

Machine Learning Engineer Nanodegree: Capstone Report

Arvato Solutions: Customer Targeting

Author: Mike Griffin

Date: 12th June 2020

CONTENTS

- 1) Domain background
- 2) Problem statement
- 3) Datasets & inputs
- 4) Solution statement
- 5) Benchmarking
- 6) Evaluation metrics
- 7) Code setup
- 8) Report - customer segmentation
 - a. Data processing & feature engineering
 - b. Principal component analysis
 - c. Clustering analysis
- 9) Report - customer acquisition
 - a. Benchmark modelling
 - b. Model training and selection
 - c. Model results
 - d. Kaggle submission
 - e. Model interpretation
- 10) Summary – conclusions and further work
- 11) References

DOMAIN BACKGROUND

This is a Udacity capstone project representing a real-life data science problem using information provided by Betrelsmann Arvato Analytics.

Arvato is a global services company that develops and implements solutions for business customers in a diverse range of industries, with a focus on innovative data analysis and automation. For this project, Arvato provided services to a German mail-order sales company aiming to understand demographics and improve customer acquisition. This project will explore unsupervised clustering to identify customer groups as well as classification modelling to predict which individuals will respond to campaigns.

Ultimately this work was designed allow the client to improve client targeting for future campaigns. My analysis is for exploration and learning only.

PROBLEM STATEMENT

The overarching objective is to identify which individuals are most likely to respond to a marketing campaign and become customers of the client. This is a binary classification problem using individual-level attributes to estimate the likelihood of a customer response. Specifically, the project will aim to maximise the ROC score on a held-out test set on Kaggle.

DATASETS AND INPUTS

Demographics information has been provided for both the general population at large as well as for prior customers of the mail-order company, stored at the level of an individual.

- 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; 191 652 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).

365 fields are common across the dataset with information on the fields and labels stored in two data dictionaries:

- DIAS Information Levels - Attributes 2017.xlsx: high-level descriptions of attributes
- DIAS Attributes - Values 2017.xlsx: detailed mappings of levels within fields

SOLUTION STATEMENT

The solution will have two components:

- 1) **Customer segmentation**— a report which will separate the population in to distinct clusters and explore the variation in attributes by clusters. This will require appropriate imputation, feature engineering, dimensionality reduction and clustering via k-means. This will allow comparison between the population and customer data to explore which clusters are over/under-represented in the customer set.
- 2) **Customer acquisition** - a set of models will be trained and compared on the binary classification of whether an individual will respond to a campaign. Segment information from (1) will be computed and fed in to the classification model. Model performance will be assessed and tuned using train/validation splits before measurement on a held-out test set on Kaggle.

BENCHMARKING

A simple decision tree will be trained to determine baseline performance, with constraints on leaf size or depth to control overfitting. Baseline performance will be assessed using the ROC AUC score and confusion matrices. This will enable clear interpretation of important metrics, thresholds and interactions which may be used to engineer features with predictive value for the main modelling.

EVALUATION METRICS

For the first stage involving customer segmentation:

- Explained variance will be used to assess the effectiveness of dimensionality reduction.
- For clustering, the “elbow method” will be used to assess the optimal number of clusters using the inertia metric to minimise cluster distances.

For the second stage involving binary classification:

- In line with the Kaggle competition, the AUC ROC score will be the main measure to assess model performance, supplemented by analysis of confusion matrices.

CODE SETUP

The analysis is performed in two workbooks:

- Majority of data processing and modelling is performed in the Udacity workspace, downloaded as notebook named “Aravato Project Workbook.ipynb”
- Basic model interpretation is performed in Google Colaboratory in order to utilise the shap and pdpbox packages. This is in a file named “model_interpretation.ipynb”

1) REPORT- CUSTOMER SEGMENTATION

a) Data processing and feature engineering

Examination of the target variable (“RESPONSE”) within the “mailout_train” set highlights that the classification problem has highly imbalanced classes with successful responses in 1.3% of cases. This means over/under sampling will be required to achieve good results.

Basic stats on the datasets are collated below – fields are shared across most of the datasets. The customer dataset is c20% the size of the population dataset and the classification train and test sets are of similar size.

Input datasets:

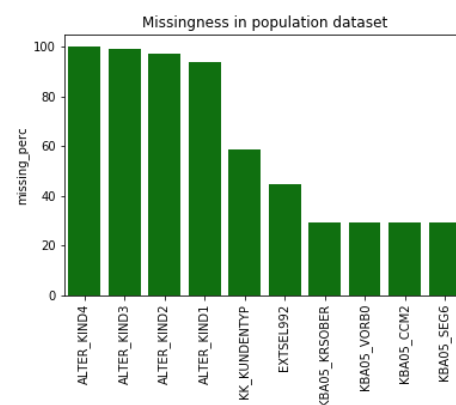
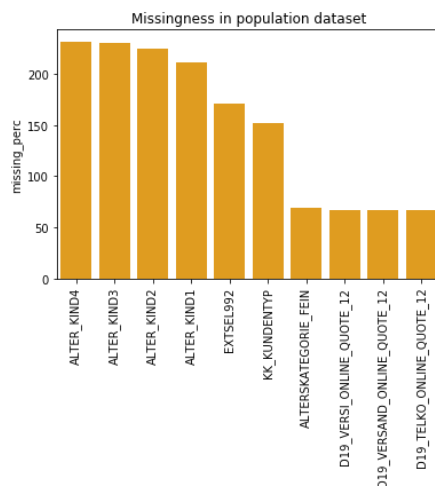
Dataset	Float columns	Integer columns	Objects	Rows	Size
Azdias	267	93	6	891,221	2.4GB
customers	267	94 (+??)	6	191,652	540MB
mailout_train	267	94 (+“RESPONSE”)	6	42,962	120MB
mailout_test	267	93	6	42,832	120MB

This highlights here are six categorical variables requiring specific treatment:

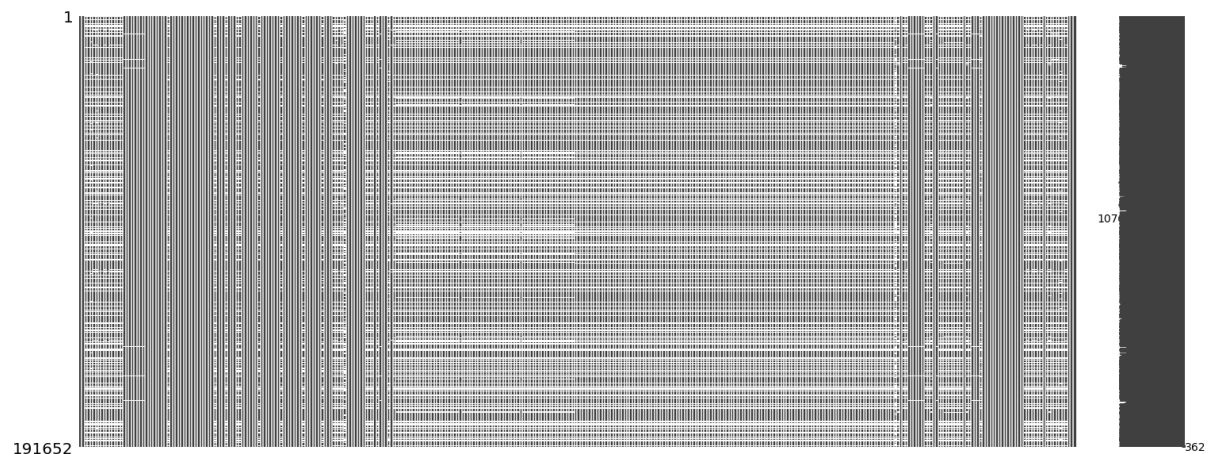
- EINGEFUEGT_AM – this is a datetime field from which the year, month and day is extracted as separate features and the datetime feed is dropped.
- CAMEO_DEUG_2015 and CAMEO_INTL_2015 - numerical 0 used to replace X or XX and all values converted to floats.
- CAMEO_DEU_2015 and D19_LETZTER_KAUF_BRANCHE – one hot-encoded, note that all dummies are included for easy interpretation with tree models although this will lead to collinearity with linear models.
- OST_WEST_KZ – “W” or “E” binary encoded with gaps as -1.

No fields were removed to allow for the possibility all feeds offer predictive power (inc “LNR”). Using the data dictionaries reveals that some field descriptions are missing so I apply common sense where possible. The large population dataset was randomly sampled by row (to include 50% data with a fixed seed) to improve computation times.

Exploring patterns of top missingness suggests this is similar between the population and customer datasets



Using the missingno library visually highlights how the missingness uniform across sets of fields suggesting these originate from consistent feeds. Imputation functions were used to replace gaps with median values, add binary flags for missingness and then compress these flags across data feed sets. This imputation approach means that the distributions of the underlying fields are unchanged and the missingness itself can be explored for predictive power. From end-to-end the data processing creates a dataset of 457 variables from the original set of 366 all of which are numerical and complete.



Examination of the detailed field information also reveals the values of -1,0 or 9 often encode “unknown”. The option to treat these in the same manner as blanks (ie replaced by median values) was also explored, by removing these values in advance of general gap-filling.

Data pipelines:

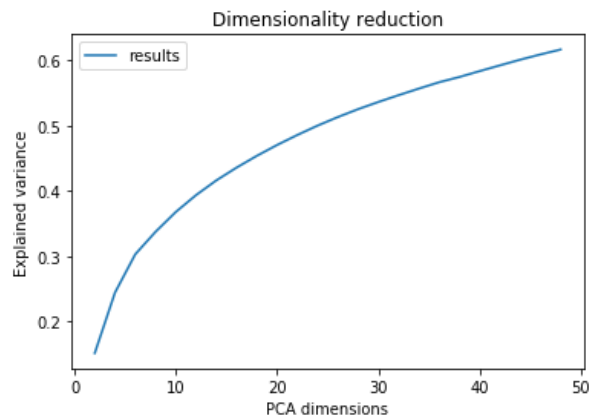
For the unsupervised and supervised tasks, two pipelines were designed using common functions where appropriate.

Step	Data prep	Approach	Unsupervised - PCA and clustering	Supervised - Binary Classification
1	Add binary flags for nas	Write as function		Y
2	Add date variables	Process_dates function to extract month, year	Y	Y
3	Categorical and gap-filling	Process_df function to impute median values and add encoding	Y	Y
4	Consolidate nas	clean_nas module to consolidate na flags across systematic data gaps		Y
5	Rescaling	Standard scaler trained on population dataset	Y	
6	Principal component analysis	Via sklearn PCA	Y	
6	Clustering	Via sklearn k-means	Y	
7	Classification modelling	Sklearn tree models and logistic regression		Y

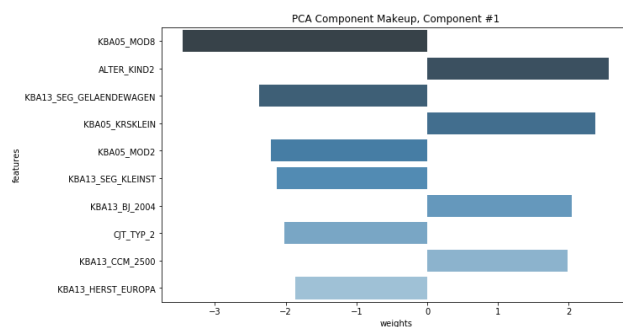
Note that flags for missingness are not used (nor consolidated) for the PCA process to avoid increasingly dimensionality unnecessarily. Rescaling and PCA is not used in the classification modelling to assist interpretation of results; the cluster results are added to the supervised dataset as a feature.

b) Principal Component Analysis

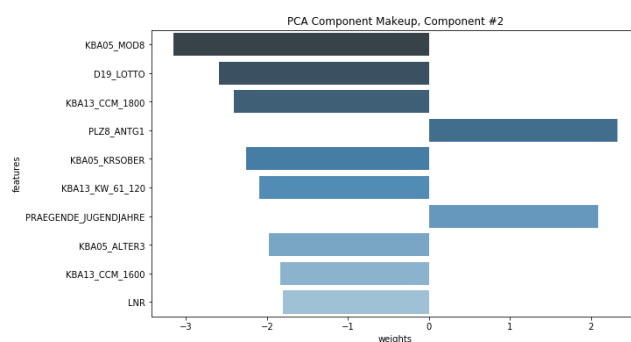
To improve clustering results, the 300+ dimension dataset was first compressed via principal component analysis. PCA was run on the underlying dataset with median imputation to avoid dimension expansion due to missingness. A PCA model was fitted using default parameters with exploration of dimension in range of 1 – 100 and the degree of explained variance was recorded.



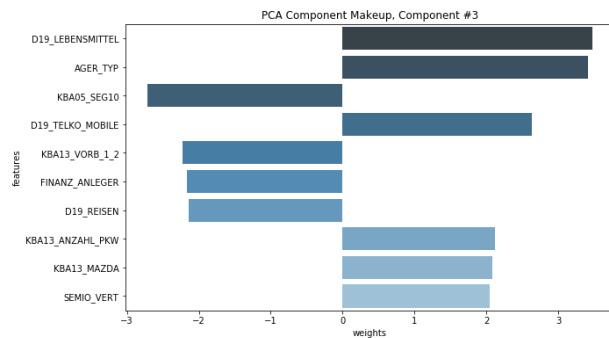
Using 100 dimensions was found to capture 76% of variance. Attributes of top PCA components are illustrated below although these do not allow for easy interpretation.



- PCA 1: Low share of vans, young children (?), low share of all-terrain vehicles in area, high share of small cars but low share of very small cars, low share of middle-class cars or European cars



- PCA 2: Low share of vans, high LOTTO transaction activity, high share of 1-2 family houses, low share of upper-class cars, relatively young

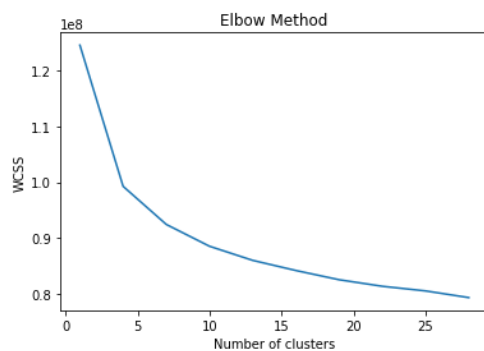


- Elderly, infrequent food purchases, low investor, high share of mazdas

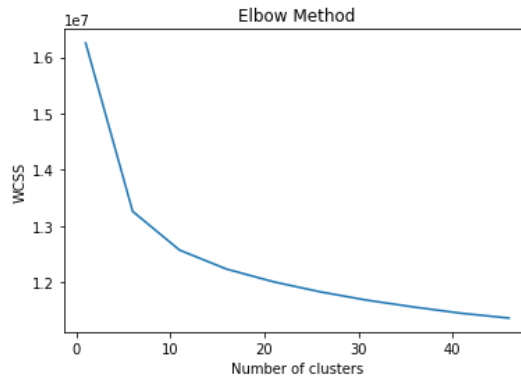
c) Clustering

K-means clustering was run on the full and PCA spaces to judge the most appropriate dimension adjustment. As expected, cluster differences are seen to be smaller in PCA-space. Visually, these charts suggest an inflection in the range of 8-10.

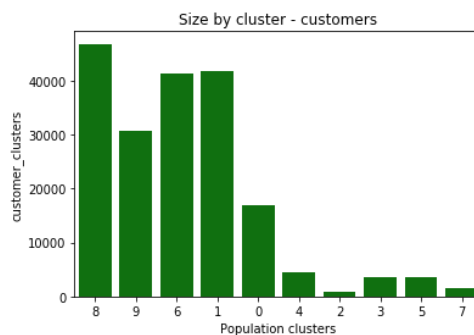
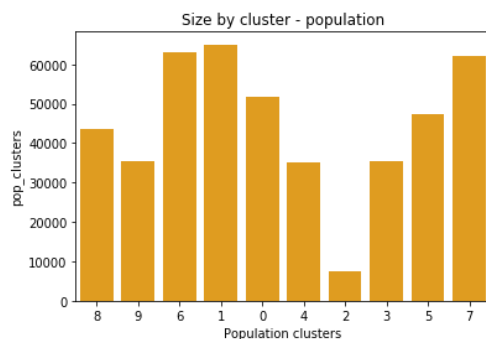
Clustering in PCA space:



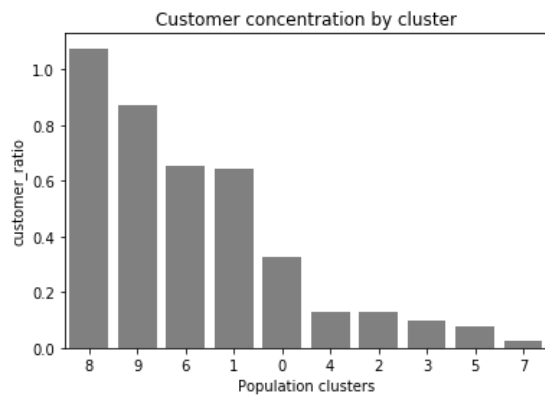
Clustering in full space



Using 10 clusters in the PCE-space yields the below results when applied across the population and customer base. The use of the PCA space provides more coherent clusters but limits ability to interpret the clusters.



Focussing on the ratio of customer cluster size to population indicates the representation of groups in the customer dataset vs the population. It can be seen that the relative prevalence varies significantly with cluster. Note that the ratio exceeds 1 for group 8; this is not necessarily a surprise given that this analysis is based on 50% of the population dataset for computational efficiency.

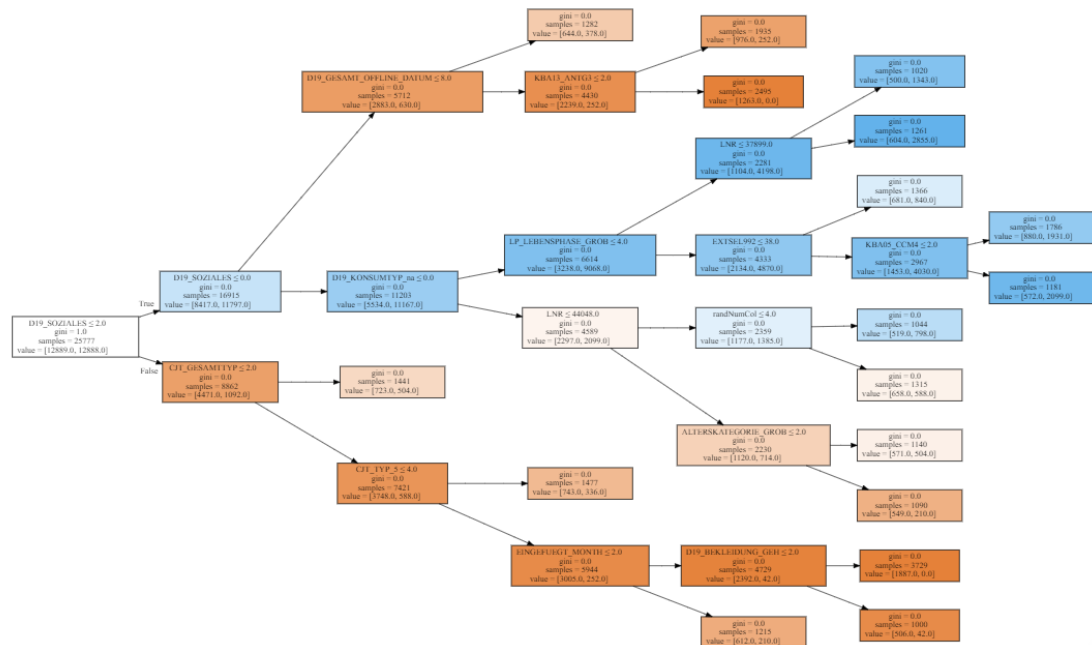


Consideration was given to feature engineering the cluster outcomes to improve predictive power in the classification task eg by reordering cluster names or adding the customer ratio. Since the main modelling approach uses tree models this was deemed unnecessary as patterns can still be detected via multiple splitting steps.

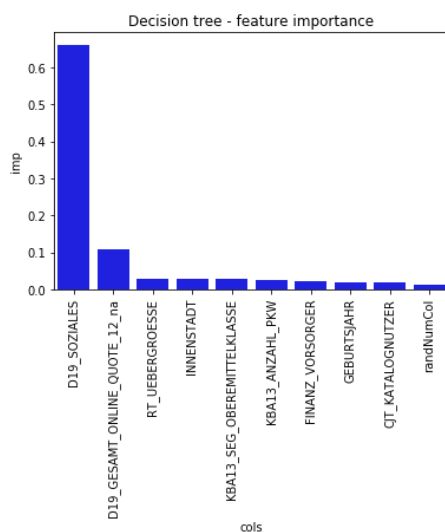
REPORT- CUSTOMER ACQUISITION

a) Benchmark modelling

In line with the proposal, a decision tree classifier was first fitted to with balanced class weights and minimum samples leaf of 1000, otherwise utilising default parameters.

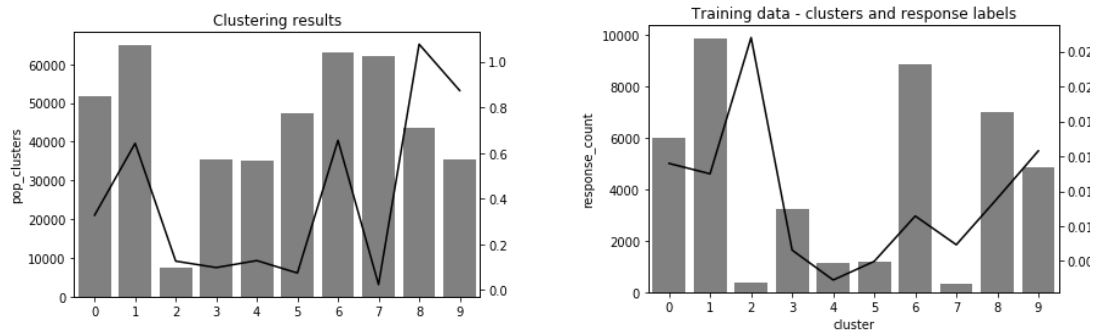


This simple model yields reasonable results with an AUC ROC score of ~0.76. This suggests the data processing pipeline and tree modelling is appropriate. Examining feature importance demonstrates that the predictive power is dominated by the variable “D19_SOZIALES”



Using clustering data

To explore the usefulness of the cluster information, the response rate by group is compared visually to the representation ratio in customer/population groups from task 1. Similarities are not obvious so cluster are not expected to be predictive of response rates, in isolation at least.



Using a random forest with just cluster information as input yields a low ROC-AUC score of ~ 0.55 indicating that the cluster information provides very limited predictive information itself. This outcome is somewhat disappointing and suggests task 1 should be revisited to explore whether improvements can be made.

b) Model training and selection

No structure is evident in the datasets so random partitions are used to model training, validation and testing. A specific held-out validation set was created to compare models and submissions to Kaggle were intentionally limited to avoid overfitting to the test set.

Dataset	Approach	Design
Training & tuning	For hyperparameter tuning	Five-fold cross validation using grid search
Validation	For comparison of model outcomes	25% held out
Final test	Kaggle testing data	Using optimal hyperparameters on combined train and validation sets for best performance

Several classical ML model types were tested along with simple neural nets.

Model	Tuning approach	Comments
Random forest	<p>Grid search:</p> <ul style="list-style-type: none">• class_weight = balanced / balanced_subsample• 'min_samples_leaf' varied in range [2,2000]• 'max_features' varied in range [<0.1,1] <p>Optimal performance found with 700 leaves and max_features=0.7</p>	<ul style="list-style-type: none">• Aggregation of decision trees with bagging to control overfitting, interpret via feature importance• Also utilised random column to determine threshold for feature importance to define condensed models with top features
Gradient boosting	<p>Tuning of macro parameters:</p> <ul style="list-style-type: none">• 'N_estimators' in range [40,70]• Max_depth in range [5,10]• Subsample in range [0.5,1] <p>Then tree-specific parameters:</p> <ul style="list-style-type: none">• 'min_samples_leaf' varied in range [2,2000]• 'max_features' varied in range [<0.1,1] <p>Optimal performance found with 15000 leaves, max depth of 8, 40 trees, subsample of 0.8 and max_features of 0.7</p>	<ul style="list-style-type: none">• Constrained by computation time so search via wide parameter space before narrow searches
Neural network	<p>Used the fast.ai libraries for model build and tuning</p> <ul style="list-style-type: none">• Used two-layer neural nets with layer 1 in the range (50,500) and layer two with (10,100) nodes• Used embeddings for categorical variable inputs with dimensions of half the cardinality• Utilised variable learning rates within epochs to accelerate convergence• Explored dropout rates in range (0-0.75) for node layers and embeddings	<ul style="list-style-type: none">• Full input proved very difficult to train so condensed input using top variables from tree modelling was used• Class imbalance created challenges with model fitting – model tends to converge on probs <5% for all items which still achieves 0.78 AUC score• Loss function calculated on the differentiable cross-entropy but performance analysed using ROC

Logistic regression	Simple approach using same training set as tree models	<ul style="list-style-type: none"> Only used for simple benchmark Limited performance and understanding as all features have not been encoded or explicitly ordered
Stacked models	Linear (parallel) combinations of above models. Combining models based on simple heuristics rather than full optimisation	<ul style="list-style-type: none"> Offer best performance with ensembling as errors tend to offset one another

c) Model results

Results are reported below:

Model	Train ROC	Validation ROC	Kaggle test ROC	Notes
Benchmark – decision tree	0.815	0.758		
Logistic regression	0.875	0.620		
Random Forest	0.821	0.781	0.789	
Gradient Boosting	0.807	0.768	0.787	
Neural network		0.785	0.791	
Best stacked model			0.7998	50:50 weighting between RF and GBM



It can be seen that ensembling decision trees does improve performance as expected. The poor performance of the logistic regression model is unsurprising as the features were not prepared specifically for linear modelling, with no interaction terms or careful ordering. Performance on the Kaggle test set slightly exceeds the validation performance; this is likely because the final models are trained on more data.

Focussing on the pure classification problem yields the below confusion matrices for the validation set. It can be seen that using default thresholds yields low accuracy for random forest models and highlights issues with the GBM model. The model results are better interpreted using the label probabilities

	Predictions							
	Random Forest		GBM		Random Forest, condensed		Neural net, condensed inputs	
	0	1	0	1	0	1	0	1
0	12280	4711	16991	0	12588	4403	16991	0
1	57	137	193	1	68	126	194	0

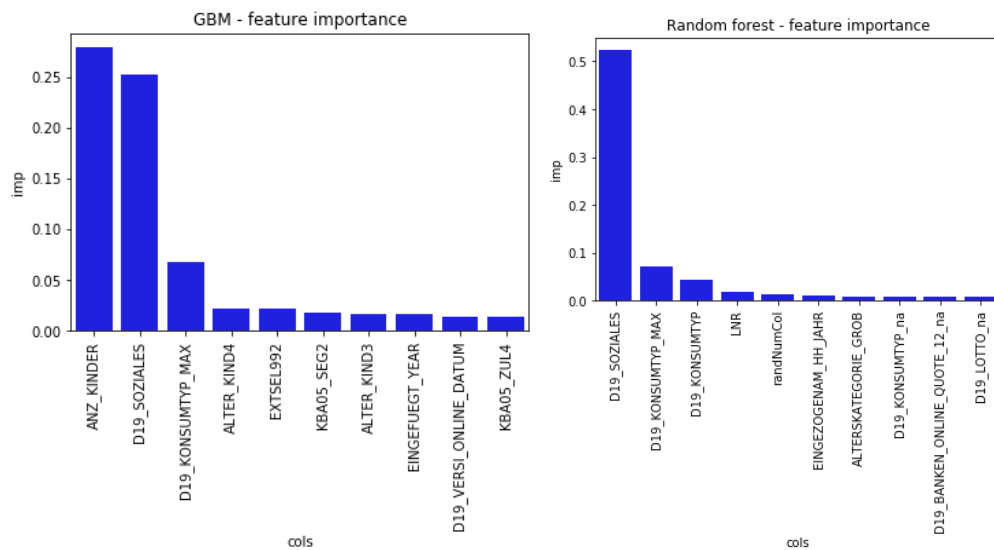
d) Kaggle submission

Used simple model stacking with 50:50 weighting between gradient boosting and random forest predictions yielded a ROC-AUC of 0.79986 and a ranking of 57/200 on submission date.

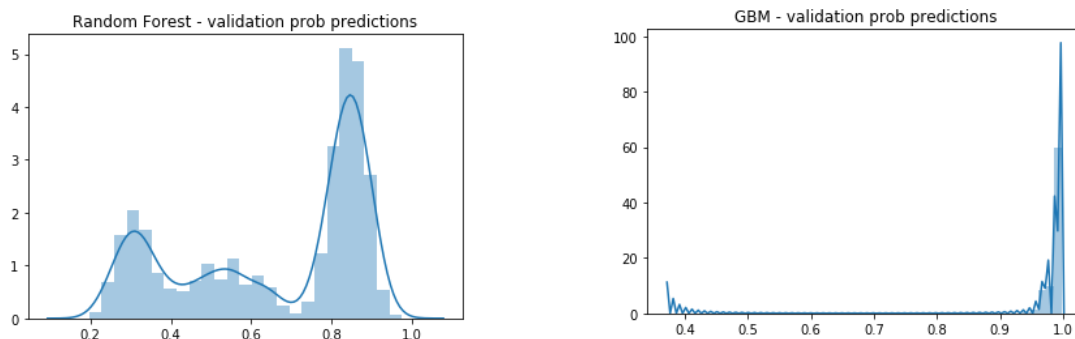
57	MGriffin		0.79986	12	~10s
Your Best Entry 					
Your submission scored 0.79974, which is not an improvement of your best score. Keep trying!					

e) Model interpretation

Feature importance is used to determine the most important variables in the tree modelling



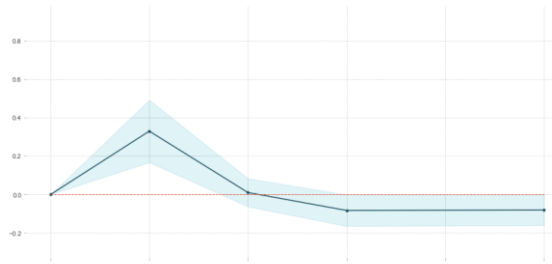
Examining the distribution of probabilities predicted by the models is also instructive. Visually the random forest appears to generate low, medium and high bucketed predictions. The GBM results are highly skewed suggesting further model tuning is required.



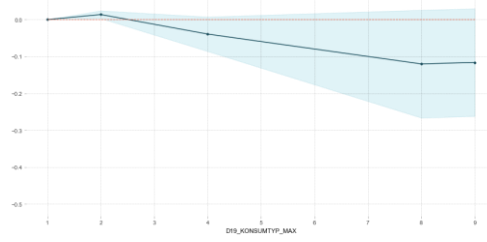
Interpretation of the results therefore focus on the random forest which produces more intuitive probability profiles. Partial dependence charts help reveal the sensitivities to these top variables:

- Potential customers assigned as “Multi-buyer 0-12 months” on social costs are significantly more likely to respond positively.
- Targets with consumption type of family, informed, modern or inactive are less likely to respond
- Surprisingly LNR is found to be predictive suggesting there may ordering within the dataset
- More recent move-in dates increase the likelihood of response
- People in the age range of 45-60 yr are less likely to respond.

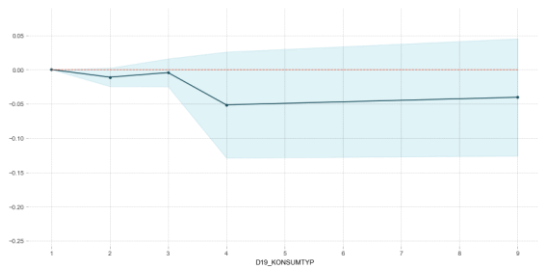
PDP for feature "D19_SOZIALES"
Number of unique grid points: 5



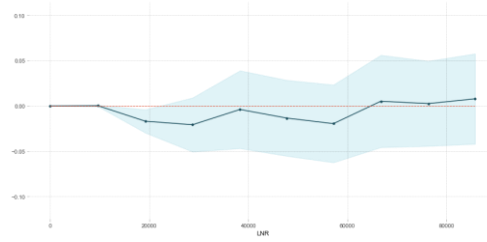
PDP for feature "D19_KONSUMTYP_MAX"
Number of unique grid points: 9



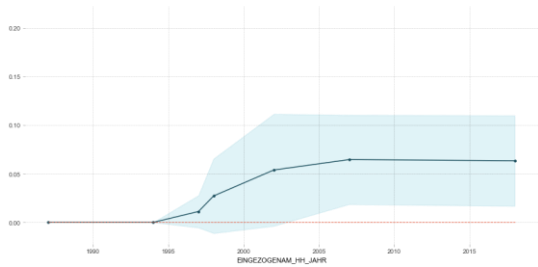
PDP for feature "D19_KONSUMTYP"
Number of unique grid points: 9



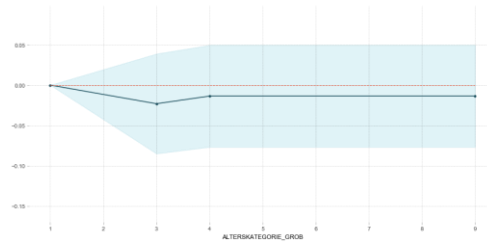
PDP for feature "LNR"
Number of unique grid points: 10



PDP for feature "EINGEZOGENAM_HH_JAHR"
Number of unique grid points: 7



PDP for feature "ALTERSKATEGORIE_GROB"
Number of unique grid points: 4



SUMMARY

Conclusions

This work explores customer clustering and predictive modelling. PCA and k-means clustering was used to identify 10 population clusters which were all present in the customer set but with varying frequency. Producing interpretable clusters with predictive power was found to be difficult, likely due to high-dimensionality and missingness. The result is somewhat dissatisfactory and I intend to revisit the analysis in future.

Despite this, the core classification task was successful with a top~25% result on Kaggle via a standardised data science pipeline using simple combinations of tree models with basic data pre-processing. As expected ensembling decision trees does help control overfitting and improves the AUC score from a benchmark of 0.76 to 0.80. Although the classification accuracy remains low, the scenario means false positives are inexpensive and the models does offer useful predictions to design better target widespread campaigns.

Further Work

There are a number of additional tasks I would like to explore further across the pipeline:

- **Data preparation:** other imputation approaches on key variables and variable-by-variable analysis to clean feeds (eg consolidating “unknown” labels and missing data”).
- **PCA & clustering:** explore clustering without PCA to produce more interpretable clusters, plus revisiting PCA/clustering to try to find groupings which offer predictive power for the classification task.
- **Modelling:** explore XGBoost for superior implementation of gradient boosting with easier tuning, SMOTE/imblearn for more-sophisticated sample rebalancing to improve classification results. Auto-ML could also optimise model stacking for improved Kaggle results. Explore possible transfer learning for neural nets using larger population/customer datasets