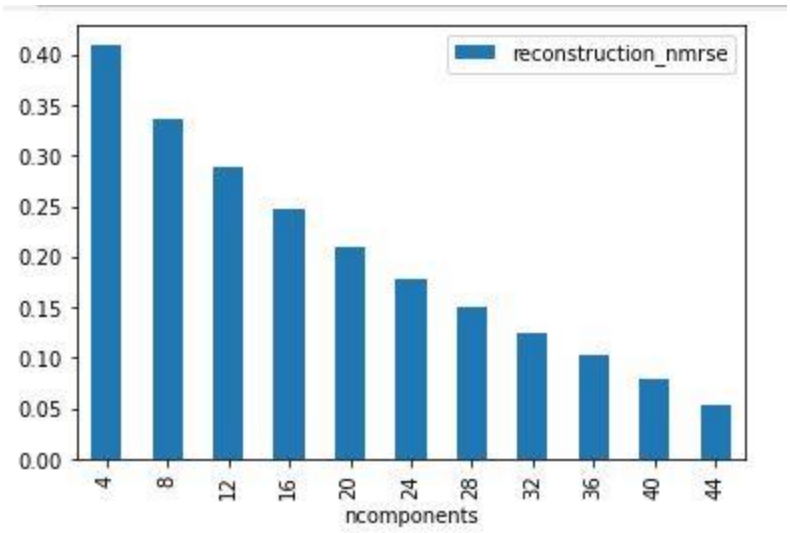

Figure 10: Reconstruction error on different number of components

Refinement:

Hyperparameter tuning is then applied using GridSearchCV. GridSearchCV is applied on three of the best supervised learning algorithms; Logistic Regression, Random Forest Classifier and XGBoost. The results are saved as: logisticresults.csv, rfcresults.csv, and XGBresults.csv.

Results:

The table below shows models with their parameters and scores on Kaggle competition. There are numerous classifiers and models, I focused on three of those: Logistic regression, random forest classifier and XGBoost classifier.

XG Boost Classifier performed the best on Kaggle competition, with a score of about 72%. The parameters used are the following: n_estimators = 100, max_depth=1, verbosity=1, random_state=42, n_jobs=-1). These parameters were chosen by default, there were no feature selection, or scaling done on the data to arrive at this result. The parameters were further tuned to achieve higher score (as shown in the table below); with feature scaling/selection and GridSearchCV. However, that decreased the scores on the Kaggle Competition. XGBoost Classifier is a tree-based model, in such models feature scaling does not make much of a difference. Ensembles of decision tree methods [XG Boost] automatically determines feature importance, and parameters based on the given data. Therefore, further feature selection, scaling, and hyperparameter tuning only lowered the performance for XG Boost. XG Boost was performing at its optimum on default settings.

The second-best performance was by Random Forest Classifier, which is also a method based on set of decision trees. The parameters used were the following: (class_weight='balanced', criterion='entropy', max_depth=4, n_estimators=200, random_state=42, max_features='auto'). Random Forest performed at its optimum after hyperparameter tuning based on GridSearchCV. However, using features selection/scaling along with GridSearchCV decreased the score on the Kaggle competition.

Logistic Regression performed the worst overall, especially at its default (as shown in table below). Logistic Regression performance improved after feature selection/scaling and hyperparameter tuning. Unlike Random Forest Classifier and XG Boost Classifier, Logistic regression is not a tree-based model. Instead, it is linear based and creates one separable line to classify the data.

Models	Parameters	Kaggle Score
Random Forest Benchmark **[Random Forest Classifier with no GridSearchCV or Feature Scaling and Selection]**	RandomForestClassifier(class_weight="balanced", max_depth=1, n_jobs=-1, verbose=1)	0.65235
rfc submission.csv **[Random Forest Classifier with GridSearchCV]**	RandomForestClassifier (class_weight='balanced', criterion='entropy', max_depth=4, n_estimators=200, random_state=42, max_features='auto')	0.70569
rfcresults.csv **[Random Forest Classifier with Feature Scaling and Selection]**	RandomForestClassifier (class_weight="balanced", max_depth=1, n_jobs=-1, verbose=1)	0.68162
GridRandomForestresults.csv **[Random Forest Classifier with GridSearchCV and Feature Scaling and Selection]**	RandomForestClassifier (class_weight='balanced', criterion='entropy', max_depth=4, n_estimators=200, random_state=42)	0.68271
Average		0.6805925

LogisticBENCHMARK.csv **[Logistic Regression with no GridSearchCV or Feature Scaling and Selection]**	LogisticRegression(C=1, class_weight="balanced", n_jobs=-1, max_iter=200, verbose=1)	0.64654
logistic submission.csv **[Logistic Regression with GridSearchCV]**	LogisticRegression (C=1, class_weight='balanced')	0.68208
LogisticRegression logisticresults.csv **[Logistic Regression with Feature Scaling and Selection]**	LogisticRegression(C=1, class_weight="balanced", n_jobs=-1, max_iter=200, verbose=1)	0.66029
LogisticRegression GridLogresults.csv	LogisticRegression(C=0.5, class_weight='balanced')	0.66120

[Logistic Regression with GridSearchCV and Feature Scaling and Selection]		
Average		0.6625275

XGBBENCHMARK.csv **[XGB Classifier with no GridSearchCV or Feature Scaling and Selection]**	XGBClassifier(n_estimators = 100, max_depth=1, verbosity=1, random_state=42, n_jobs=-1)	0.72336
XGBClassifier XGBresults.csv **[XGB Classifier with Feature Scaling and Selection]**	XGBClassifier (n_estimators = 100, max_depth=1, verbosity=1, random_state=42, n_jobs=-1)	0.66689
XGBBoostGridresults.csv **[XGB Classifier with GridSearchCV *and Feature Selection]***	XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.7, gamma=0, gpu_id=-1, importance_type='gain', interaction_constraints='', learning_rate=0.300000012, max_delta_step=0, max_depth=5, min_child_weight=1, missing=nan, monotone_constraints='()', n_estimators=50, n_jobs=0, num_parallel_tree=1, random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=0.7, tree_method='exact', validate_parameters=1, verbosity=None)	0.61602
Average		0.66875666666

The final model and solution are okay but it is not significant enough to have adequately solved the problem. A score of 72% is not a great model and it has lots of room for improvement. I think to solve the problem adequately, so the company can channel their resources based on the results, the score should at least be of 90%. I will continue to work on this model further by experimenting with under sampling, over sampling and SMOTE techniques.

Conclusion:

Reflection:

In this project, a large dataset was given to determine a segment of population that are most likely to become customers to the mail-order company. The two most interesting and challenging parts of the project was cleaning the data to decrease noise and determining which model works best for the problem.

I learned that Data cleaning is very crucial step, and it requires a lot of thinking to determine what data is worth keeping. It is important to get rid of the noise, but it is also very important to retain valuable. That was a bit of struggle for me, keeping the balance is the key.

The second challenging part was analyzing each model and determining which one performs the best and why. There are large number of models to choose from, but its counterproductive to try using all of them. It is not only important to choose the optimum model, but to also choose the optimum parameters for the models. Each model has its own set of advantages and disadvantages, there is no one correct answer, but decisions must be based on proper results and analysis.

Improvement:

There are lots of ways to improve the machine learning models in this project. I focused on three models, but there are many more models to experiment with, and endless numbers of parameters that can be tuned further to improve the data. I applied a lot of hyperparameter tuning and feature selections myself, but there are so many ways they can be refined and changed to improve model performance. However, it is important to note that constant attempts at hyperparameter tuning will not always improve the performance of the model.

References:

- Arvato Financial Solutions. 2020. *Arvato Financial Solutions*. [online] Available at: <https://finance.arvato.com/en-us/> [Accessed 11 November 2020].

- James M. Curran, Sajeev Varki, Deborah E. Rosen. (2010) Loyalty and Its Antecedents: Are the Relationships Static?. *Journal of Relationship Marketing* 9:4, pages 179-199. https://www.tandfonline.com/doi/citedby/10.1080/15332660902991197?scroll=top&needAccess=true

- Talabis, Mark Ryan M., et al. “Analytics Defined.” *Information Security Analytics*, Syngress, 5 Dec. 2014, www.sciencedirect.com/science/article/pii/B9780128002070000010.