

Machine Learning Engineer Nanodegree

Capstone Project

Pawel Wisniewicz
08-06-2020

I. Definition

Project Overview

How many times you received an email, got a call from a pushy salesman who was trying to sell you something which you didn't want? You were irritated because of getting an unwanted offer and the salesman lost time so the company lost money. What was a chance that you would get it? What if the salesman could select only prospective customers who are most likely to purchase the product? Nowadays more and more data is collected, due to that, there are new possibilities for businesses to explore that data and find some insights which can boost their effectiveness. Forbes says that Machine learning is using for a variety of applications in marketing. User personalization and Customer segmentation are just a few processes supported by algorithms. Machine learning can eliminate the guesswork and improve targeted advertising effectiveness.

In this project, I developed a set of algorithms to improve the process of selecting the right people. Using demographics data about the general population I applied the K-Means clustering technique to select prospective customers. I also trained the XGBoost supervised machine learning model to predict the probability of each individual becoming a new customer.

Problem Statement

The main challenge is to improve the selection of people for targets of a marketing campaign. There is a lot of data gathered about each customer, that data can provide some useful insights to perform the population segmentation where the new demographic data can be analyzed. Similar attributes can be found to help identify prospective customers. Using K-Means algorithms data can be grouped into clusters. The main goal is to identify attributes in the population dataset which indicate that a person could buy a product. The second task is to develop a classification model to predict how likely a single person is to become a new customer.

Metrics

Model performance is measured using AUC which results in the ratio between True positive and False positive rates. The data is unbalanced because there are many more individuals who were targeted and did not respond. Using simple accuracy score our model could perform very well but its sensitivity could be very low:

$$TPR = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

To overcome this problem, I also measured a True positive rate to check how sensitive the model is detecting new customers.

II. Analysis

Data Exploration

The data is delivered by Alvato Financial Solutions. They provide 4 datasets:

- Demographics information about the general population in Germany. 891,211 samples with 366 features.
- Demographics information about customers of the sales company in Germany. 191,652 samples with 369 features.
- Demographics information about people targeted with the campaign. 42,982 samples with 367 features.
- Demographics information about people targeted with the campaign. 42,833 samples with 366 features.

The general population dataset includes different information such as age, gender, personal profile, purchase information, state of possession, and many other features provided for all individuals.

In the customers' dataset, there are the same features plus 3 additional 'CUSTOMERS_GROUP', 'PRODUCT_GROUP', and 'ONLINE_PURCHASE'. These extra columns provide more precise information about every customer profile. The third dataset contains information about targeted people along with additional column 'RESPONSE' indicating whether each person was attracted by the campaign or not. The remaining features are the same. The last dataset includes test data, the same columns like in the third dataset except for 'RESPONSE' column, for this portion of data we need to predict how likely each person is to become a new customer.

There are also two excel files:

- List of attributes and descriptions segmenting them by the type of attribute e.g. Person, Household, etc.
- A detailed list of attributes with their values described.

Fig. 1 Dataset sample

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3
0	910215	-1	NaN	NaN	NaN	NaN	NaN
1	910220	-1	9.0	0.0	NaN	NaN	NaN
2	910225	-1	9.0	17.0	NaN	NaN	NaN
3	910226	2	1.0	13.0	NaN	NaN	NaN
4	910241	-1	1.0	20.0	NaN	NaN	NaN

There are various data types and also some missing values. In the list of attributes, some additional information can be found.

Fig. 2 Dictionary sample

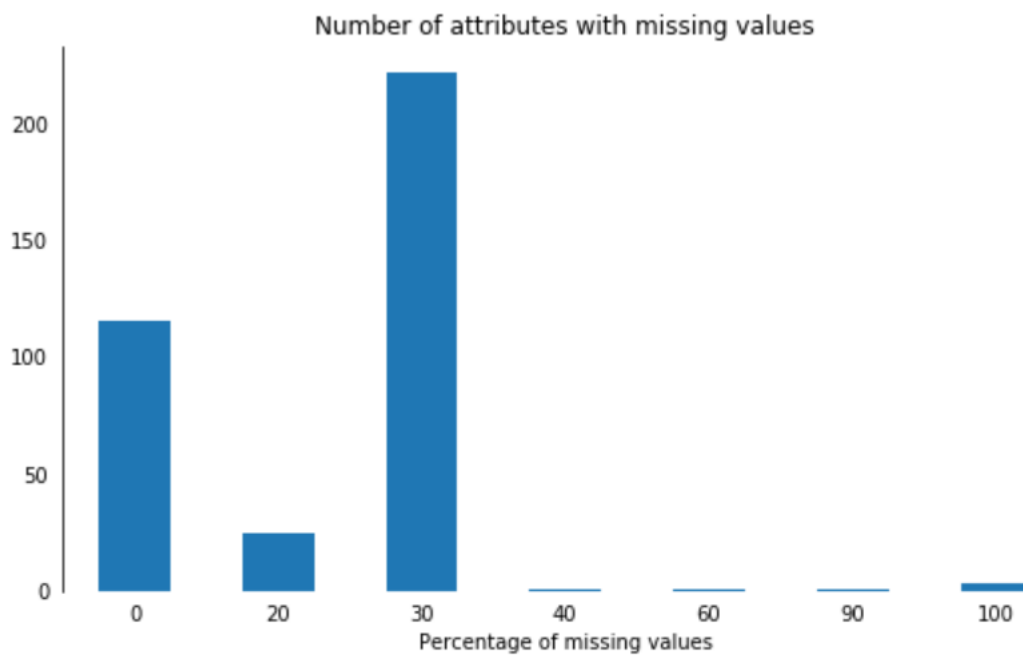
	Attribute	Description	Value	Meaning
0	AGER_TYP	best-ager typology	-1	unknown
1	AGER_TYP	best-ager typology	0	no classification possible
2	AGER_TYP	best-ager typology	1	passive elderly
3	AGER_TYP	best-ager typology	2	cultural elderly
4	AGER_TYP	best-ager typology	3	experience-driven elderly

Exploratory Visualization

Variables are represented in different scales. There are nominal, ordinal, interval, ratio, and binary data. Some columns are not described in the dictionary, so it is challenging to guess what they mean.

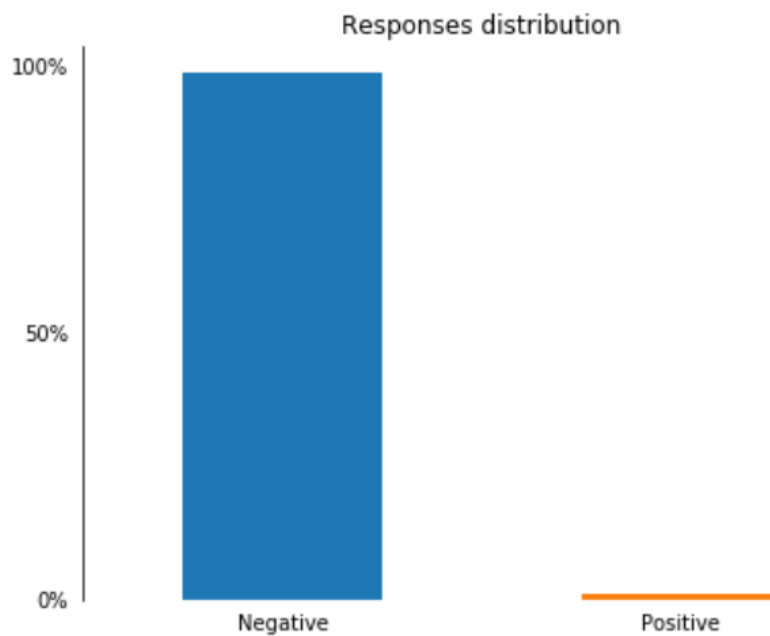
There is quite a lot of attributes with missing values. Over 200 attributes have around 30% of the missing values in their columns. There are over 100 attributes in good shape and around 20 with 20% missing values.

Fig. 3 Percentage of missing values in attributes



The dataset with responses from targeted people is imbalanced because there are many more people who responded negatively.

Fig. 4 Responses imbalance

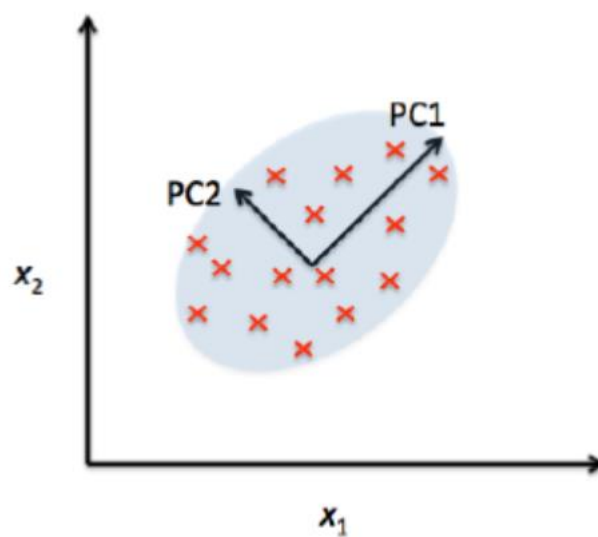


There is a bit more than 1% of positive responses. Creating a classification model, it must be considered to avoid the model leaned towards negative responses.

Algorithms and Techniques

There are two problem statements to be solved. The first one is population segmentation which can be handled using unsupervised machine learning techniques to cluster data. After the data cleaning process, I used the Principal component analysis to reduce the dimensionality of the data. A highly dimensional dataset leads to model overfitting so the model loses the accuracy on unseen data. PCA calculates the correlation between features and combines similar or redundant features to form a new, smaller feature space.

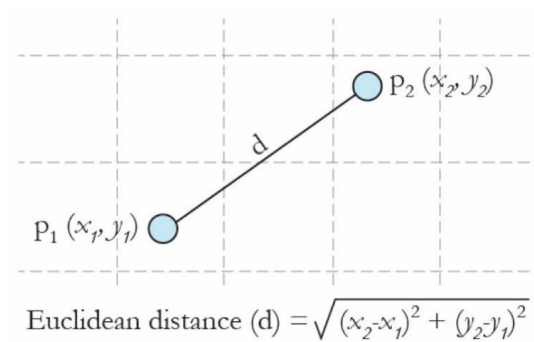
Fig. 5 PCA 2-dimensional data



On the above figure, we can see 2-dimensional feature space which can be reduced to 1-dimensional by fitting the linear function which best correlates with given points. Each feature expresses some data variance captured in it. We can select the number of components that represent high enough variance.

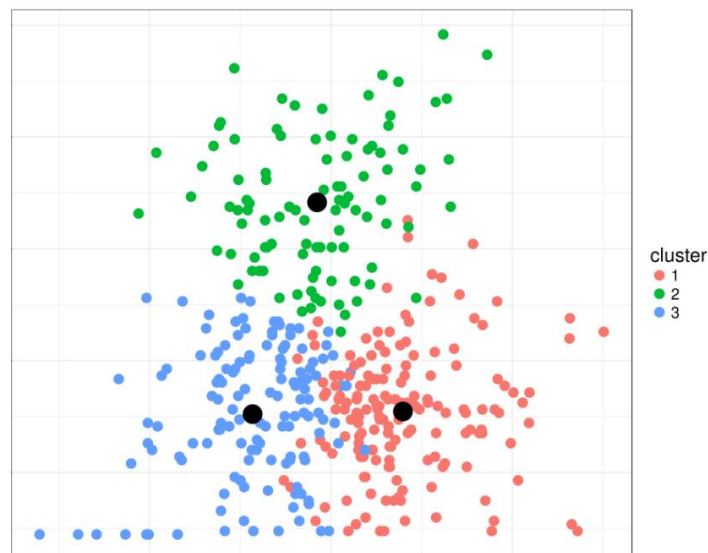
After selecting new features, I applied K-Means clustering which is a very powerful technique. It can group similar data points by calculating the distance between the cluster centroid and each point, the closest points are clustered together. The number of clusters needs to be arbitrarily selected then random centroids are assumed, and Euclidean distance is calculated between each point and the centroids.

Fig. 6 Euclidian distance



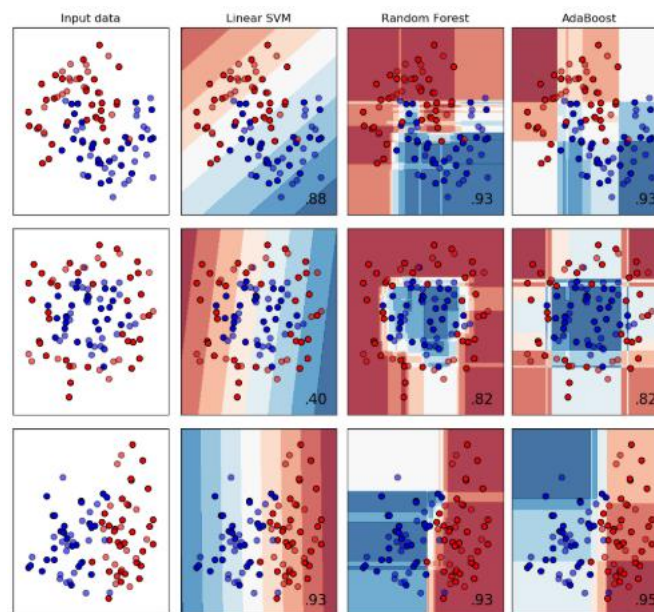
The closest points are assigned to each centroid, so clusters are formed. Then the position of the new centroid is taken as the mean of cluster points and the whole process is repeated until there is only negligible change in the position of new centroids.

Fig. 7 K-Means Clustering



The second task is to build a prediction model that can calculate the probability for each person becoming a new customer. Supervised machine learning algorithms can be used to classify each observation based on demographic data. Depends on the nature of the input data different algorithms may return different results. Below is an example of 3 portions of 2-dimensional input data and Linear SVC, Random Forest, and AdaBoost classifier's results.

Fig. 8 Supervised learning classification



Models characteristics:

- Linear SVC can fit the hyperplane that divides samples into 2 classes in the n-dimension space. Hyperparameters:
 - o class_weight: defines weights of classes.
 - o C: regularization parameter.
- Random Forest is an ensemble algorithm that builds a set of decision trees from the subset of training data. It aggregates results from the number of trees to compute the final class. Hyperparameters:
 - o n_estimators: number of decision trees.
 - o max_depth: the maximum depth of the tree.
 - o min_samples_leaf: the minimum number of samples required to be at a leaf node.
 - o class_weight: defines weights of classes.
- AdaBoost is an ensemble algorithm that uses boosting to convert weak learner to strong ones. Hyperparameters:
 - o n_estimators: number of estimators.
 - o learning_rate: how fast the model is trained.

Hyperparameter tuning is discussed in the refinement section.

Benchmark

Because this is a Kaggle competition a benchmark model can be the best AUC score for the test set. The best model got AUC value of 0.81 which is a very good result due to the number of missing values and noise. I compared my model with results from the Leaderboard.

III. Methodology

Data Preprocessing

The data cleaning process consists of the following steps:

- Replacing all missing values with NaN and removing columns with more than 30% null values.
- Removing features with the variance lower than 5%.
- Calculating the Spearman correlation between ordinal variables and removing columns with a correlation greater than 0.8. For binary and continuous features Pearson correlation was used.
- Manually analyzing all remaining columns for any hidden errors. Dropping irrelevant and binning some wide range features.

After feature engineering, I ended up with 174 columns left. To successfully cluster samples calculating Euclidian distance further dimensionality reduction needs to be applied. PCA algorithm was able to reduce the number of features to 10 maintaining 30% of the original data variance.

For the classification problem, I used recursive feature elimination with cross-validation selecting the best performing features fitting the cleaned dataset into the Random Forest classifier. As a result, I got 28 most important features.

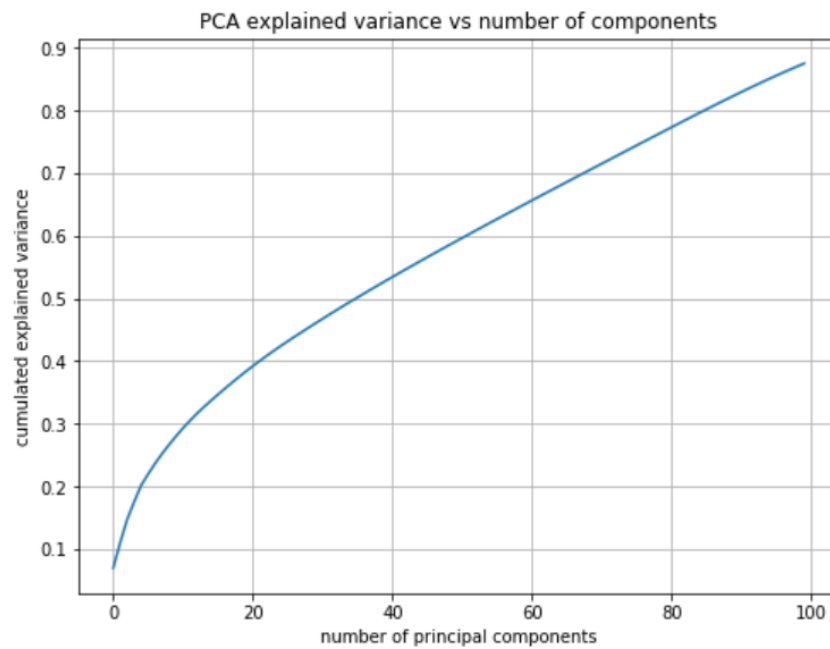
Implementation

The implementation process can be split into two problems:

- Population segmentation using the clustering algorithm
- Prediction model using the classification algorithm

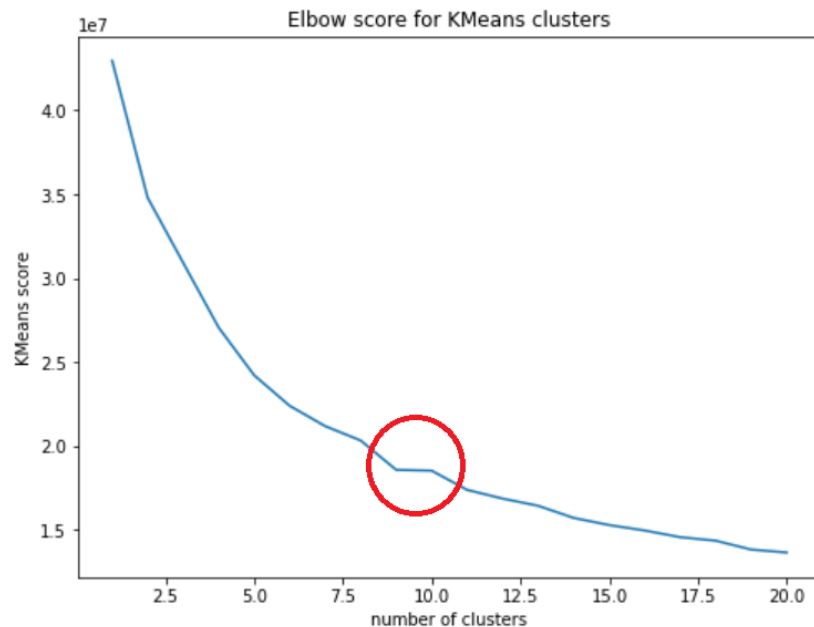
In the first task, I selected the top 10 components which capture 30% of the original data variance.

Fig. 9 PCA analysis plot



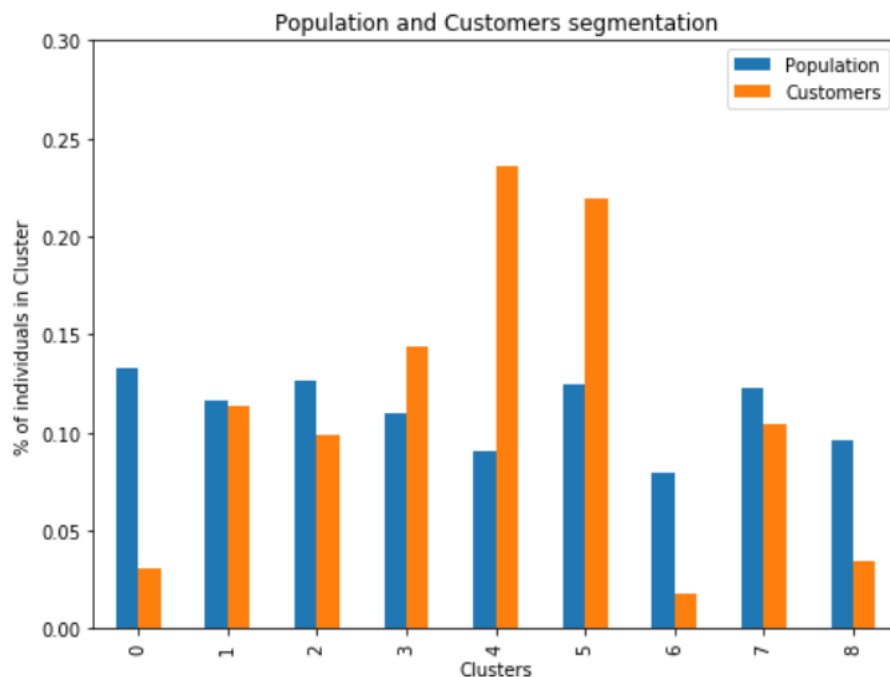
Then I fitted transformed data into the MiniBatchKMeans estimator to decide on the best number of clusters. After trying different values for k in the range from 1 to 20, at some point the centroid distance reaches "elbow" which is indicated by a sharp stop in decreasing. I chose $k=9$ for the number of clusters to create using K-Means.

Fig. 10 Elbow method



After clustering all samples, I plotted the percentage of samples assigned to each cluster for both the population and customers dataset.

Fig. 11 Population segmentation



Observations:

- Cluster 4 and 5 may indicate potential customers as in these sits the most current customers
- Clusters 0 and 6 and rarely occupied by customers.

To build the most accurate prediction model I split the training dataset into 80% train and 20% validation sets. Due to high-class imbalance, I used stratified sampling to maintain the same proportion of positive and negative responses in both datasets.

Code

```
X_train, X_val, y_train, y_val = train_test_split(train_X, train_y, test_size=0.20,  
                                                stratify=train_y, random_state=1)
```

To select the best performing model, I created a learning curve showing the capabilities of different models to learn on the given dataset.

Code

```
def learning_curve_graph(estimator, X_train, y_test, title, ax):
    """
    Generate a learning curve.
    Params:
        estimator: model object
        X_train (Dataframe): training set
        y_test (Series): labels
        title: plot title
        axes: subplot position
    Return:
        plt: matplotlib object
    """
    sf = StratifiedKFold(n_splits=10, shuffle=True, random_state=1)
    sample_size, train_scores, test_scores = learning_curve(
        estimator, X_train, y_test, scoring='roc_auc', cv=sf)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    plt.axes = ax
    # plot train score with std
    plt.plot(sample_size, train_scores_mean, 'o-', color="red", label="Training score")
    plt.fill_between(x=sample_size, y1=train_scores_mean - train_scores_std,
                     y2=train_scores_mean + train_scores_std, alpha=0.1, color="red")
    # plot test score with std
    plt.plot(sample_size, test_scores_mean, 'o-', color="green", label="Validation score")
    plt.fill_between(x=sample_size, y1=test_scores_mean - test_scores_std,
                     y2=test_scores_mean + test_scores_std, alpha=0.1, color="green")

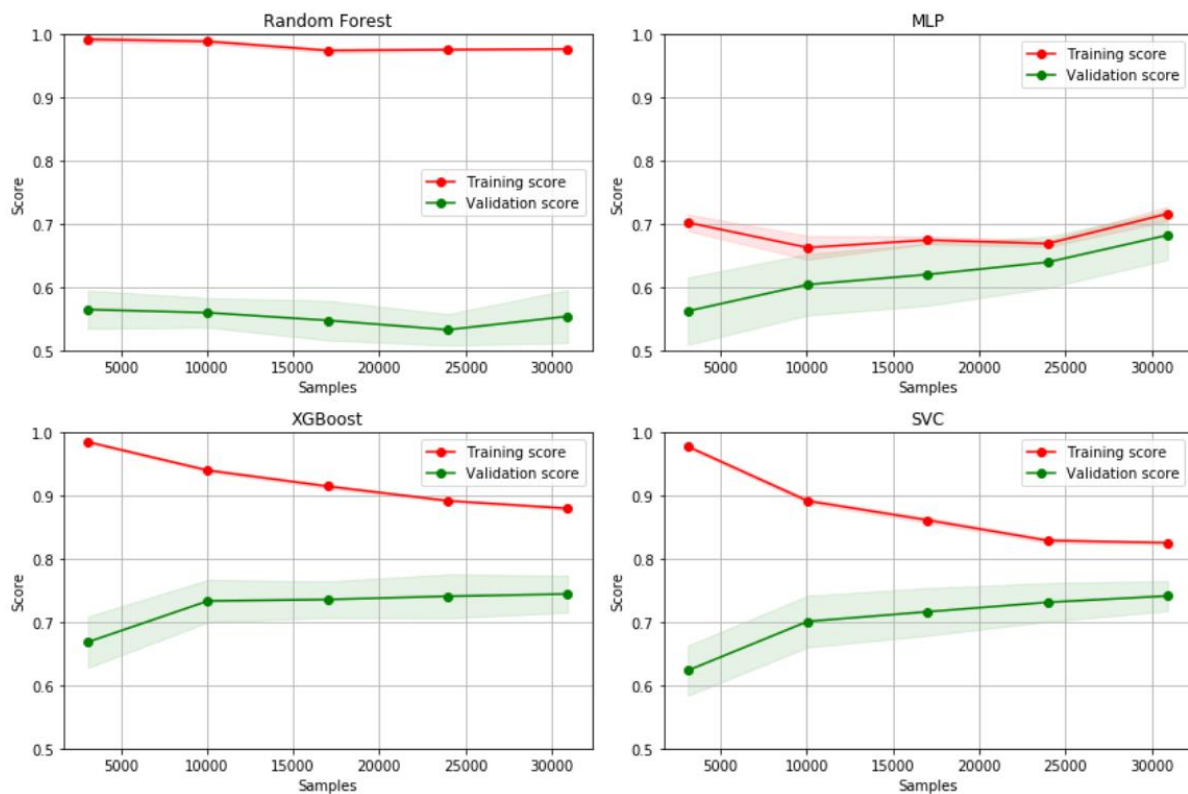
    # graphical settings
    plt.ylim(0.5, 1)
    plt.title(title)
    plt.xlabel('Samples')
    plt.ylabel('Score')
    plt.grid()
    return plt
```

This function takes an estimator object for the selected model and the dataset. I compared the performance of 4 models:

Code

```
model_1 = RandomForestClassifier(random_state=1, class_weight='balanced')
model_2 = MLPClassifier(random_state=1, activation='logistic', hidden_layer_sizes=(30,30))
model_3 = XGBClassifier(random_state=1, objective='binary:logistic', scale_pos_weight=negative/positive)
model_4 = LinearSVC(random_state=1, class_weight='balanced')
```

Fig. 12 Predicting model's learning curves



Conclusion:

- Random Forest is overfitted predicting training data very good but on the validation data, it's not doing good.
- XGBoost is doing not bad but there is quite a big gap between the training and validation predictions, hyperparameters tuning may improve it.
- MLP has the lowest variance between the training and validation set but the AUC score is quite low compare to other models.
- Linear SVC performs the best on this dataset.

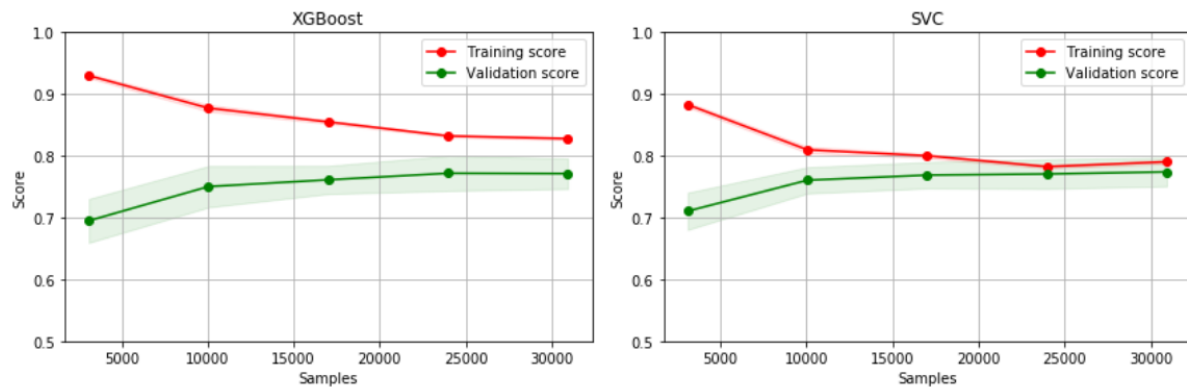
Refinement

Code

```
def select_features(all_X, all_y):
    """
    Using Reacursive feature elimination and cross validation select the most important features
    Params:
        all_X: DataFrame with training data
        all_y: Series with labales
    """
    all_X = all_X.fillna(all_X.mode())
    model = RandomForestClassifier(n_estimators=100, random_state=1, max_depth=2, min_samples_leaf=2)
    sf = StratifiedKFold(n_splits=10, shuffle=True, random_state=1)
    selector = RFECV(model, cv=sf, step=5, scoring='roc_auc')
    selector.fit(all_X, all_y)
    best_features = all_X.columns[selector.support_]
    return best_features
```

To select the most appropriate features I used the recursive feature elimination with cross-validation fitting the cleaned dataset into the Random Forest classifier. Due to that, I reduced the number of features to 28. After that, I plotted the learning curve for two selected models.

Fig. 13 XGBoost and Linear SVC learning curves



Observations:

- SVC has a learning curve for both training and validation dataset very close to each other which may indicate that it's immune to overfitting.
- XGB has a good increasing rate and similar results.

Lastly, I tuned hyperparameters for both models using Grid search and got the highest AUC for the following parameters:

Code

```
# XGBoost hyperparameters
XGBClassifier(booster='gbtree', gamma=0.5, learning_rate=0.1,
              max_depth=2, min_child_weight=1, n_estimators=100,
              scale_pos_weight=negative/positive, subsample=1.0,

# SVC hyperparameters
LinearSVC(C=1, class_weight='balanced')
```

Model performances after hyperparameters tuning with 10-fold cross-validation:

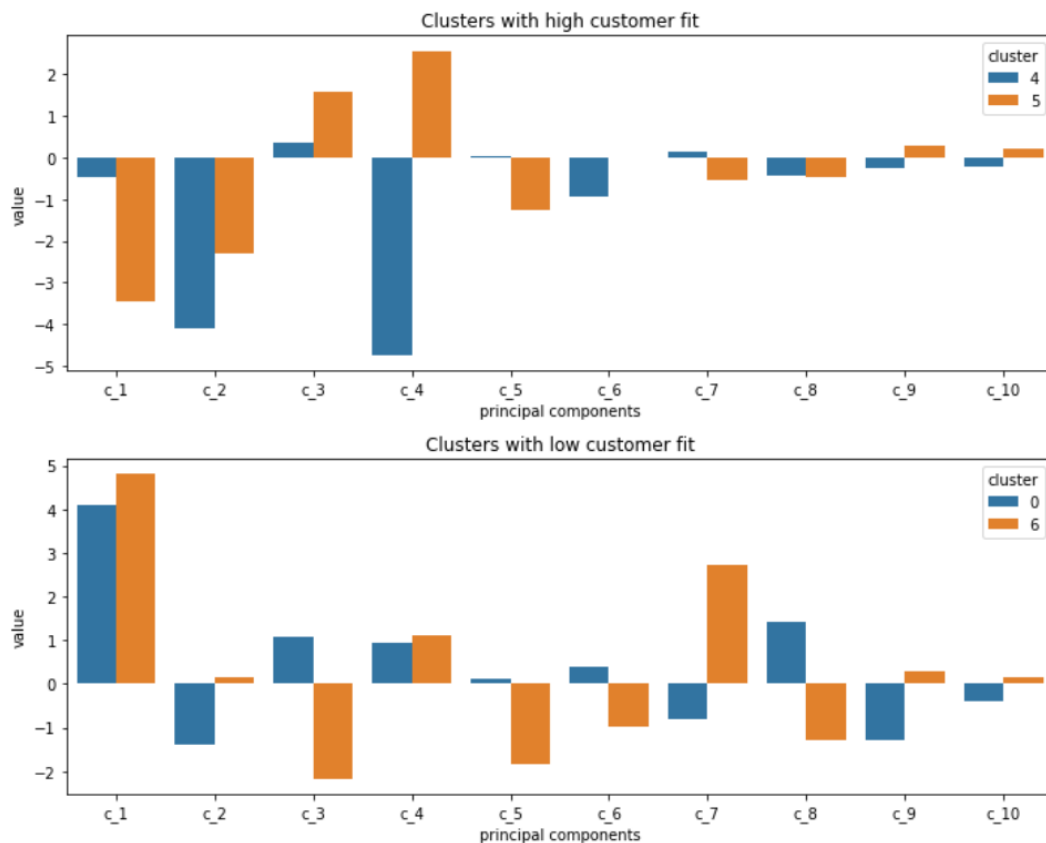
- XGB model AUC score: 0.773
- SVC model AUC score: 0.774

IV. Results

Model Evaluation and Validation

To find which features, decide to which cluster each sample was assigned I looked at the composition of clusters with a high share of customers as well as these with a low share of customers and a greater share of the population to compare both.

Fig. 14 Principal components importance in clusters



Cluster 5 is dominated by components 3 and 4 which the highest positive weights and by components 1 and 2 with the highest negative weights. After looking at each principal component weight, I could find the most important features which characterize each component.

Conclusion:

After analyzing all principal components, I concluded that new customer may become a person who doesn't like online purchases, has a rather low income, is religious, its gender: male.

In terms of the supervised model, both models performed quite good in cross-validation, next they were tested on the unseen validation set.

Metric	XGBoost	LVC
Sensitivity	0.698	0.708
AUC	0.719	0.717

Observations:

- SVC model scored higher sensitivity which indicates that was able to catch more true positives
- XGB model is a little less sensitive but it got slightly better results in the AUC score which is our final metric





Decision:

I selected the XGBoost classifier as a final model.

Justification

In the Kaggle competition, I scored 0.79 scores on the test data. At the moment of writing this report, I was 96th on the Leaderboard, which puts me among the top 48% of participants. The best-achieved score was 0.81 which is not very far from mine.

Fig. 15 Kaggle Leaderboard

95	DataDaku		0.79192	1	1mo
96	dev-pawel		0.79162	1	2d
Your First Entry  Welcome to the leaderboard!					
97	Santosh Kannan		0.79141	1	5d

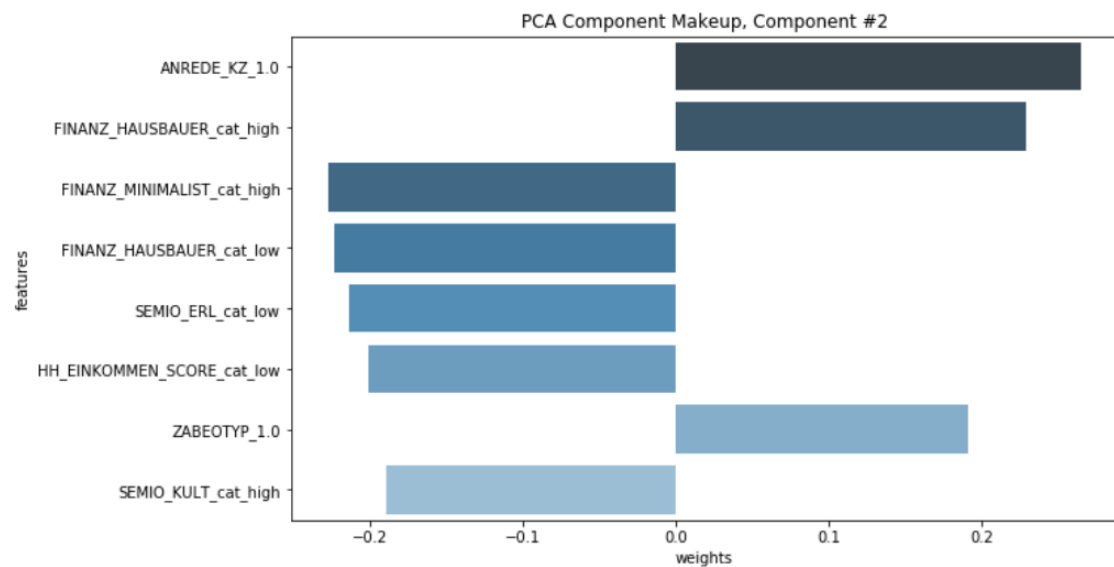
I think that on such noisy and hard to clean data that's a pretty good result. My model could help marketing people to make better decisions.

V. Conclusion

Free-Form Visualization

Each PCA component makeup can be visualized based on the weights of the original features which are included in that component. The following plots show the top 8 features of each component.

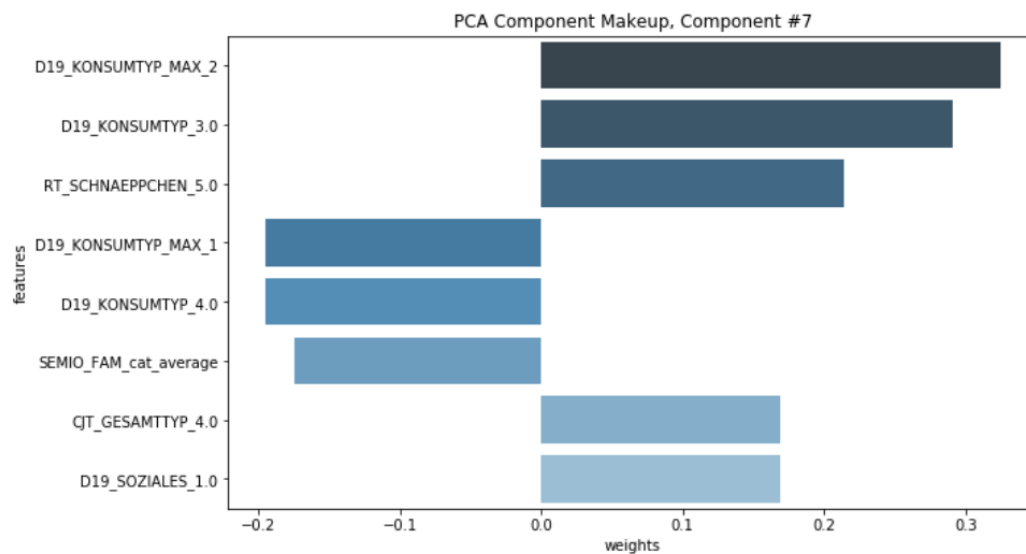
Fig. 16 2 principal components weights



Observations for the second component:

- High weights:
 - o D19_GESAMT_ONLINE_QUOTE_12 = using online transactions
 - o ONLINE_AFFINITAET = buying online
- Low weights:
 - o SEMIO_ERL = person eventful oriented
 - o CJT_KATALOGNUTZER = not in the dictionary
 - o SEMIO_FAM - a person not familiar minded

Fig. 17 7 principal components weights



Observations for the second component:

- High weights:
 - o D19_KONSUMTYP = gourmet
 - o CJT_GESAMTTYP = advertising interested online-shopper
- Low weights:
 - o D19_KONSUMTYP = family
 - o SEMIO_FAM = person familiar minded neutral

Reflection

The process used for this project can be summarized in the following steps:

1. Data exploration
2. Data cleaning
3. Dimensionality reduction using PCA
4. Population segmentation using K-Means cluster
5. Searching for patterns in clusters
6. Predictive model building
7. Features selection and hyperparameters optimization
8. Validating predicted values

The first difficulty was to clean data and understand what all features mean. It was not fully explained in the additional lists. Some of the features were not described at all. I had to review all feature's values and descriptions to assign the proper type of scale each feature. Another challenge was the number of features to understand so I needed to divide them into smaller chunks to tackle with. I spend most of the time on the data preparation process.

One of the most interesting parts is that applying a bit of mathematics and coding, very time consuming and arduous processes can be done more efficiently. I think that the final prediction model can be used to support the decision-making process while selecting the right people to target with the campaign.

Improvement

One of the improvements I could make would be to get to know more about the features, to acquire some information about features not described in the dictionary. Based on the additional information I could take better actions in the data preparation phase. Better quality data can improve model accuracy.

Calculating the correlation between ordinal features I thought to use a polychronic correlation, but I could find such an algorithm in python libraries. In terms of the model selection. I could also test some PyTorch Neural network algorithms.

References

<https://medium.com/apprentice-journal/pca-application-in-machine-learning-4827c07a61db>

<https://towardsdatascience.com/using-unsupervised-learning-to-plan-a-paris-vacation-geo-location-clustering-d0337b4210de>

<https://rpubs.com/cyobero/k-means>

https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html

<https://www.kaggle.com/c/udacity-arvato-identify-customers/leaderboard>