

Project Report

Customer Segmentation & Marketing

Strategy:

Enhancing Acquisition and Retention for Arvato-Bertelsmann

By Satyajit Ovelil

Data Scientist Nanodegree

The School of Data Science

Table of Contents

Table of Contents	2
Project Definition	3
Project Overview	3
Problem Statement	3
Metrics	4
Analysis	4
Methodology	7
Data Preprocessing	7
Feature Engineering	7
Dimensionality Reduction	8
Implementation	8
Data Processing	8
Cluster Analysis	8
Customer Segmentation	9
Supervised Learning	9
Refinement	10
Results	10
Model Evaluation and Validation	10
Justification	11
Customer Segmentation	11
Supervised Learning	12
Conclusion	12
Reflection	12
Improvement	12
References	13
Appendix	15

Project Definition

Project Overview

The project aims to analyse the demographics data of a mail-order sales company in Germany and compare it against demographics information for the general population to identify the core customer base of the company.

The data has been provided by Bertelsmann Arvato Analytics and consists of four files:

"Udacity_AZDIAS_052018.csv," "Udacity_CUSTOMERS_052018.csv,"

"Udacity_MAILOUT_052018_TRAIN.csv," and "Udacity_MAILOUT_052018_TEST.csv."

The project will involve the use of unsupervised learning techniques on the first two datasets to perform customer segmentation and identify the parts of the population that best describe the core customer base of the company. Then, the same analysis along with the application of supervised learning algorithms on the third and fourth datasets to predict which individuals are most likely to convert into becoming customers for the company.

Two Excel spreadsheets "DIAS Information Levels - Attributes 2017.xlsx" and "DIAS Attributes - Values 2017.xlsx" have been provided to aid in data preprocessing and understanding, containing attribute descriptions and data value mappings. The spreadsheets contain attribute descriptions and data value mapping for each feature.

Problem Statement

The problem at hand is to improve the mail-order sales company's marketing campaign by converting more individuals into customers. The company needs to analyse its customer base's demographics data and compare it with the general population's demographics data to identify potential customers and target them effectively.

To achieve this, the first step is to clean the datasets and re-encode certain features by referring to an attributes excel file. Rows or columns with missing information will be dropped, and new features will be created using metadata.

Since the datasets have a large number of dimensions, which can lead to noise in the clustering algorithm, the next step is to perform dimensionality reduction. This will capture all relevant information and improve the performance of the clustering algorithm. To do this, the data will be scaled and an imputer will be used to deal with remaining missing values. Unsupervised learning techniques will be used to perform customer segmentation and identify individuals who share demographic characteristics with the company's core customer base.

To predict the successful conversion of individuals into customers by the mail-order company, ensemble classification algorithms will be used. Feature engineering techniques will also be used to incorporate information extracted from the customer segmentation analysis and metadata file.

The expected outcome of the customer segmentation task will be a clustering model that identifies clusters of the general population that find a large representation in its own customer base. The classification models are expected to predict individuals more likely to convert by using the above mentioned techniques.

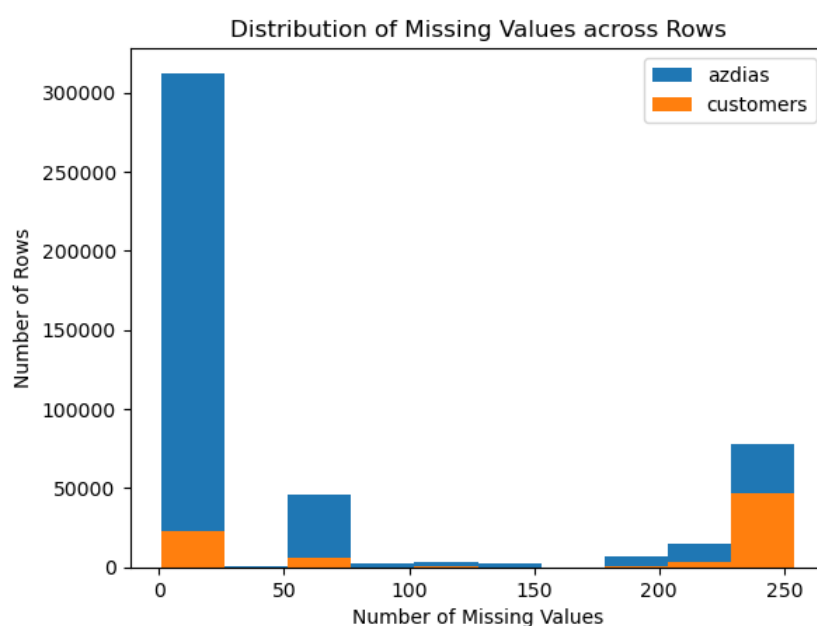
By focusing on individuals with a high likelihood of converting, the efficiency of the marketing campaign can be improved, ultimately leading to more sales for the company.

Metrics

In the unsupervised learning task, the clustering algorithm will be evaluated using the within-cluster sum of squares (WCSS) metric and the dimensionality reduction technique will be evaluated using the explained variance ratio. The aim is to have a low WCSS score and to keep the number of principal components that explain at least 80-90% of the total variance. In the supervised learning task, the performance of the predictive model will be evaluated using the receiver operating characteristic (ROC) curve and the area under the curve (AUC) score. The goal is to achieve a high AUC score to indicate an effective model for predicting potential customers for the mail-order company.

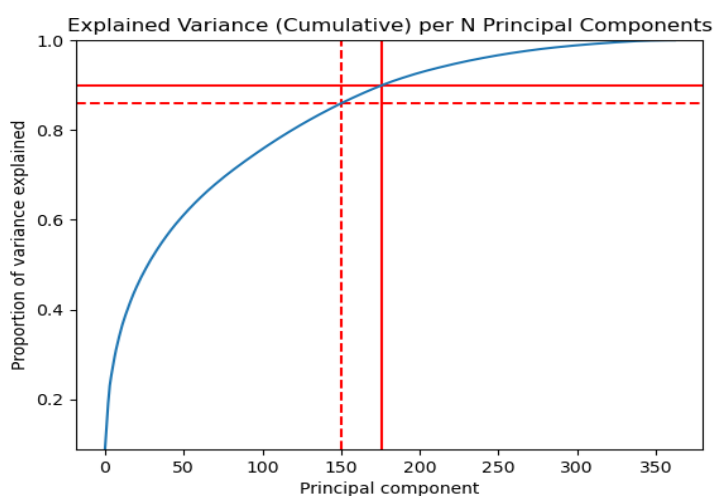
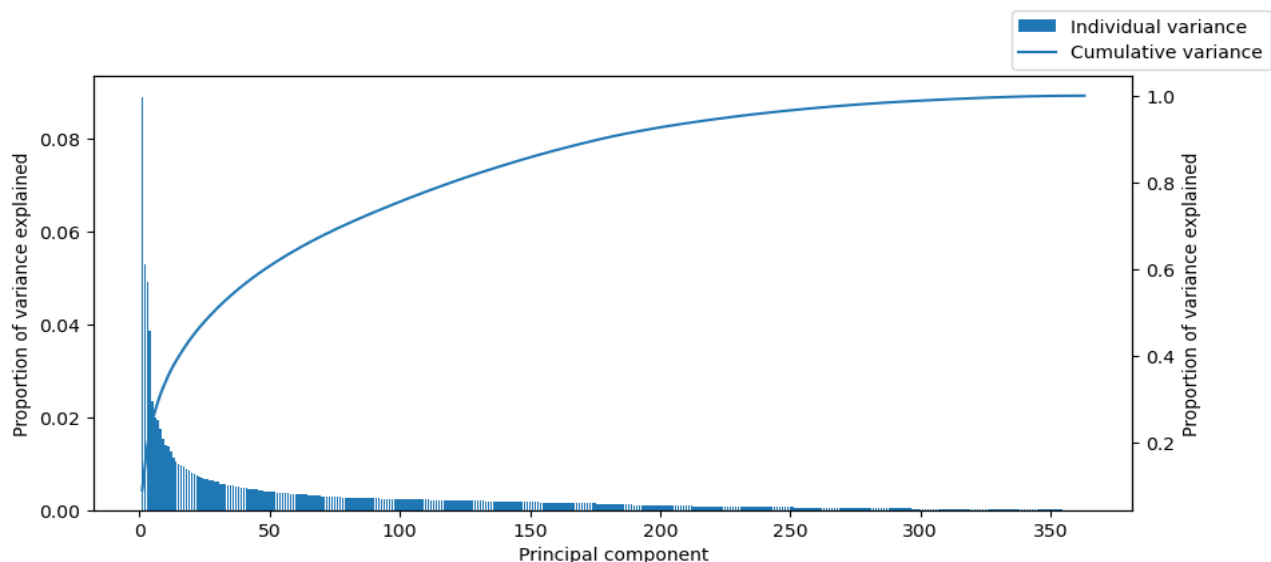
Analysis

The AZDIAS and CUSTOMERS datasets were loaded, and warnings were encountered due to mixed data types in columns 'CAMEO_DEUG_2015' and 'CAMEO_INTL_2015'. The metadata file was consulted, and it was discovered that there were ambiguous values represented by 'X' and 'XX' in these columns. The metadata file also provided the column-values mapping used in order to replace or re-encode values that represented missing information in the data. Categorical features were identified using various approaches, and some columns needed further transformation or contained information that could be made into a different column. String encodings for binary categories were replaced, and columns with ambiguous information were replaced with NaNs (Not a Number). It was found that certain columns had values used to code ambiguous information for an observation.



The bar plot shows the distribution of missing values across columns: There are quite a few columns with more than 70% of the information missing; this means over for these particular columns they have missing values for over 70% of the samples/rows.

The explained variance ratio can help decide the number of components to keep after performing PCA (Principal Component Analysis). It provides a measure of how much information each principal component retains from the original data. The scree plot is a graph that plots the cumulative sum of the explained variance ratio against the number of components. It can be used to visually determine the number of components that explain a significant amount of the total variance in the data.

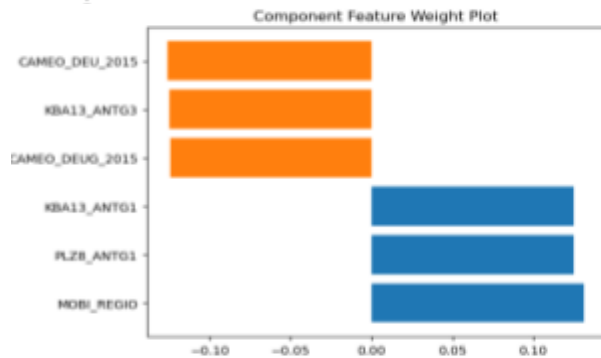


The dotted red lines show the number of components that explain ~85 % of the variance whereas the red lines denote the number of components that explain 90 % of the variance. 90% of variance is explained by around 176 components and around 86 % of variance is explained by 150 components. The model with 176 components was finally chosen for analysis.

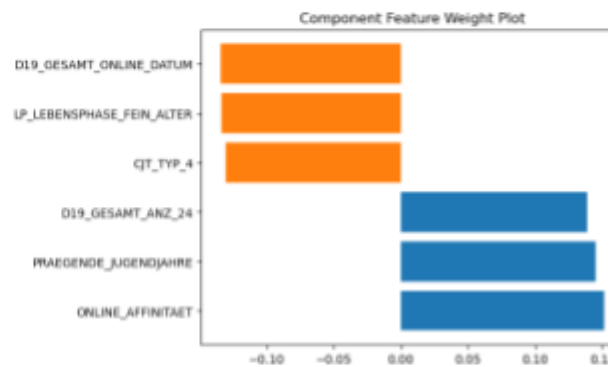
A good idea to understand the contribution of different features towards the variation in the data is to plot the feature weights. Plotting the feature weights for different components of a PCA model can provide a useful tool for understanding the structure of the data and selecting which features to retain for subsequent analysis.

The feature weights assigned by the top 4 components have been displayed below. The weights with high positive as well as negative values have been chosen. Note that these 4 components rank the highest in terms of explained variance.

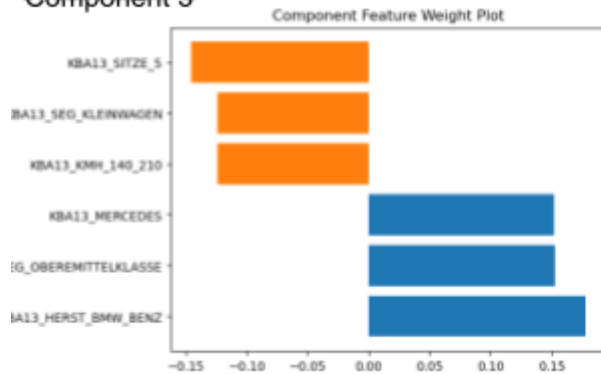
Component 1



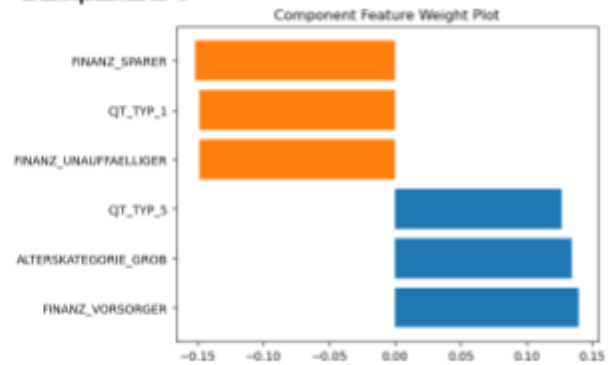
Component 2



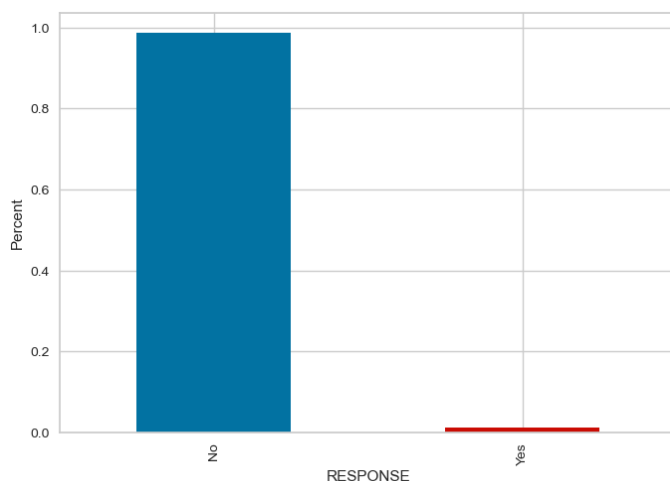
Component 3



Component 4



For the supervised learning task the training dataset was quite imbalanced. There's no way of knowing whether the proportion of imbalance is preserved in the test set as well since the target variables have not been made available for the test set. Also, note that the train and test split ratio is 50%.



Methodology

Data Preprocessing

The first file, "Udacity_AZDIAS_052018.csv," contains demographics data for the general population of Germany and has 891,211 persons (rows) and 366 features (columns). The second file, "Udacity_CUSTOMERS_052018.csv," contains demographics data for customers of the mail-order company and has 191,652 persons (rows) and 369 features (columns). The third and fourth files, "Udacity_MAILOUT_052018_TRAIN.csv" and "Udacity_MAILOUT_052018_TEST.csv," contain demographics data for individuals who were targets of a marketing campaign and have 42,982 persons (rows) and 42,833 persons (rows), respectively.

The "CUSTOMERS" file contains three extra columns ("CUSTOMER_GROUP," "ONLINE_PURCHASE," and "PRODUCT_GROUP"), which provide broad information about the customers depicted in the file.

The data cleaning steps mentioned in the preliminary analysis were followed. This included replacing mixed data types, converting categorical variables, imputing missing values, and creating new features based on socio-economic and lifestyle information. To ensure the columns have the correct data types, the ambiguous values in 'cameo' columns were replaced with NaNs, and the strings were converted to integers. A dictionary was created using the metadata files that provide attribute descriptions and data value mappings. This dictionary had columns/attributes as keys and another dictionary mapping the unknowns for that attribute to NaNs as values.

Finally, the rows and columns that had more than 25-30% missing information, meaning that the rows or columns contained at least ~70-75 non-na values were dropped. Any remaining missing values were imputed using medians.

The application of PCA required that the data be scaled lest any feature be weighted unfairly. This concluded the data preprocessing and preparation required for the application of KMeans (Lloyd, 1982) used in order to cluster or segment the data.

Feature Engineering

New features are created using information learned in previous sections, with a focus on columns/attributes that contain categorical information, have high cardinality, and include extra encodings not mentioned in the attributes file. These columns mainly pertain to socio-economic information, lifestyle, and class.

Upon observation of the lifestyle and class columns with high cardinality, it is noted that they contain considerable information that appears to represent ordinal information about various subclasses/categories grouped together. As a result, a decision is made to re-encode these columns and generate new columns that capture these ordinal scales and sub-classifications. For instance, new columns are created to depict the range of income class, family type, age groups, etc for each customer, based on the existing columns.

The next step involves dropping columns that have become redundant in terms of information. Columns that represent metadata, like the 'LNR' column that represents the ID

for each observation (customer), and the 'EINGEFUEGT_AM' column that literally translates to 'inserted on' will also be dropped.

Dimensionality Reduction

The preprocessing for this involved imputing any missing values and scaling the data. PCA was chosen to reduce the dimensionality of the data while preserving important information. The "AZDIAS" dataset was then subjected to PCA to reduce its dimensions. It was necessary to keep components that explain variance greater than ~85-90%, which was achieved by selecting the explained variance ratio for n components.

Implementation

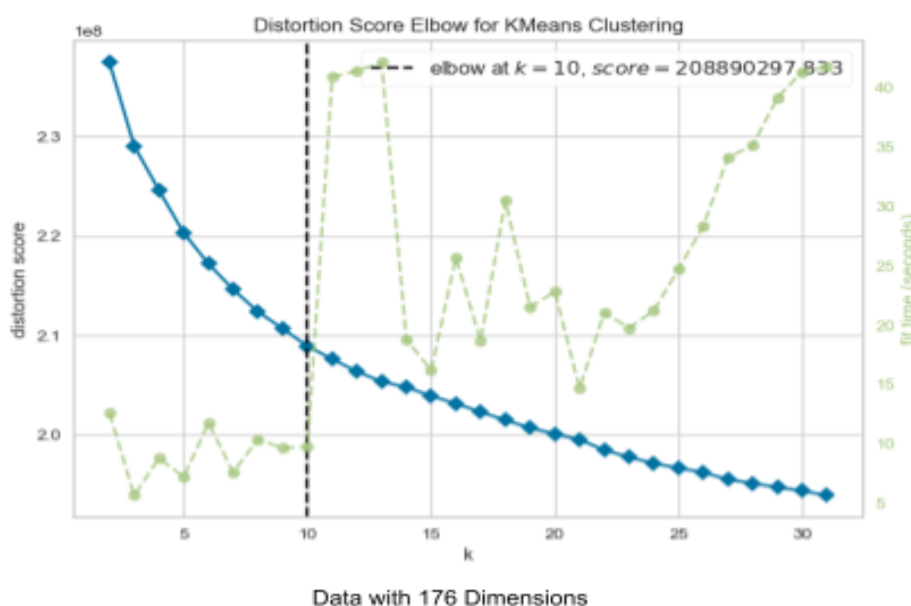
Data Processing

The data cleaning and preliminary feature engineering discussed in the previous sections was done by using custom python classes that we wrote. Since we primarily used dictionary mappings or functions to replace, re-encode or create new columns we wrote python classes that helped track and store all the dictionaries and functions used to process or transform the data. These also helped apply the same processes or transformations to another dataset in the same order.

Data preprocessing for PCA involves using scikit-learn's (*Scikit-Learn: Machine Learning in Python*, n.d.) SimpleImputer to impute missing values and StandardScaler to scale the datasets. Performing PCA on the azdias dataset, we created two PCA models with 176 and 150 components.

Cluster Analysis

The elbow graph was used to determine the optimal number of clusters for KMeans clustering on the dataset with 176 dimensions. It was found that 10 clusters were optimal using the elbow method and the KMeans clustering algorithm was implemented with 10 clusters.

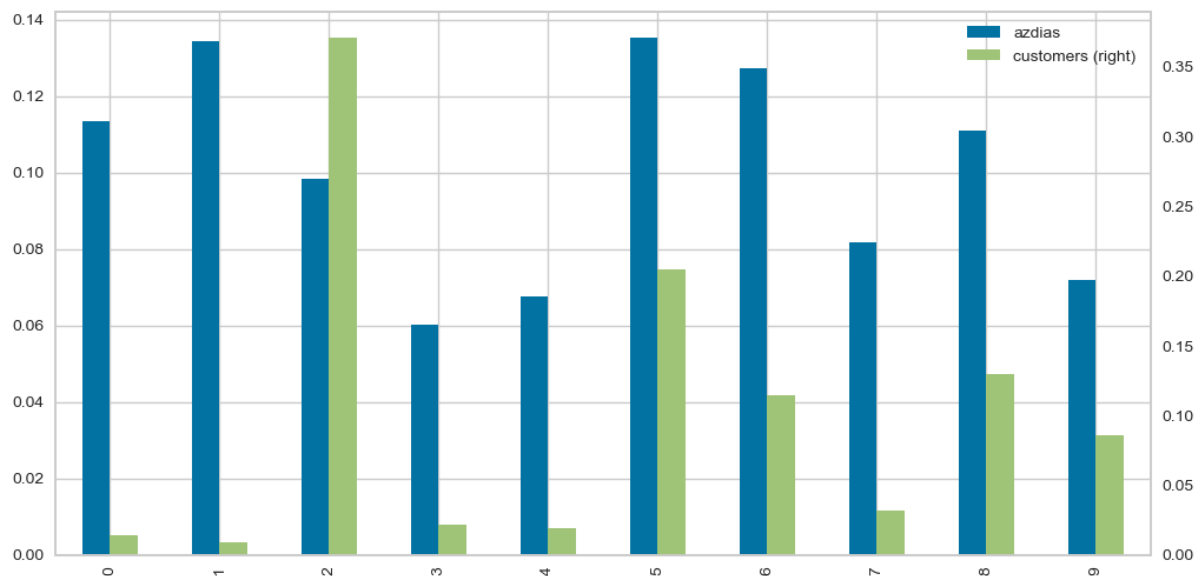


The elbow graph was created using yellowbrick's KElbowVisualizer (Bengfort et al., 2018), while either Pandas (The pandas development team, 2023) or Matplotlib (Bengfort et al., 2018; Hunter, 2007) was used for all other kinds of graphs.

Additional data cleaning and feature engineering were performed to improve results.

Customer Segmentation

The KMeans algorithm was implemented for the datasets with reduced dimensions obtained using PCA. Scikit-learn's KMeans implementation with 'kmeans++' initialization (Gabow, 2007) was used to predict the cluster for each sample/observation in both our datasets. The KMeans clustering algorithm yielded the following results once applied to the datasets with 176 components:



Supervised Learning

For the supervised learning part, the training dataset was read in and cleaned using a class developed for cleaning. Preliminary feature engineering was performed using a feature engineer class, and a grid search was used to find the best parameters for three different classifiers: XGBoost (Chen & Guestrin, 2016), LightGBM (Machado et al., 2019), and Random Forest (Breiman, 2001). SMOTE (Chawla et al., 2002) was used to oversample the minority class. The implementation for SMOTE was done using the imbalanced-learn python module (Lemaître et al., 2017).

The dataset was split into features and labels, and missing values were imputed. New features were created by scaling a copy of the data, using the PCA & KMeans models to predict clusters for each observation, and using this predicted cluster information to create a new feature. One-hot encoding was also applied to certain columns, and scale_pos_weight was calculated to handle class imbalance. Grid search was then performed for all models, with SMOTE implemented for each. Models were trained with the best parameters obtained from the grid search on the entire train set, and the probability of labels was predicted. ROC AUC scores were calculated for each model, with a dataframe created to compare results. The test dataset was then read in, cleaned, and features were created. The preprocessing and data preparation steps performed on the training dataset were performed on the test set, and predictions were made using the trained models to create CSV files for Kaggle submission to evaluate the performance of the models.

Refinement

The unsupervised learning task involved Customer segmentation using the KMeans clustering algorithm. To improve performance, dimensionality reduction using PCA was considered due to the size of the dataset. PCA helps in speeding up computation, reducing noise, and avoiding overfitting for high dimensional data that is often sparse.

The number of components to keep after performing PCA was chosen by looking at the explained variance ratio as explained in the methodology section.

The elbow method was used to select the optimal number of clusters for KMeans algorithm by plotting the within sample sum of squares for different k values. The yellowbrick's KElbowVisualizer was used to determine the elbow point as the graph was not easy to visually discern. Another option was to inspect the silhouette scores for the clusters with different k values.

For the supervised learning task, we had to deal with a dataset that was fairly similar to the datasets used in the customer segmentation tasks when it came to the variables that were provided. Hence re-applying the cleaning and feature engineering and other data pre-processing was quite straight-forward and easy.

In the feature engineering stage, two types of features were added. The first type involved one-hot encoding the columns that represented class or type from the attributes excel, after removing the ones with high cardinality. The second type of features were created by utilizing the information from KMeans clustering. This was accomplished by using the trained KMeans models to predict the cluster for each observation in the training dataset, and then saving the predicted cluster for each sample in new column(s).

The dataset used in this project was found to be imbalanced with respect to the target classes. Two methods were used to address this issue: hyperparameters provided by the algorithms implemented such as `scale_pos_weight`, and sampling techniques such as SMOTE in the modelling pipeline.

Results

Model Evaluation and Validation

GridSearchCV was used to cross-validate the models. The cross-validation used 3 folds, and the hyperparameters were tuned for each model based on its performance. Since each algorithm deals with class imbalances a little differently, the relevant hyperparameters were also tuned. Each algorithm was also cross-validated using a pipeline that involved SMOTE before.

The best scores and associated parameters selected by GridSearchCV for each model are given in the image below::









	model	best_params	score
0	xgboost	{'learning_rate': 0.1, 'max_depth': 3, 'scale_pos_weight': None}	0.764681
1	lightgbm	{'boosting_type': 'dart', 'is_unbalance': None, 'max_depth': 50}	0.758657
2	randomforest	{'max_depth': 10, 'n_estimators': 900}	0.666860
3	xgboost with smote	{'classifier__learning_rate': 0.01, 'classifier__max_depth': 3}	0.744482
4	lightgbm with smote	{'classifier__boosting_type': 'gbdt', 'classifier__max_depth': 50}	0.689835
5	randomforest with smote	{'classifier__max_depth': 90, 'classifier__n_estimators': 900}	0.637198

The final models were trained using the best parameters obtained above on the entire training set. The results of these models in terms of the roc_auc score is given below:

	model	training score
0	xgboost	0.878150
1	lightgbm	0.930149
2	randomforest	0.955797
3	xgboost with smote	0.769509
4	lightgbm with smote	0.897398
5	randomforest with smote	0.991983

The models clearly perform quite well on the training set. But this in itself is not a clear picture of the performance. It's expected that the performance would dip slightly after sampling considering that we are generating synthetic samples. In some cases the models might have overfit.

Here are the results evaluated against the test set:

Submission and Description	Private Score 	Public Score 
 arvato_kaggle_submission_xgb.csv Complete (after deadline) · now	0.76026	0.79461
 arvato_kaggle_submission_light.csv Complete (after deadline) · 29s ago	0.7413	0.79007
 arvato_kaggle_submission_forest.csv Complete (after deadline) · 1m ago	0.6746	0.71289
 arvato_kaggle_submission_xgb_imb.csv Complete (after deadline) · 1m ago	0.74495	0.78326
 arvato_kaggle_submission_light_imb.csv Complete (after deadline) · 2m ago	0.68156	0.69981
 arvato_kaggle_submission_forest_imb.csv Complete (after deadline) · 3m ago	0.62758	0.64471

Justification

Customer Segmentation

Application of KMeans clustering after Principal Component Analysis (PCA) can be beneficial in several ways (Ding & He, 2004). Firstly, it can speed up computation as the computational cost of clustering increases exponentially with the number of dimensions in a dataset. By reducing the number of dimensions, PCA can lower the computational cost of clustering. Secondly, PCA can improve clustering results by eliminating noise and redundant

information in high-dimensional datasets, which can lead to poor clustering results. Finally, PCA can help avoid overfitting, which can result in poor generalisation performance in high-dimensional datasets. By reducing the number of dimensions, PCA can lower the risk of overfitting and improve the generalisation performance of the clustering algorithm.

Supervised Learning

The project found that SMOTE may not be helpful for all algorithms and cases, but it did improve the results for the RandomForest algorithm. Feature engineering played a significant role in improving the model's performance, with preliminary feature engineering involving re-encoding and creating new features, and later feature engineering using KMeans. Ensemble models were used for their reliability, and the team may still make a late submission to the Kaggle competition to evaluate their performance.

Conclusion

Reflection

The initial phase of the project involved analysing the provided datasets, which shared most of the columns or variables in common. However, the data cleaning process proved challenging due to the significant amount of noise in the dataset, which could not be resolved using the standard preprocessing methods provided by scikit-learn. Additionally, feature engineering required extraction of information from the attributes metadata file. For the customer segmentation task, an unsupervised learning model was applied using the KMeans clustering algorithm. To handle the high dimensionality of the data, Principal Component Analysis (PCA) was used, and the data was preprocessed by imputing missing values and scaling. For the supervised learning problem, ensemble models like XGBoost were applied, utilising the cleaning and feature engineering done earlier, as well as some additional feature engineering and information extracted by the unsupervised learning techniques. The final model achieved a good ranking in both the public and private leaderboards, demonstrating the effectiveness of the approach taken.

Improvement

To improve customer segmentation, additional features can be re-encoded and the optimal number of clusters for KMeans segmentation can be determined by examining the silhouette score. Imbalanced algorithm pipelines could also be better implemented, and feature engineering could be improved by inspecting the mutual information score. Despite limitations, the results are sufficient for profitable business decisions. Further iterations and trials of the marketing campaign can lead to more growth. Further iterations and trials of the marketing campaign would take the campaign a long way, leading to more growth.

References

1. Bengfort, B., Danielsen, N., Bilbro, R., Gray, L., McIntyre, K., Richardson, G., Miller, T., Mayfield, G., Schafer, P., & Keung, J. (2018). *Yellowbrick v0.6*. Zenodo.
<https://doi.org/10.5281/ZENODO.1206264>
2. Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32.
3. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *The Journal of Artificial Intelligence Research*, 16, 321–357.
4. Chen, T., & Guestrin, C. (2016, August 13). XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco California USA.
<https://doi.org/10.1145/2939672.2939785>
5. Ding, C., & He, X. (2004). K-means clustering via principal component analysis. *Twenty-First International Conference on Machine Learning - ICML '04*. Twenty-first international conference, Banff, Alberta, Canada.
<https://doi.org/10.1145/1015330.1015408>
6. Gabow, H. (2007). *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics.
7. Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3), 90–95.
8. Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research: JMLR*, 18(17), 1–5.
9. Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory / Professional Technical Group on Information Theory*, 28(2), 129–137.

10. Machado, M. R., Karray, S., & de Sousa, I. T. (2019, August). LightGBM: An effective decision tree gradient boosting method to predict customer loyalty in the finance industry. *2019 14th International Conference on Computer Science & Education (ICCSE)*. 2019 14th International Conference on Computer Science & Education (ICCSE), Toronto, ON, Canada. <https://doi.org/10.1109/iccse.2019.8845529>
11. *Scikit-learn: Machine Learning in Python*. (n.d.). Paperpile. Retrieved May 2, 2023, from <https://paperpile.com/app/p/05fb3c3a-c9b3-04b8-8ce7-d711cf9faf33>
12. The pandas development team. (2023). *pandas-dev/pandas: Pandas*. Zenodo. <https://doi.org/10.5281/ZENODO.3509134>

Appendix

1. The competition was hosted here: [Udacity+Arvato: Identify Customer Segments | Kaggle](#).
2. For helper functions used in the project: [source code](#).
3. The [Project Notebook](#) contains end-to-end Project Implementation.
4. [README](#) contains information on Project Motivation, modules used and a snapshot of Results.
5. This Project has been submitted towards the competition of [The Data Scientist Nanodegree](#).
6. More information on what a Nanodegree is can be found here: [Nanodegree 101: What is a Nanodegree Program?](#)
7. Terms & Conditions for the project can be found [here](#).
8. All images used in the above paper can be found either in the project workbook or in the ['figures directory'](#).
9. [Project Repository](#).