Machine Learning Engineer Nanodegree

Capstone project report

Customer Segmentation – Arvato Financial Solutions

Compiled by: Rajesh Mittal

Date: 05/18/2020

Contents

Project Overview

Problem domain

The goal of this project is to use a combination of supervised and unsupervised ML techniques to identify which customers are likely to buy a class of organic produce marketed by a company.

Problem background

The project is solving a real world business problem. Arvato is a services company that provides financial services, Information Technology (IT) services and Supply Chain Management (SCM) solutions for business customers. As part of its business, Arvato focuses on solving automation and data analytics needs for its customers.

In this project, Arvato is helping a Mail-order company, which sells organic products in Germany, to understand its customers segments in order to identify potential customers who would benefit from these products and likely to buy them.

Datasets and inputs

There are four data files associated with this project:

- 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).

Additionally, 2 metadata files have been provided to give attribute information:

- 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

In addition, the Train and Test data have been provided to evaluate supervised learning algorithms.

Problem Statement

The problem statement can be formulated as, “Given the demographic data of a person, how can a mail order company acquire new customers in an efficient way”.

This means two tasks

1. The demographic data of the general population and customers is to be studied using unsupervised learning algorithms. The goal in this step is to identify cohorts in the general population and in the existing customers. Using these segments, we need to analyze how the general population in Germany compares with customers for the mail-order company.
2. A result set of actual marketing campaigns is provided as a labeled training data. Supervised learning algorithms should be used to build a model that can predict whether a person is a potential customer for the mail order campaign.

Evaluation Metrics

Customer Segmentation using unsupervised algorithms

This part of the project uses PCA to reduce the number of dimensions of population feature vectors. For selecting the top features, the explained variance ratio of each feature is evaluated . Using this variance, we pick top features that capture cumulative variance that meets our desired threshold.

This is followed by a clustering exercise that detects customer segments that are in close proximity in terms of feature space. K-Means Clustering is used as an algorithm. One important parameter in K-means is to select the optimal number of clusters or value of K. For this, we plot the average square distance between all clusters for each value of K. The right value of K is selected with the help of an elbow plot.

What is K-means and why is it used for this exercise- K-means is a form of unsupervised learning technique. This is when you have unlabeled data essentially given a set of features X for each sample you don't have the Y value for that sample. The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the K-means clustering algorithm are:

- The centroids of the K clusters, which can be used to identify cluster labels for new data
- Cluster labels for the training data i.e which cluster the training sample belongs to.

We are using K-means clustering here because the problem at hand is we have a bunch of PCA features computed for population sets but don't have labels for these sets. Our goal is to cluster these points based on the similarity in multidimensional space so K-means is a suitable choice here.

Customer Acquisition using supervised learning algorithms

We have been given the labeled data for this exercise. We first split this data into train and evaluation sets. The model will be trained on the training set and will be evaluated on the test set.

The class label distribution is highly imbalanced. This is a binary classification problem as we need to predict true/false for each new customer. As part of labeled data, there are 42,430 observations with label '0' and only 532 observations with label '1'.

Accuracy or precision metrics alone wouldn’t work for this unbalanced data. AUROC metric which considers both true positive rate and false positive rate seem to be a good choice for this problem, since we want to be able to correctly predict both cases i.e. whether a person becomes a customer or not.

The AUROC gives an idea about overall performance of the model. A good performing model will have an AUROC of 1. So higher the number better the performance of the model.

Analysis and Methodology

Data Exploration and Preprocessing

One of the primary functions of any ML analysis is to ensure the data is clean and properly normalized before the models are trained. We also want to sample values in the data to ensure there are no missing or unknown values in the dataset.

For this problem, here are a few data cleaning steps we have performed.

Inspect and drop columns with too much granular information

We inspect unique values for columns like EINGEFUEGT_AM, D19_LETZTER_KAUF_BRANCHE and CAMEO_DEU_2015.
EINGEFUEGT_AM contains dates while other fields contain large variation in the data based on unique value analysis.

Handling columns with mixed features

We also notice that some features contain data values like X, XX that should be mapped to a NaN or missing value. Examples of such columns are ‘CAMEO_DEUG_2015’ and ‘CAMEO_INTL_2015’.

Re-encoding columns with categorical values

We have columns where default values are used and these values don’t correspond to the set of valid values for the column. Example is ‘LP_FAMILIE_FEIN’ where 0 is used as data which likely corresponds to a missing data value. We have the following columns that go through this transformation.
LP_FAMILIE_FEIN
LP_FAMILIE_GROB
LP_LEBENSPHASE_FEIN
LP_LEBENSPHASE_GROB
LP_STATUS_FEIN
LP_STATUS_GROB
WOHNLAGE

In addition, we also find columns like ANREDE_KZ: This column represents the Gender, which was encoded with values 1,2 for male and female. We re-encode these values to contain 0-male and 1-female to bring uniformity into our data for later stages. Another such column is OST_WEST_KZ.

Addressing Unknown values

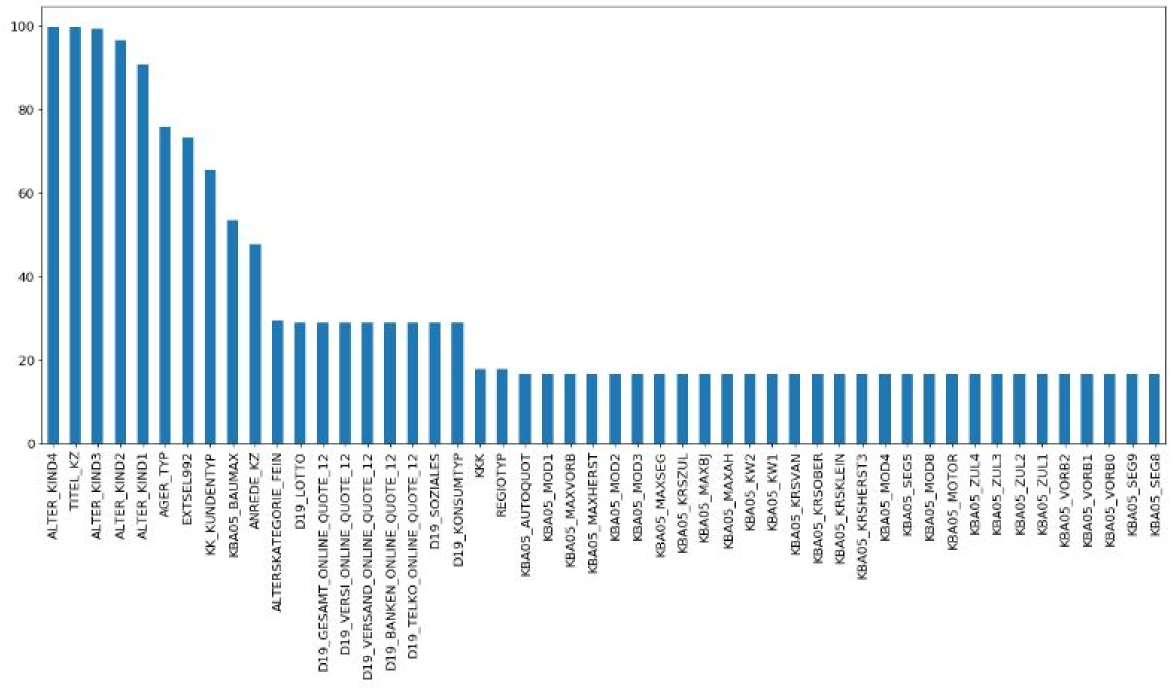
The second step is to fix the unknown representations in all the columns. The ‘Attribute-values’ excel sheet contains the information about which columns contain unknown values and how they are entered specified in the dataset. With this information all the unknown values are replaced with NaN values in the dataframes. In total, there were 232 columns which contained unknown representations.

Drop columns and rows with too many missing values

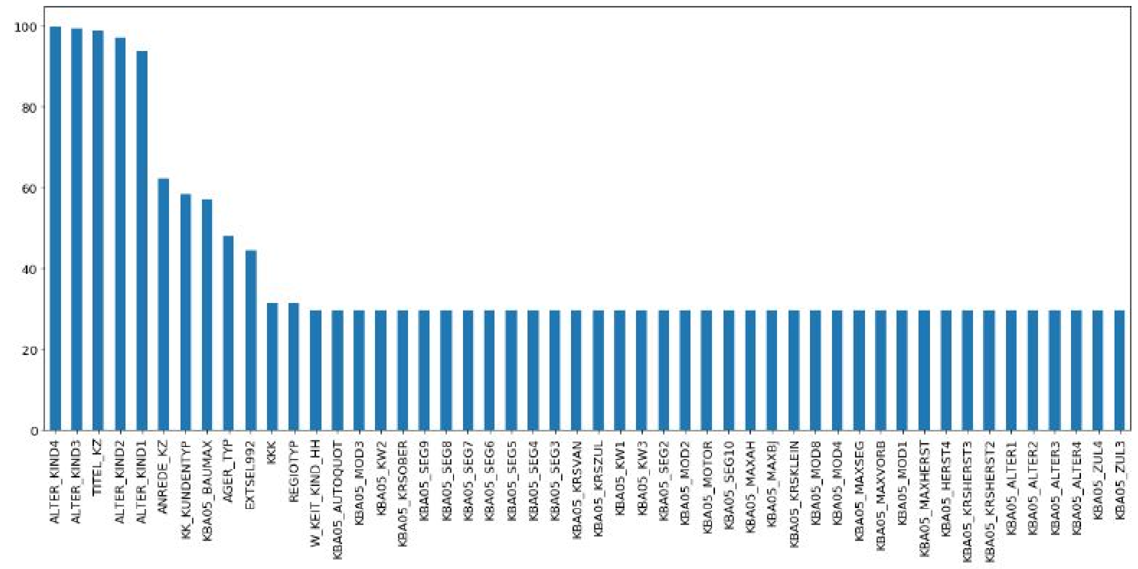
We plot the percentage of missing values(including NaNs) in columns for both Germany dataset and customer dataset. Here are the graphs

<u>Column analysis</u>

Germany population:



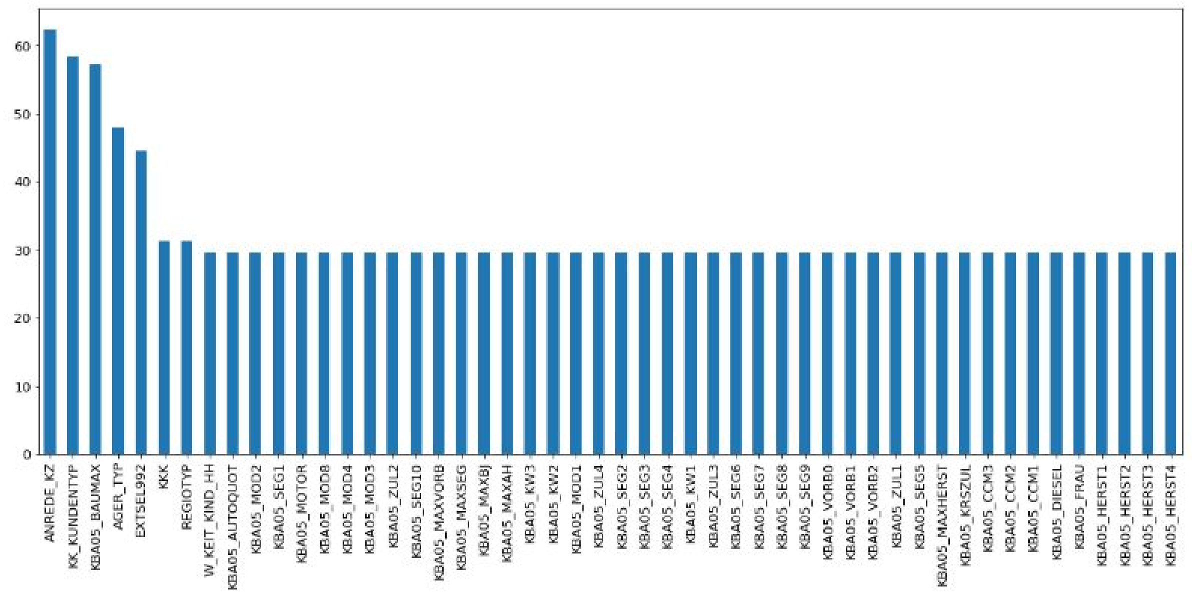
Customer data:



Based on the data, we identify that 65% is a good threshold to retain the data as there is a big gap between cols that have higher missing values than this threshold compared to median.

<u>Row analysis</u>

We do similar analysis for rows.



The data distribution shows that most rows contain 70% valid values. So we use 0.7 as the threshold to drop rows from both datasets.

Drop columns that are unique to customer dataset

We find columns that only present in customer dataset and drop it as it doesn't help us in data clustering and correlation analysis. The identified columns that get dropped are 'AGER_TYP', 'EXTSEL992', 'KK_KUNDENTYP', 'PRODUCT_GROUP', 'CUSTOMER_GROUP', 'ONLINE_PURCHASE'.

Using Data Imputer and Scaling

Next, we use two data filtering techniques provided by Scikit learn.

Data imputer - Substitutes missing values with most frequent values in the column.

Data scaling - 'Centers' the feature data values by deducing mean from each of the features. Also, it applies unit variance to normalize each value. This helps during training such that no single feature gets too much representation because of bigger numerical values and corresponding skews during matrix computations.

Algorithms, Techniques and Methodology

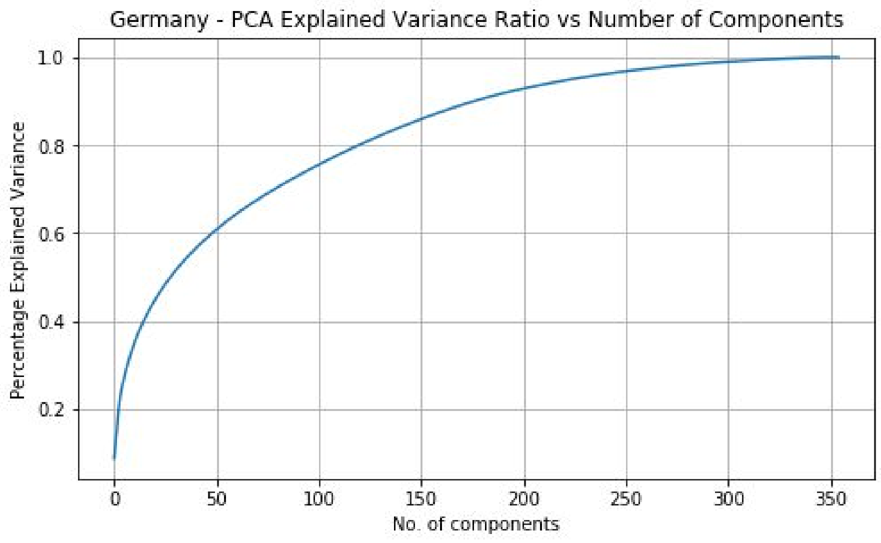
Customer segmentation

After data cleaning, we move to customer segmentation. We perform following steps

Dimensionality Reduction using PCA

After data cleaning, we still left with 355 feature vectors(columns). In real life, there is a lot of cross-correlation among these features and a reduced set of vectors can be computed using dimensionality analysis. These new vectors or principal components are able to retain most of latent features in the data. Essentially PCA allows us to “compress” the data in multi dimensional space.

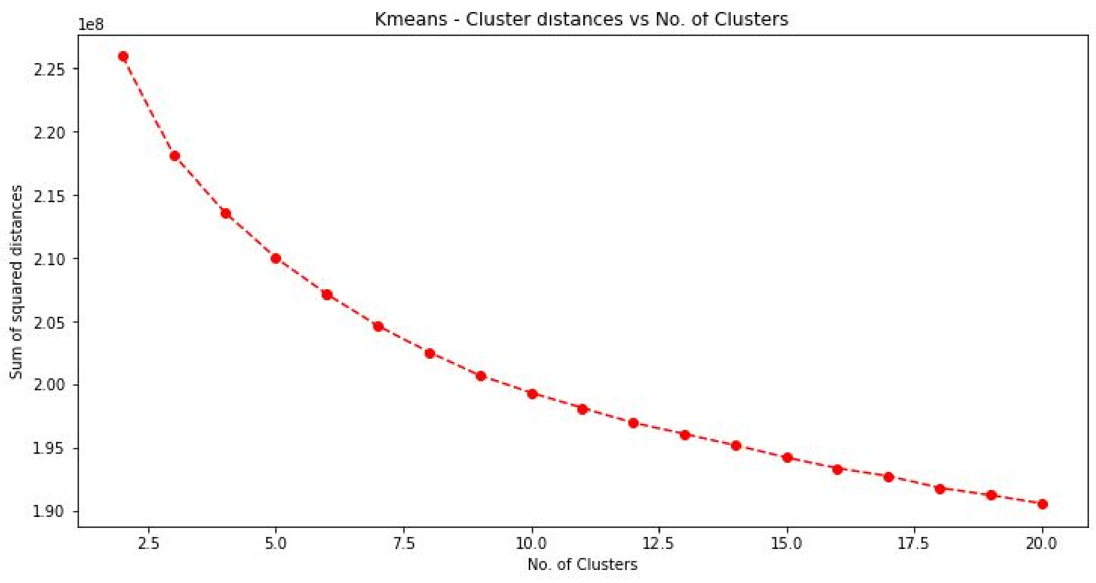
To determine which principal components are most useful, we can measure the explained data variance of each component and select top n.



From the graph, it seems around 200 components, the capture variance for each new component is pretty low. So we select 200 components in our PCA feature vector.

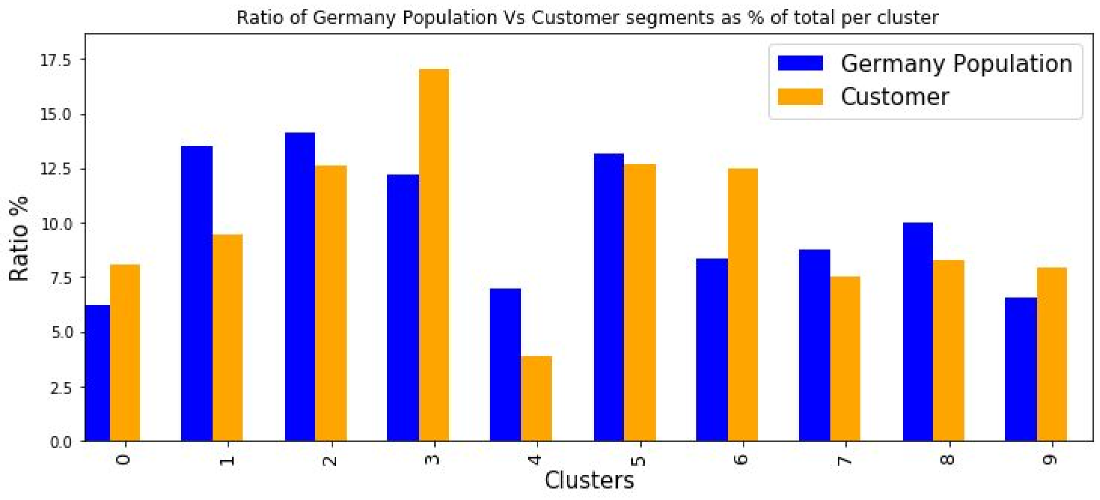
Clustering using K-means

Next, we perform clustering analysis by identifying segments of customers that are close in feature space. As we don't know the number of desired clusters or value of K, we search for optimal value by plotting cluster distance for a range of K values and finding an "elbow" in the graph where increase in K doesn't result in a significant drop in cluster distance. The reason to use this metric is because the distance helps us identify the efficiency of the clustering scheme.



Looking athe the graph, we select 10 for the number of clusters as after 10 the value of distance doesn't reduce too quickly compared to lower values of K.

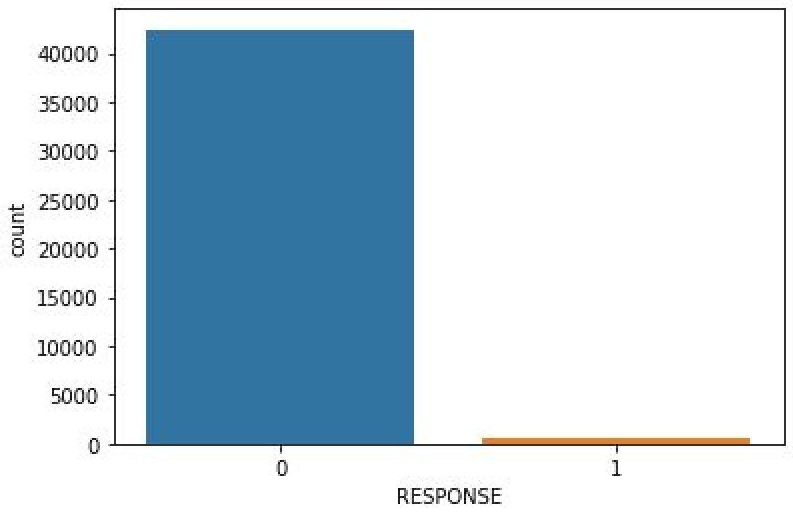
Finally, we plot how many people fall into each cluster for both Germany and Customer dataset. This helps us identify if customer segments are skewed towards certain features in a more pronounced way than the general population of Germany.



Customer acquisition analysis

In the second part of the project, we use supervised learning algorithms to predict whether a person will be a customer or not based on the demographic data. The file 'Udacity_MAILOUT_052018_TRAIN.csv' is provided with the same features as the general population and customers demographic data. An extra column 'RESPONSE' has been provided with this data. The response column indicates whether this person was a customer or not. This data has been cleaned by following similar cleaning and processing steps that were followed for general population and customer data.

The data is highly unbalanced as shown in figure below



Benchmark

The first step in the supervised learning is to set a benchmark, which is the base performance with the simplest model possible. This benchmark is set to compare the results from future steps in order to evaluate the used models. The data is split into train and validation splits and a logistic regression model was trained on unscaled training data and evaluated on the unscaled validation data.

The benchmark score obtained with the logistic regression model – 0.67 AUROC score.

What is logistic regression model and why is it suitable as a baseline to establish a benchmark score ?- Logistic regression is considered a generalized linear model because the outcome always depends on the sum of the inputs and parameters. Due to its simplicity, it is generally easy to train a logistic model as a quick predictive classifier for any binary classification problem. For this reason, using this as a benchmark is logical as any sophisticated models we explore to improve our scores shouldn't perform worse than a bare bones logistic regression mode.

Results

Model Evaluation and Validation

Baselining

After setting the benchmark, the data has been scaled with the standard scaler and is split into training and validation split. Different algorithms have been trained on the training split and have been evaluated on validation split.

The algorithms that have been selected for this step are:

- Logistic Regression
- Decision Tree Classifier
- Random Forest Classifier
- Gradient Boosting Classifier
- AdaBoost Classifier
- XG Boost Classifier - Didn't evaluate due to setup issues.

Here is reasoning behind selecting these algorithms

Logistic Regression - Used as a baseline. Reasoning to use it as baseline is already explained in our benchmark section.

Decision Tree Classifier - While there are many reasons cited for suitability for tree based algorithms for regression problems (https://www.displayr.com/decision-trees-are-usually-better-than-logistic-regression/), the primary reason I chose it because tree based models are next logical choices after logistic regression for binary classification as they strike a good balance between model complexity and predictive power of the model because of ability of a tree model to handle a zigzag separability between positive and negative sample space. In contrast, linear models only perform well when data is linear separable(using straight lines). As we can't assume that our problem domain is linear separable, tree models should give us better flexibility.

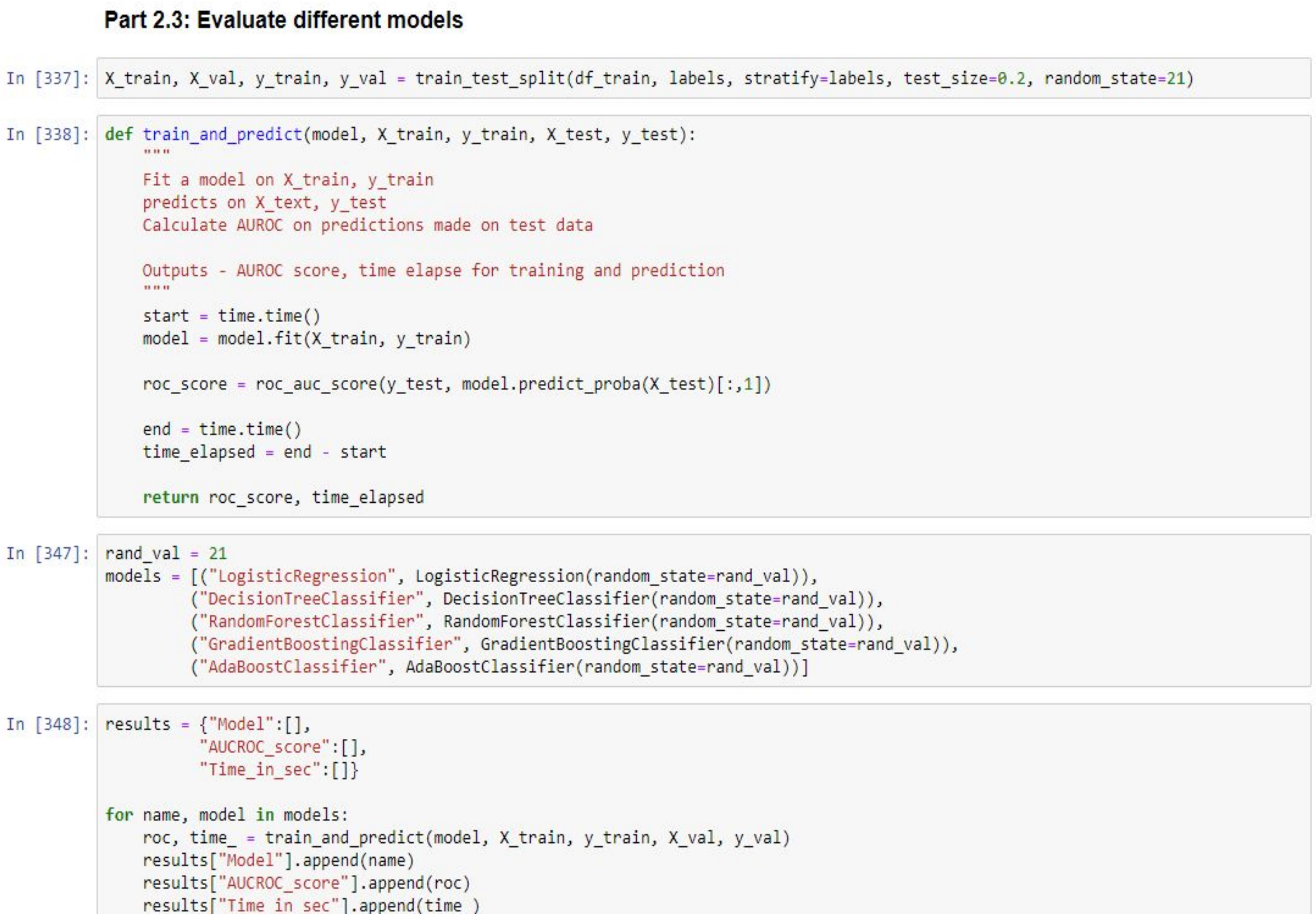
Random Forest Classifier - Random forest lets us tap in to ensemble technique to see if we can improve our score. In general, Random forest takes advantage of decision tree classifiers to the next level where it uses bagging to grow a set of decision trees and use voting between these trees to make the prediction. The reason it improves prediction is because different decision trees overfit the data in a different way, and through voting those differences are averaged out.

Gradient Boosting Classifier - It is also a tree ensemble similar to Random forest but uses a technique called boosting rather than bagging to build trees. Boosting in a general sense helps tackle both bias and variance in predictive error compared to Bagging which mainly improves variance. For this reason, we would like to try this model as part of our portfolio of models to use.

AdaBoost Classifier & XG Boost Classifiers - These are more variations of ensemble models that employ slightly different techniques in Boosting to create ensembles classifiers. We would like to try both to explore which technique can give us a good score on our objective (AUROC).

How did we train and use using these algorithms

The first step I used was test the waters and get a sample score from each of these models without doing any model tuning. This can best be explained by the code from Notebook.



Part 2.3: Evaluate different models

```
In [337]: X_train, X_val, y_train, y_val = train_test_split(df_train, labels, stratify=labels, test_size=0.2, random_state=21)
```

```
In [338]: def train_and_predict(model, X_train, y_train, X_test, y_test):
              """
              Fit a model on X_train, y_train
              predicts on X_text, y_test
              Calculate AUROC on predictions made on test data

              Outputs - AUROC score, time elapse for training and prediction
              """
              start = time.time()
              model = model.fit(X_train, y_train)

              roc_score = roc_auc_score(y_test, model.predict_proba(X_test)[:,1])

              end = time.time()
              time_elapsed = end - start

              return roc_score, time_elapsed
```

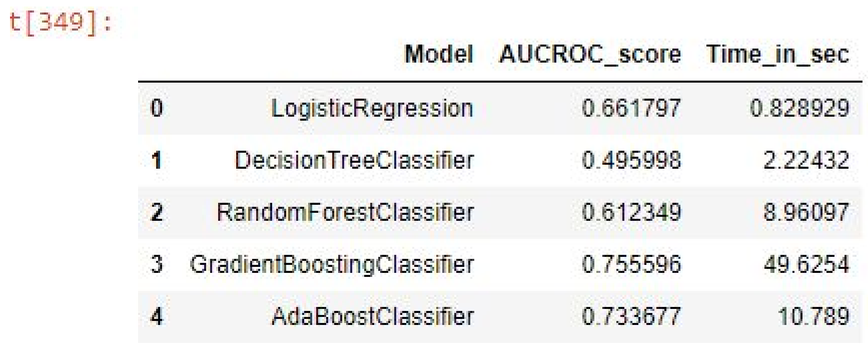
```
In [347]: rand_val = 21
          models = [("LogisticRegression", LogisticRegression(random_state=rand_val)),
                    ("DecisionTreeClassifier", DecisionTreeClassifier(random_state=rand_val)),
                    ("RandomForestClassifier", RandomForestClassifier(random_state=rand_val)),
                    ("GradientBoostingClassifier", GradientBoostingClassifier(random_state=rand_val)),
                    ("AdaBoostClassifier", AdaBoostClassifier(random_state=rand_val))]
```

```
In [348]: results = {"Model":[],
                     "AUCROC_score":[],
                     "Time_in_sec":[]}

          for name, model in models:
              roc, time_ = train_and_predict(model, X_train, y_train, X_val, y_val)
              results["Model"].append(name)
              results["AUCROC_score"].append(roc)
              results["Time_in_sec"].append(time_)
```

As the code shows, I used baseline Sklearn models to do one round of train and validation set(80-20% split) for each model and computed AUROC scores. My goal was to see how these models stack against each other and pick the top winner to do further optimization using hyperparameter tuning.

Here are the results of this evaluation.


t[349]:

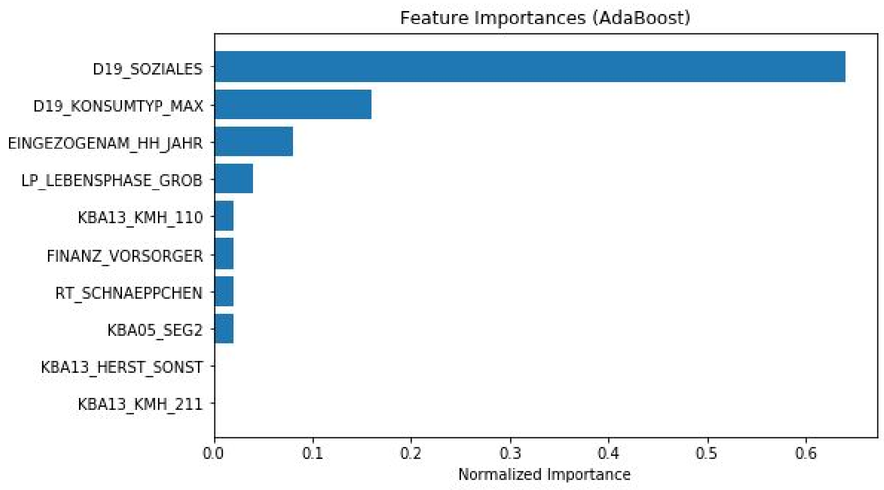
	Model	AUCROC_score	Time_in_sec
0	LogisticRegression	0.661797	0.828929
1	DecisionTreeClassifier	0.495998	2.22432
2	RandomForestClassifier	0.612349	8.96097
3	GradientBoostingClassifier	0.755596	49.6254
4	AdaBoostClassifier	0.733677	10.789

Hyperparameter Tuning

AdaBoost and GradientBoosting provide the highest AUROC score. We pick the AdaBoost for hyperparameter tuning. We use grid search algorithms to find the best performing model.

Feature contribution graph

We use the best AdaBoost model to figure out which features are most representative in identifying the right decision boundary in the dataset.



Final Validation scores

Provided data set:

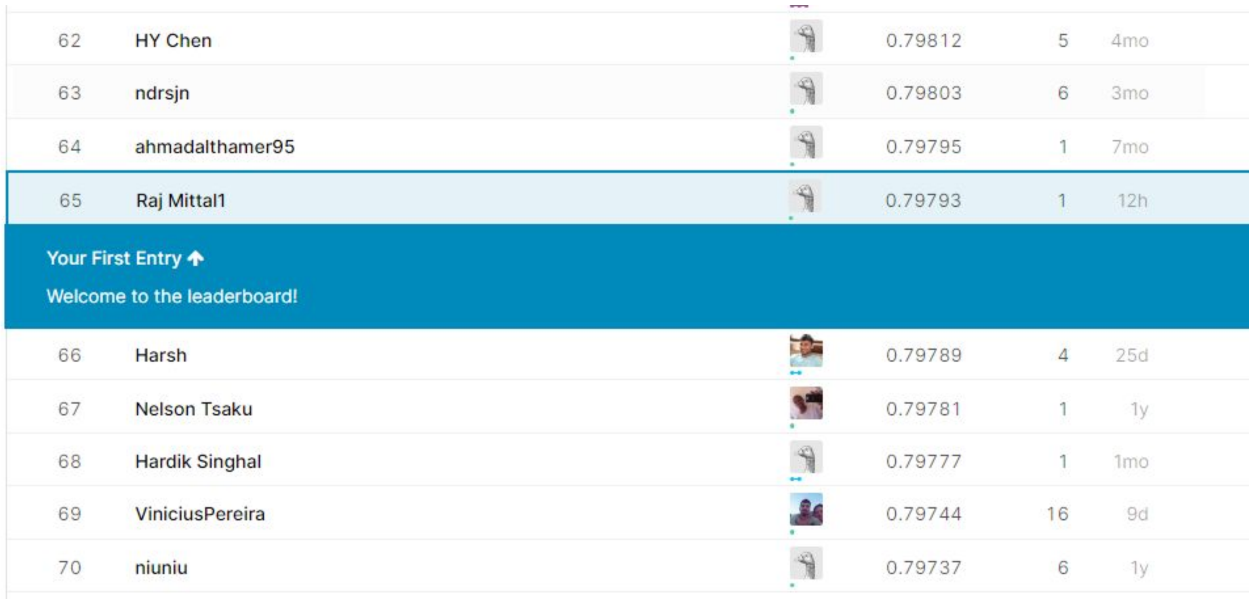
After trying different algorithms and doing hyperparameter tuning, we are able to identify our best AdaBoost model that gives us a significant boost in AUROC score. This gives us

ROC score on validation data: 0.7700

Kaggle submission

The model was used to predict on the Kaggle dataset. Here are the results on the Kaggle validation set.

The resulting score was 0.7979.



62	HY Chen	0.79812	5	4mo
63	ndrsjn	0.79803	6	3mo
64	ahmadalthamer95	0.79795	1	7mo
65	Raj Mittal1	0.79793	1	12h

Your First Entry
Welcome to the leaderboard!

66	Harsh	0.79789	4	25d
67	Nelson Tsaku	0.79781	1	1y
68	Hardik Singhal	0.79777	1	1mo
69	ViniciusPereira	0.79744	16	9d
70	niuniu	0.79737	6	1y

Justification

A combination of technique helped us achieve a good AUROC score on the provided problem dataset.

- Properly cleaning and scaling data prior to the training.
- Having a good benchmark established to identify our baseline.
- Trying out multiple models and choosing the best one in terms of our desired metric (AUROC).
- Using hyperparameter search to optimize our top model candidates.

Future Improvements

Few thoughts on what can be improved for this problem domain

- Using one-hot encoding to map categorical values and use these features in training.
- Using up-samping to balance positive and negative samples.
- Try a bigger search space for hyperparameter tuning. I could only do limited amounts of tuning because of availability of hardware resources.