1 Machine Learning Engineer Nanodegree

1.1 Capstone Project

Sten Ruben Strandheim
August 14th, 2019

2 Definition

2.1 Project Overview

Domain: Media publishing

Background: Intermediates (Google, Facebook, and so forth) try to profit first-party data published by newspapers. In order for the publishers to compete back, they need to customize the way they present their content and ads.

Why: By identifying a reader's demography, a web publisher can expose more customized content to the visitor, such as premium content or ads.

My dataset is from a Norwegian newspaper, made by my colleague. I will try to identify Pensioners in this dataset, because they have a specific consumption and reading pattern.

By doing this, the publisher can

1. Leverage on its first-party content in competition with other intermediates.
2. Sell targeting of ads, in order to increase the revenue potential.

The Washington University paper "How well can machine learning predict demographics of social media users?" (https://arxiv.org/ftp/arxiv/papers/1702/1702.01807.pdf) concludes that demographic traits such as age and race/ethnicity are more challenging to predict.

I will test which combination of techniques provides a better precision when classifying Pensioners:

- Normalization/Transforming
- Downsampling by DBscan clustering: Outliers will be removed
- Upsampling by ADASYN
- Hyperparameter tuning for algorithms GaussianNB, Logistic Regression, Random Forest, Gradient Boosting, XGBoost

2.2 Problem Statement

Problem: Identify pensioners at a level that justify spent time.

As to illustrate the level of difficulty, calculate:

- general % of pensioners in the dataset
- specific % of pensioners that visited each of the magazines

2.3 Metrics

Verify results from each algorithm with:

- F_beta scoring, providing an mix of Precision and Recall:
  - Precision
    - P = TP/(TP+FP)
    - Good content customization for the readers. Argument for higher ad prices.
  - Recall
    - R = TP/(TP+FN)
    - Optimal distribution of the publisher's content and ads
  - F_beta:
    - F_beta = (1+beta^2)*P*R/(R+P*beta^2)
    - Beta=0.5 provides an accepted mix with emphasis on precision, which I think will suite an imbalanced dataset
- Show result with ROC curve and ROC_AUC
- Show curves for cumulative gain and lift to assess business value
- Evaluate % of pensioners in predicted dataset.

3 Analysis

3.1 Data Exploration

3.1.1 Raw dataset to be analyzed

Contains data from two sources:

- Collected over a timeframe from the publisher's cookies, each visitor gets its own ID from the cookie
- Demographic data is collected through surveys among visitors, conducted by the same cookies providing the same visitor ID.

Data was then (inner-)joined, aggregated and prepped in various age segments. No need for data impute. Pensioners were here stated to be aged 60 and above.

Aggregate: Number of times the customer visited a given kind of section in the various magazines/newspapers owned by the publisher, over a timeframe. In this dataset, the sections has been anonymized for it to be used in this project. These sections may be sports, culture, celebrities, etc

The visitor's choice of sections may help indicate the age, as the reader-ID from the cookies doesn't disclose demography in daily use.

Number of rows: 4214
Number of features (magazine sections): 12.
Target: Pensioner is marked as 1, else 0

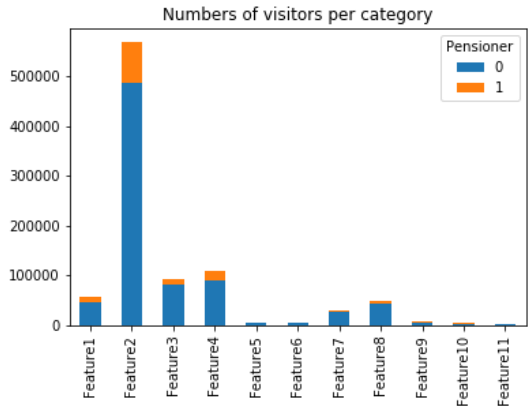
3.1.2 Imbalanced dataset: Target baseline

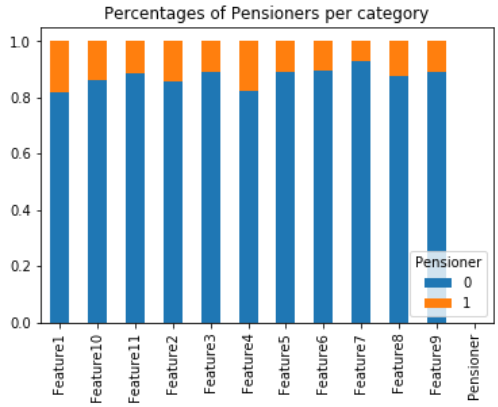
Illustrate the level of imbalance in the dataset:

- general % of pensioners in the dataset
- specific % of pensioners that visited each of the sections

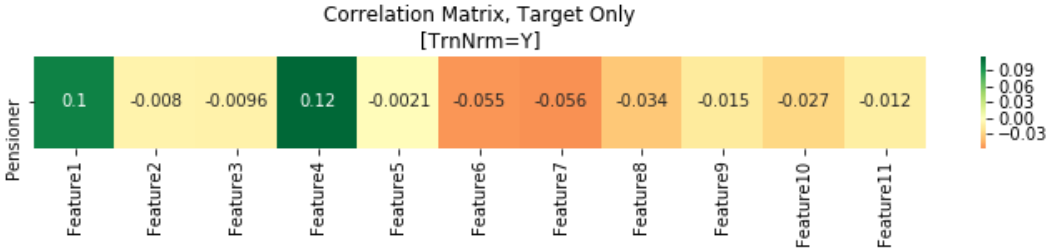
3.2 Exploratory Visualization

Target baseline for entire dataset is ~13%:





Target is slightly correlating with the rest of the dataset, but not with very high correlation factors:



3.3 Algorithms and Techniques

3.3.1 Algorithms

GaussianNB classifier:
The Naive Bayes Classifier is used as an example of a relatively "simple" model to test the more complex models against. The GaussianNB is based on Bayes' theorem, and treats the variables independently.

Logistic Regression classifier:
A popular algorithm for classification problems, which uses the same technique as linear regression, but applies a sigmoid function in order to be able to classify the classes, with a probability between 0 and 1.

Random Forest classifier:
A tree based algorithm, more sophisticated than ordinary decision trees, in that it randomly samples training points, so that it does not so easily overfit.

Gradient Boosting classifier:
The basic ensamble boosting algorithm based on trees. XGboost builds on this, and a comparison to XGboost would be interesting.

XGBoost classifier:
This is a gradient boosting algorithm which is scalable and fast, and has a proven track-record for solving classification problems. Fitting a number of weak learners in the form of decision trees, it makes classification based on the combination of said learners, while reducing overfitting.

SVC classifier:

The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data.

DBscan clustering

Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996. A very common clustering algorithms.

SMOTE upsampling

SMOTE is an oversampling technique that generates synthetic samples from the minority class. It is used to obtain a synthetically class-balanced or nearly class-balanced training set, which is then used to train the classifier.

PCA

- pca.explained_variance_ratio_: Used for PCA results
- pca.components_: Used for Biplot

Learning curves

Sklearn methods used to display learning curves and model coplexity

- learning_curve. Used during ModelLearning
- validation_curve. Used during ModelComplexity

make scorer

Customized scorer called f05. Used for GridSearch

- Scorer: fbeta_score. Beta=0.5

GridSearchCV

Tests all combinations of parameters supplied by the user. Returns the combination that represent the best scoring specified by the user.

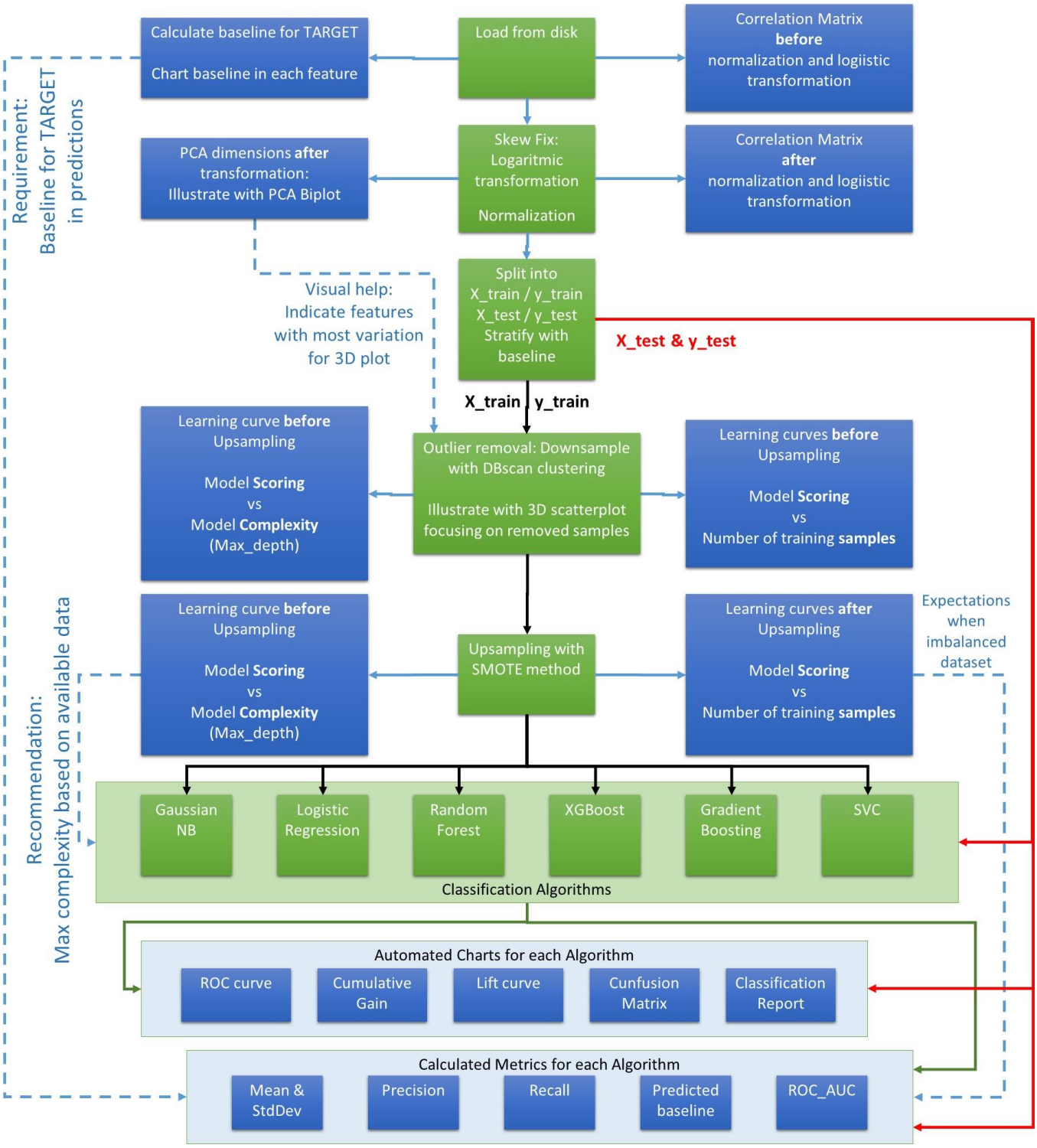
- Scoring: Customised scoring model f05

  Cross validation

  Used to provide more robust metrics, like measuring performance while working on training set alone

- n=5

3.3.2 Techniques



Split into training and test datasets:

- Stratification while splitting: **Similar % pensioners in both datasets.**
- 80% train / 20% test split: As the dataset is small and imbalanced, as much as possible data should reside in the training dataset. This gives me 3371 rows in X_train and 843 rows in X_test.

Correlation Matrix:

- Each feature is correlated with each and all of the other features. **Shows correlation between each pair of every feature**

PCA Biplot:

- Shows two things simultaneously: **Indicates the most important features.**
  1. Dots: Each sample scored along the two most Principal Components
  2. Vectors: Component loadings of features (the % of variance in that feature, explained by the vector's length)

Normalization and Logarithmic Transformation:

- Centering of values around a given center-value.
- Logarithmic reduction of large values

Downsampling

Removal of non-Target outliers by use of DBscan clustering.

- DBSCAN - Density-Based Spatial Clustering of Applications with Noise. Finds core samples of high density and expands clusters from them. Good for data which contains clusters of similar density.
- Un-clustered Target samples are preserved, by being allocated to the basic cluster.
- The rest of the un-clustered samples are interpreted as outliers, and removed.

Upsampling

- Use SMOTE to **create a new minority classes** from the existing dataset. These are not exact copies of the original rare samples, but placed in-between them. In this way randomness can be introduced into the dataset to make sure that there won't be an overfitting problem.

Learning curves

- Model scoring vs. Number of training samples: **Learning Performance.** Expected balance: Model precision increases with model complexity, but needs more data to achieve desired precision.
- Model scoring vs. Model Complexity: **Complexity Performance.** Expected precision, while the difference between curves for training and testing precision is kept below an accepted level.

3.4 Benchmark

Among all the unique readers that returned the demographic survey, there are 13% pensioners. This baseline is the benchmark I will align my results to.

All configurations that predict a baseline that differ very much from the benchmark baseline will be rejected.

4 Methodology

I will suggest a classification model that is capable of predicting readers of age above 60. This may be more challenging as the dataset will be unbalanced, as this age group is less active on the web than younger groups.

4.1 Data Preprocessing

No further preprocessing of the raw data is necessary, as there is only one dataset and all data is populated.

4.2 Implementation process

4.2.1 Metrics

Metrics calculated by use of sklearn's shipped functions:

- roc_auc_score
- precision_score
- recall_score

Metrics calculated as a function of precision_score and recall_score

- f_beta=(1+beta^2)*recision*recall/(recall+precision*beta^2) beta=0.5
- baseline_predicted: Count_of_Target / Total_count

4.2.2 Algorithms

Most of the algorithms have more options and features than I have employed. Especially the classifiers may have options I haven't used.

I have not used any options for upsampling method SMOTE

I guessed input values for the clustering algorithm DBscan

- For normalized and logarithmic transformed dataset:
  epsilon=0.54 / min_samples=3
- For dataset not normalized and transformed:
  epsilon=80 / min_samples=3

This gave about 5-10 different clusters, and about 40-65 samples that were removed as outliers.

I had to run the learning curves before I was able to set a parameter Max_Depth for those classifiers that has this parameter.

I had to run the PCA and Biplot before I was able to know which Features to use in order to make an optimum visualizing of the DBscan clustering algorithm.

4.2.3 Techniques

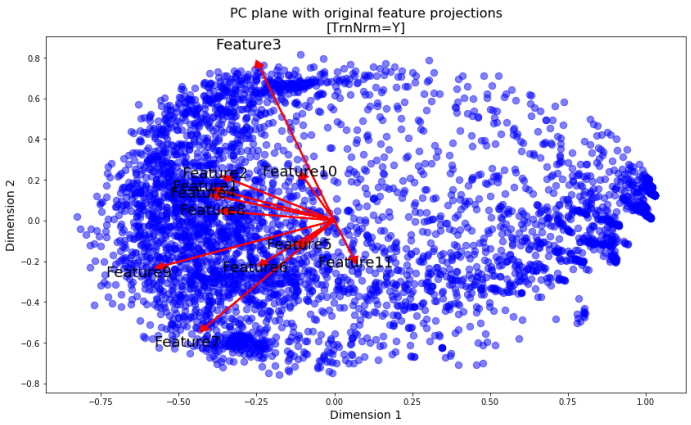
The downsampling routine with DBscan clustering is made of two basic ideas:

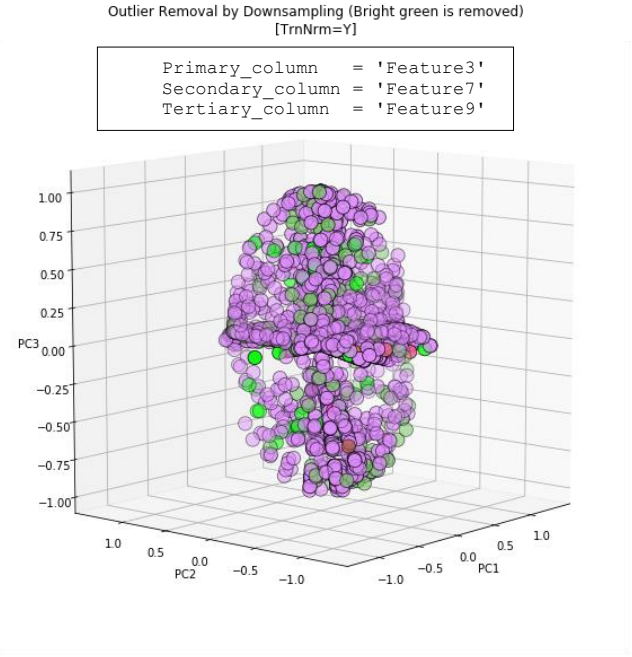
1. DBscan suggests samples not clustered
2. Samples that are not clusteres may be of the rare Target. In these cases the sample must be preserved: The sample gets labeled with the first cluster made by DBscan.

4.3 Refinement process

The goal is to find an optimal configuration before I start tuning my hyperparameters.

PCA Biplot:
Told me which Features to use to best visualize results from DBscan clustering, as these explains the most of the variance after Normalization and Logarithmic Transformation:

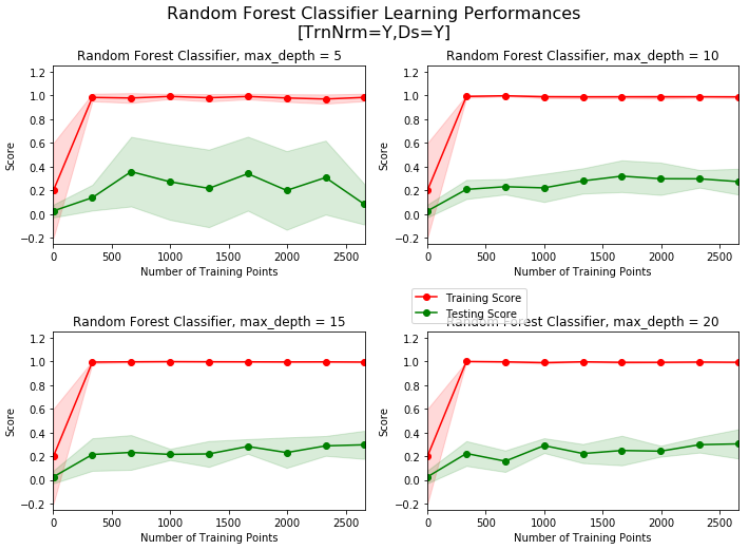




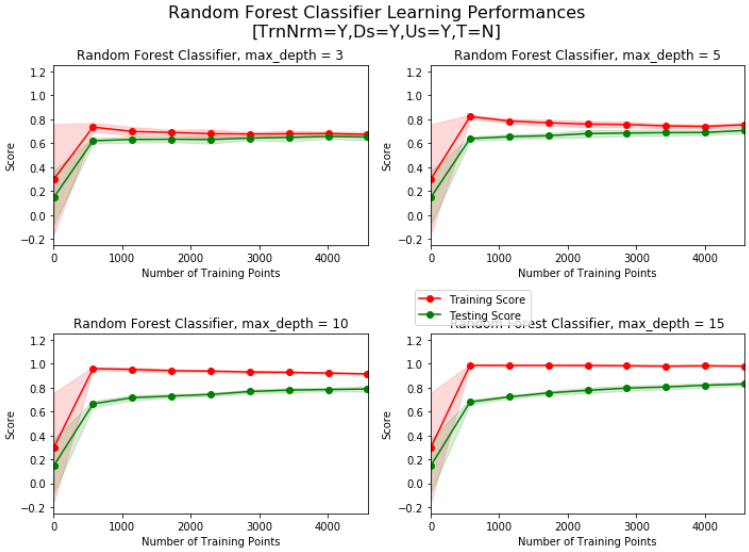
Learning curves

Gives an initial expectation towards:

- Expected level of precision for most models. **What I learned:** My dataset has too few samples to accommodate a requirement of high scores
    1. High performance requires complex models that in turn requires more data. I have 3371 rows in my training dataset.

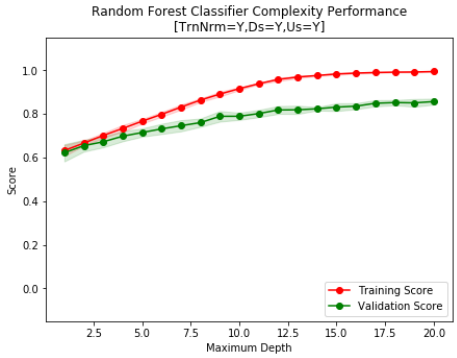


2. After upsampling, a lot more of the rare Target was introduced, and I could get a sense of what to expect:



For classification algorithms based on trees, I will not be able to make use of a higher Max_Depth than 3, and the precision will not be more than 0.6. Remember that these figures are "doped" by the upsampling, maybe expected precision must be reduces even more.

- Maximum complexity supported by a limited dataset. **What I learned:** At higher complexity (Max_Depth >3) my models will overfit.



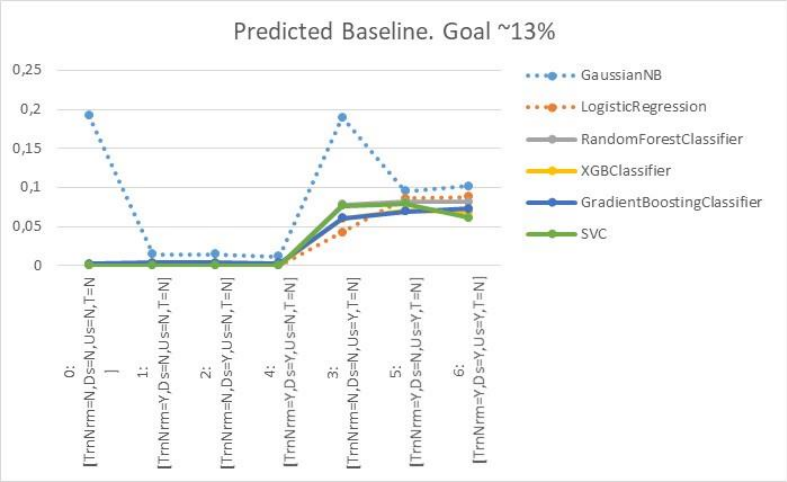
4.3.1 Compare the F_0.5 score between separate runs through the flowchart:

Run with different enabling of each of these techniques:

- Normalize and Logarithmic Transformation
- Downsampling for outlier removal
- Upsampling to compensate for an imbalanced dataset
- Tune hyperparameters with GridSearch

Configuration explained	Transform and Normalize	Downsampling	Upsampling	Train with GridSearch
0: [TrnNrm=N,Ds=N,Us=N,T=N]	N	N	N	N
1: [TrnNrm=Y,Ds=N,Us=N,T=N]	Y	N	N	N
2: [TrnNrm=N,Ds=Y,Us=N,T=N]	N	Y	N	N
3: [TrnNrm=N,Ds=N,Us=Y,T=N]	N	N	Y	N
4: [TrnNrm=Y,Ds=Y,Us=N,T=N]	Y	Y	N	N
5: [TrnNrm=Y,Ds=N,Us=Y,T=N]	Y	N	Y	N
6: [TrnNrm=Y,Ds=Y,Us=Y,T=N]	Y	Y	Y	N
7: [TrnNrm=Y,Ds=Y,Us=Y,T=Y]	Y	Y	Y	Y

Setups that comply with the benchmark baseline criterion:



Setups that provides ~ 0% in predicted baseline:

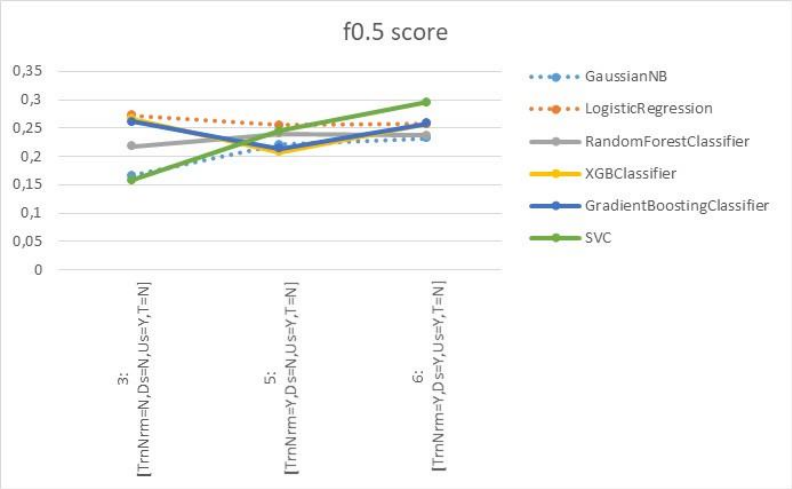
0: [TrnNrm=N,Ds=N,Us=N,T=N]
1: [TrnNrm=Y,Ds=N,Us=N,T=N]
2: [TrnNrm=N,Ds=Y,Us=N,T=N]
4: [TrnNrm=Y,Ds=Y,Us=N,T=N]

Required baseline is ~ 13%.

As one can see:
Upsampling is not member of all these setups

These setups will be removed from further analysis

Setups with the ultimate f0.5 score, before hyperparameter tuning:



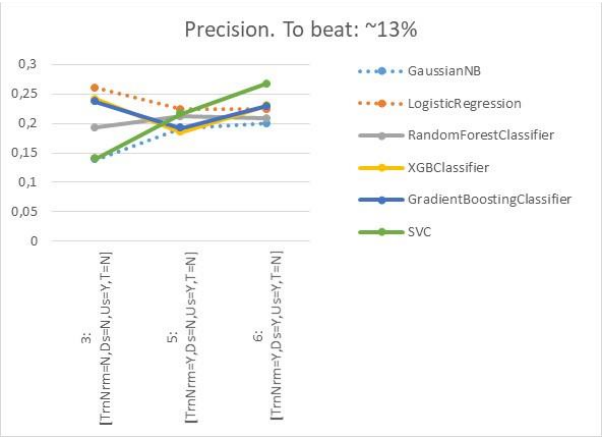
3: [TrnNrm=N,Ds=N,Us=Y,T=N]
5: [TrnNrm=Y,Ds=N,Us=Y,T=N]
6: [TrnNrm=Y,Ds=Y,Us=Y,T=N]

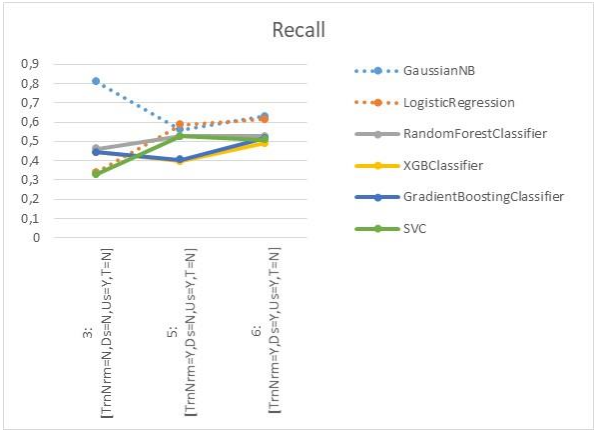
As one can see:
Upsampling is member of all these setups

Downsampling is member only of the most promising setup.

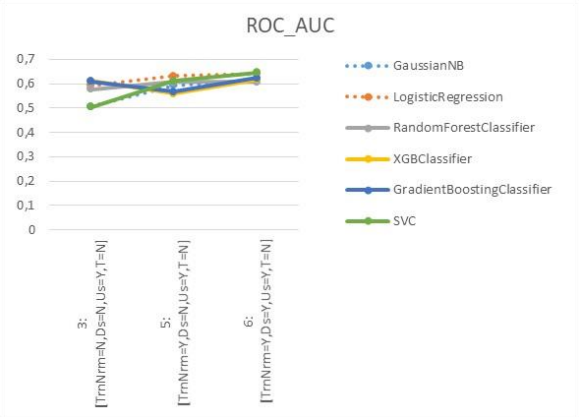
Some Classifiers perform better without TrnNrm: LogisticRegression and maybe GradientBoosting,

Precision and recall on qualified setups:





ROC Area Under Curve on qualified setups



Selected setup to proceed with:
6: [TrnNrm=Y,Ds=Y,Us=Y,T=N]

This setup uses all techniques:

- Normalize and LogTransformation
- Downsampling for outlier removal
- Upsampling for imbalanced dataset

Scores on selected setup, before GridSearch:

Radetiketter	GaussianNB	LogisticRegression	RandomForestClassifier	XGBClassifier	GradientBoostingClassifier	SVC
f_beta	0,23166	0,256975	0,236967	0,257827	0,258319	0,295316
precision	0,2	0,224359	0,208333	0,230453	0,229572	0,267281
pred_base	0,101695	0,0881356	0,0813559	0,0686441	0,0725989	0,0612994
recall	0,631579	0,614035	0,526316	0,491228	0,517544	0,508772
roc_auc	0,618259	0,641037	0,606779	0,617356	0,622969	0,645332

ROC_AUC is somewhat above 0.5 for all models. This is encouraging, given the small dataset and imbalanced dataset.

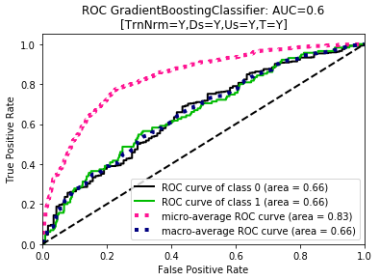
Achieved scores after GridSearch:

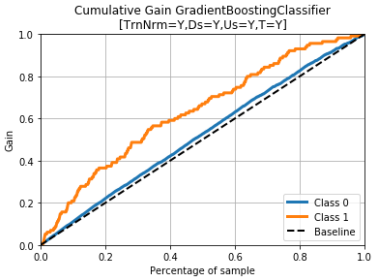
Radetiketter	GaussianNB	LogisticRegression	RandomForestClassifier	XGBClassifier	GradientBoostingClassifier	SVC
f_beta	0,221953	0,255061	0,282413	0,253165	0,27668	0,230769
precision	0,19573	0,225	0,26506	0,241611	0,26087	0,203571
pred_base	0,0793785	0,079096	0,0468927	0,0420904	0,0454802	0,079096
recall	0,478261	0,547826	0,382609	0,313043	0,365217	0,495652
roc_auc	0,583911	0,624875	0,607513	0,578912	0,600878	0,594667

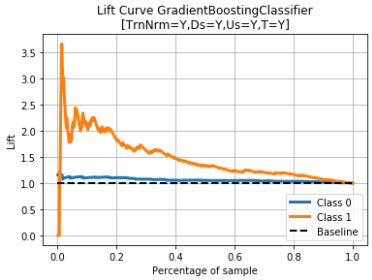
The ultimate metric f0.5 dropped for almost every Classifier during GridSearch, only GradientBoostingClassifier performed better.

Champion model:

GradientBoostingClassifier after Normalize+LogTransformation, Downsampling and Upsampling







By ranging predicted Targets along it's probability, one can see that targeting only the 40% with highest probability can give between 1.5 and 2 times better hitrate than random selecting 40% from all of the predicted Targets. Thus, more than half the effort is saved when using this model on the most promising part of the population.

5 Results

5.1 Model Evaluation and Validation

Three Classifiers range on the top, with respect to f0.5 and precision:

	Radetiketter	XGBClassifier	GradientBoostingClassifier	SVC
f_beta	0,257827	0,258319	0,295316
precision	0,230453	0,229572	0,267281
roc_auc	0,617356	0,622969	0,645332

XGBClassifier

	0	1	accuracy	macro avg	weighted avg
f1-score	0.794251	0.252747	0.677343	0.523499	0.719738
precision	0.882353	0.185484	0.677343	0.533918	0.786461
recall	0.722146	0.396552	0.677343	0.559349	0.677343
support	727.000000	116.000000	0.677343	843.000000	843.000000

Confusion Matrix:

	false	true
negative	525	70
positive	202	46

GradientBoostingClassifier

	0	1	accuracy	macro avg	weighted avg
f1-score	0.798491	0.260388	0.683274	0.529439	0.724446
precision	0.884615	0.191837	0.683274	0.538226	0.789286
recall	0.727648	0.405172	0.683274	0.566410	0.683274
support	727.000000	116.000000	0.683274	843.000000	843.000000

Confusion Matrix:

	false	true
negative	529	69
positive	198	47

SVC

	0	1	accuracy	macro avg	weighted avg
f1-score	0.785714	0.306533	0.672598	0.546123	0.719777
precision	0.901961	0.216312	0.672598	0.559136	0.807613
recall	0.696011	0.525862	0.672598	0.610937	0.672598
support	727.000000	116.000000	0.672598	843.000000	843.000000

Confusion Matrix:

	false	true
negative	506	55
positive	221	61

5.2 Justification

Once an imbalanced dataset is present, the performance will drop. If performance drops less than the balance, one can still focus on precision or – as stated in the Project Definition – the f0.5 score which balances precision and recall with emphasis on precision.

By using Learning Curves I found the optimal Max_Depth that is a parameter for several of my selected Classifiers. During this work I sometimes overshot these limitations and saw that performance dropped.

This setup performed this way, by providing precision above Target baseline. Even better, there is proof that this model still can save effort when deployed.

Even though results without GridSearch was promising, the results from the GridSerach was worse. This tells me that there are small margins, and I have not yet discovered where to fine-tune my models.

6 Conclusion

I select GradientBoostingClassifier as my champion model. It has a good price lift, it is on top with respect to predicted baseline, and all classical metrics are somewhat better than random decision.

When dealing with a imbalanced and small dataset, a good way of spending effort is to delve into learning curves.

6.1 Improvement

Proposed improvements:

- Optimize number of outliers, by tuning the DBscan parameters
- Find better value ranges for GridSearch
- Explore further on Model Complexity in the Learning Curves: There are more parameters that are common for several Classifiers.

7 Appendix: Metrics

Metrics collected after different setup configurations:

Configuration	Metrics	GaussianNB	LogisticRegression	RandomForestClassifier	XGBClassifier	GradientBoostingClassifier	SVC
0: [TrnNrm=N,Ds=N,Us=N,T=N]	acc_mean	0,268065	0,856493	0,863594	0,857662	0,844602	0,864777
0: [TrnNrm=N,Ds=N,Us=N,T=N]	acc_stddev	0,0337586	0,00824006	0,0032717	0,0122729	0,0156	0,00174326
0: [TrnNrm=N,Ds=N,Us=N,T=N]	f_beta	0,160099	0	0	0,0704225	0,136986	0
0: [TrnNrm=N,Ds=N,Us=N,T=N]	precision	0,133431	0	0	0,285714	0,5	0
0: [TrnNrm=N,Ds=N,Us=N,T=N]	pred_base	0,192059	0,000281611	0	0,00197128	0,00225289	0,000281611
0: [TrnNrm=N,Ds=N,Us=N,T=N]	recall	0,798246	0	0	0,0175439	0,0350877	0
0: [TrnNrm=N,Ds=N,Us=N,T=N]	roc_auc	0,493773	0,499314	0,5	0,505343	0,5148	0,49931
1: [TrnNrm=Y,Ds=N,Us=N,T=N]	acc_mean	0,80429	0,863581	0,863581	0,867146	0,852888	0,863581
1: [TrnNrm=Y,Ds=N,Us=N,T=N]	acc_stddev	0,0276782	0,00039686	0,00039686	0,011567	0,0122973	0,00039686
1: [TrnNrm=Y,Ds=N,Us=N,T=N]	f_beta	0,185759	0,037037	0	0,0381679	0,122699	0
1: [TrnNrm=Y,Ds=N,Us=N,T=N]	precision	0,230769	0,2	0	0,25	0,333333	0
1: [TrnNrm=Y,Ds=N,Us=N,T=N]	pred_base	0,0146934	0,00141283	0	0,00113026	0,00339079	0
1: [TrnNrm=Y,Ds=N,Us=N,T=N]	recall	0,104348	0,00869565	0	0,00869565	0,0347826	0
1: [TrnNrm=Y,Ds=N,Us=N,T=N]	roc_auc	0,524701	0,501601	0,5	0,502287	0,511897	0,5
2: [TrnNrm=N,Ds=Y,Us=N,T=N]	acc_mean	0,80429	0,863581	0,863581	0,867146	0,852888	0,863581
2: [TrnNrm=N,Ds=Y,Us=N,T=N]	acc_stddev	0,0276782	0,00039686	0,00039686	0,011567	0,0122973	0,00039686
2: [TrnNrm=N,Ds=Y,Us=N,T=N]	f_beta	0,185759	0,037037	0	0,0381679	0,122699	0
2: [TrnNrm=N,Ds=Y,Us=N,T=N]	precision	0,230769	0,2	0	0,25	0,333333	0
2: [TrnNrm=N,Ds=Y,Us=N,T=N]	pred_base	0,0146934	0,00141283	0	0,00113026	0,00339079	0
2: [TrnNrm=N,Ds=Y,Us=N,T=N]	recall	0,104348	0,00869565	0	0,00869565	0,0347826	0
2: [TrnNrm=N,Ds=Y,Us=N,T=N]	roc_auc	0,524701	0,501601	0,5	0,502287	0,511897	0,5
3: [TrnNrm=N,Ds=N,Us=Y,T=N]	acc_mean	0,573415	0,862398	0,863581	0,863581	0,862398	0,863581
3: [TrnNrm=N,Ds=N,Us=Y,T=N]	acc_stddev	0,16947	0,00223441	0,00039686	0,00750634	0,00435864	0,00039686
3: [TrnNrm=N,Ds=N,Us=Y,T=N]	f_beta	0,166131	0,272727	0,218107	0,265902	0,261538	0,158465
3: [TrnNrm=N,Ds=N,Us=Y,T=N]	precision	0,138599	0,26	0,192727	0,241706	0,237209	0,140221
3: [TrnNrm=N,Ds=N,Us=Y,T=N]	pred_base	0,189655	0,0423968	0,0777275	0,0596382	0,0607688	0,0765969
3: [TrnNrm=N,Ds=N,Us=Y,T=N]	recall	0,808696	0,33913	0,46087	0,443478	0,443478	0,330435
3: [TrnNrm=N,Ds=N,Us=Y,T=N]	roc_auc	0,50737	0,593329	0,577962	0,611849	0,609102	0,50519
4: [TrnNrm=Y,Ds=Y,Us=N,T=N]	acc_mean	0,832739	0,864777	0,864777	0,851731	0,843418	0,864777
4: [TrnNrm=Y,Ds=Y,Us=N,T=N]	acc_stddev	0,0168936	0,00174326	0,00174326	0,00815248	0,00795125	0,00174326
4: [TrnNrm=Y,Ds=Y,Us=N,T=N]	f_beta	0,124113	0,0423729	0	0,0384615	0,0649351	0
4: [TrnNrm=Y,Ds=Y,Us=N,T=N]	precision	0,166667	1	0	0,25	0,2	0
4: [TrnNrm=Y,Ds=Y,Us=N,T=N]	pred_base	0,0118177	0,000281373	0	0,00112549	0,00281373	0
4: [TrnNrm=Y,Ds=Y,Us=N,T=N]	recall	0,0614035	0,00877193	0	0,00877193	0,0175439	0
4: [TrnNrm=Y,Ds=Y,Us=N,T=N]	roc_auc	0,506696	0,504386	0,5	0,502328	0,503285	0,5
5: [TrnNrm=Y,Ds=N,Us=Y,T=N]	acc_mean	0,831507	0,861212	0,862396	0,855316	0,850554	0,862403
5: [TrnNrm=Y,Ds=N,Us=Y,T=N]	acc_stddev	0,013881	0,00597216	0,00439292	0,0110698	0,0051858	0,00181698
5: [TrnNrm=Y,Ds=N,Us=Y,T=N]	f_beta	0,22139	0,255255	0,240536	0,207581	0,214416	0,245177
5: [TrnNrm=Y,Ds=N,Us=Y,T=N]	precision	0,192308	0,223684	0,211806	0,185484	0,191837	0,216312
5: [TrnNrm=Y,Ds=N,Us=Y,T=N]	pred_base	0,0953456	0,0857546	0,0812412	0,0699577	0,0691114	0,0795487
5: [TrnNrm=Y,Ds=N,Us=Y,T=N]	recall	0,560345	0,586207	0,525862	0,396552	0,405172	0,525862
5: [TrnNrm=Y,Ds=N,Us=Y,T=N]	roc_auc	0,592415	0,630793	0,60681	0,559349	0,56641	0,610937
6: [TrnNrm=Y,Ds=Y,Us=Y,T=N]	acc_mean	0,836346	0,861227	0,864777	0,850547	0,841066	0,864777
6: [TrnNrm=Y,Ds=Y,Us=Y,T=N]	acc_stddev	0,0241369	0,00772311	0,00174326	0,0112819	0,0123537	0,00174326
6: [TrnNrm=Y,Ds=Y,Us=Y,T=N]	f_beta	0,23166	0,256975	0,236967	0,257827	0,258319	0,295316
6: [TrnNrm=Y,Ds=Y,Us=Y,T=N]	precision	0,2	0,224359	0,208333	0,230453	0,229572	0,267281
6: [TrnNrm=Y,Ds=Y,Us=Y,T=N]	pred_base	0,101695	0,0881356	0,0813559	0,0686441	0,0725989	0,0612994
6: [TrnNrm=Y,Ds=Y,Us=Y,T=N]	recall	0,631579	0,614035	0,526316	0,491228	0,517544	0,508772
6: [TrnNrm=Y,Ds=Y,Us=Y,T=N]	roc_auc	0,618259	0,641037	0,606779	0,617356	0,622969	0,645332
7: [TrnNrm=Y,Ds=Y,Us=Y,T=Y]	acc_mean	0,831516	0,633387	0,738805	0,797707	0,71632	0,863581
7: [TrnNrm=Y,Ds=Y,Us=Y,T=Y]	acc_stddev	0,017783	0,0148907	0,0177462	0,0182627	0,0194098	0,00039686
7: [TrnNrm=Y,Ds=Y,Us=Y,T=Y]	f_beta	0,221953	0,255061	0,282413	0,253165	0,27668	0,230769
7: [TrnNrm=Y,Ds=Y,Us=Y,T=Y]	precision	0,19573	0,225	0,26506	0,241611	0,26087	0,203571
7: [TrnNrm=Y,Ds=Y,Us=Y,T=Y]	pred_base	0,0793785	0,079096	0,0468927	0,0420904	0,0454802	0,079096
7: [TrnNrm=Y,Ds=Y,Us=Y,T=Y]	recall	0,478261	0,547826	0,382609	0,313043	0,365217	0,495652
7: [TrnNrm=Y,Ds=Y,Us=Y,T=Y]	roc_auc	0,583911	0,624875	0,607513	0,578912	0,600878	0,594667