1 Machine Learning Engineer Nanodegree

1.1 Capstone Project

Sten Ruben Strandheim
August 15th, 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, recorded over two moths 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.
- In order to be a relevant newspaper for everybody, avoid identifying younger persons as Pensioner.

Proposed problem solution:

1. First: Explore dataset in order to

a. Verify existence of variance in the data that relate to my target.
b. Find extreme values or situations, in order to plan how to deal with these situations.
c. As to illustrate the level of difficulty/imbalance, calculate:
    - general % of pensioners in the dataset
    - specific % of pensioners that visited each of the magazines

2. Split dataset into Train and Test dataset. As this is a imbalanced dataset, keep as much as possible into the Train dataset.

3. Find and remove outliers. This will be done together with a method I have called Downsampling, where only outliers from non-Pensioners are deleted.

4. Look at the learning curves, and explore how Upsampling changes their focus.
    a. Take advice from the Learning Curves on model complexity
    b. Be aware of doping-effect on precision, introduced by Upsampling

5. Fit each classifier to the Train dataset, and extract metrics scoring and relevant curves as ROC, Cumulative Gain and Lift Curve.

6. This project will use the metrics after fitting to compare the effects from each of the methods designed for preparing the data:
    a. Transformation
    b. Downsampling
    c. Upsampling.

7. Therefore multiple runs with different enabling of methods (arranged in different notebooks with the same random-state) will be performed and metrics being collected.

2.3 Metrics

2.3.1 F_beta scoring, with beta=0.5

Provides a 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) and beta=0.5

2.3.2 Justification for beta=0.5:

The problem statement indicates a way to think in the opposite direction: Elderly people have once been young, but younger people have not yet been old!

- Elderly persons can relate to experiences and issues that are typical for younger people.

- But elderly persons may have more unique experience and issues, over younger persons.

Therefore it is more OK for the publisher to identify a Pensioner as a younger person, than the other way around.

As the dataset is imbalanced, there will be only a few chances to correctly predict the Target, rare as it is. But here will be a lot of chances to make false predictions of the Target!

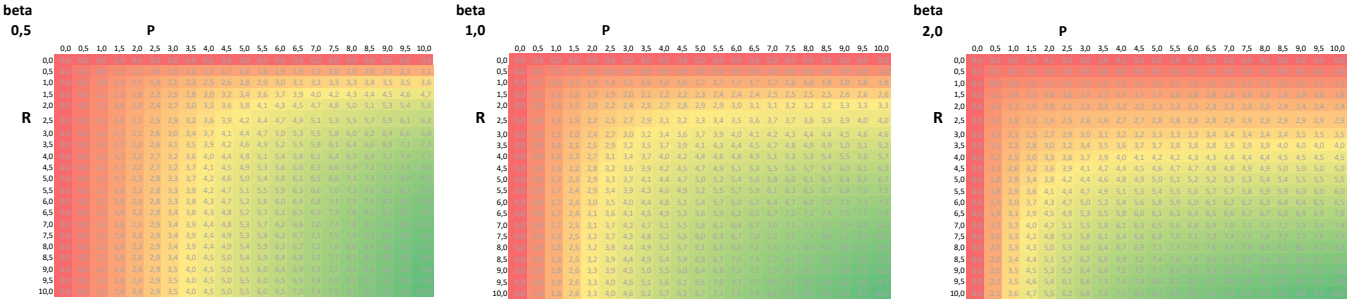
Technically speaking, False Positive are more un-popular than False Negative:

- The publisher wish to avoid False Positives, more than avoiding False Negatives.
- There is a good chance to ruin the precision with False Positives

Looking at the formulas for Precision and Recall, Precision is reduced by False Positives, and Recall is reduced by False Negatives. Hence it is more important to keep a good Precision, as False Positives are more un-wanted.

Beta=0.5 (called "f0.5") provides an accepted mix with emphasis on precision:

- f0.5 will be more sensitive to changes in precision than in recall
- f1.0 is evenly sensitive to changes in precision and recall
- f0.0 is entirely insensitive to changes in recall
- f3.0 is insensitive to changes in precision



2.3.3 Other indicators for prediction quality

- 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

Data sample showing count of visits pr section, including original name of the features:

	Feature1 VG+	Feature2 NetVision	Feature3 Sports	Feature4 News	Feature5 Meetings	Feature6 Parents	Feature7 Page1	Feature8 Page2	Feature9 Blogs	Feature10 Mens	Feature11 Woman	Pensioner
0	3	22	4	5	0	0	1	4	0	0	0	0
1	13	306	14	50	0	1	2	9	1	1	0	1
2	67	228	4	56	0	2	14	31	1	1	0	0
3	3	6	0	9	0	0	1	2	3	0	1	1
4	65	545	76	134	0	3	5	51	1	1	0	1
5	0	0	0	0	0	1	12	1	1	0	0	0
6	5	172	45	16	0	0	1	6	1	1	0	0
7	39	47	2	59	0	1	1	12	2	0	1	0
8	0	59	5	8	0	1	5	1	1	1	0	0
9	26	429	28	143	0	0	1	26	2	0	0	0
10	11	85	53	16	0	3	4	9	1	1	0	1
11	4	46	3	20	1	1	2	2	1	1	0	0
12	13	190	12	21	0	1	1	9	1	0	0	0
13	12	57	30	13	1	0	4	6	0	1	0	1
14	7	31	0	21	0	0	2	7	1	0	1	0

3.1.2 Statistics on the raw dataset

	count	mean	std	min	25 %	50 %	75 %	max
Feature1	4214	13,185335	30,772653	0	1	6	16	1417
Feature2	4214	134,631941	292,294704	0	5	47	146	5659
Feature3	4214	21,721168	69,585073	0	0	2	13	1252
Feature4	4214	25,763408	73,092197	0	2	11	30	3734
Feature5	4214	0,85121	2,060888	0	0	0	1	31
Feature6	4214	1,195064	2,428452	0	0	0	1	50
Feature7	4214	6,802563	28,376741	0	1	2	4	726
Feature8	4214	11,540104	21,723467	0	1	5	14	681
Feature9	4214	1,415282	6,446723	0	0	1	1	335
Feature10	4214	0,688182	0,463291	0	0	1	1	1
Feature11	4214	0,230422	0,421153	0	0	0	0	1
Pensioner	**4214**	**0,136213**	**0,343055**	**0**	**0**	**0**	**0**	1

By grouping values (count of visits pr Feature) in most of the features, one can see the main categories used by Pensioners as a group, vs the whole population. As one will see: The most of the single rows are here grouped into the first grouping.

Top 50 countings:

Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7	Feature 8	Feature 9	Feature 10	Feature 11	Count of Pensioner	Count of All population
0-99	0-99	0-99	0-99	0-9	0-9	0-99	0-99	0-99	0-9	0-9	355	2584
								100-199	0-9	0-9	0	1
								300-399	0-9	0-9	0	1
							100-199	0-99	0-9	0-9	0	16
							200-299	0-99	0-9	0-9	0	1
						100-199	0-99	0-99	0-9	0-9	2	23
							100-199	0-99	0-9	0-9	1	2
						200-299	0-99	0-99	0-9	0-9	0	7
						300-399	0-99	0-99	0-9	0-9	0	1
						400-499	0-99	0-99	0-9	0-9	0	3
						700-799	0-99	0-99	0-9	0-9	0	1
					10-19	0-99	0-99	0-99	0-9	0-9	0	15
						100-199	0-99	0-99	0-9	0-9	0	1
						200-299	0-99	0-99	0-9	0-9	0	2
					40-50	0-99	0-99	0-99	0-9	0-9	0	1
				10-19	0-9	0-99	0-99	0-99	0-9	0-9	1	5
						100-199	0-99	0-99	0-9	0-9	0	4
						200-299	0-99	0-99	0-9	0-9	0	1
						300-399	0-99	0-99	0-9	0-9	0	1
					10-19	0-99	0-99	0-99	0-9	0-9	0	2
						100-199	0-99	0-99	0-9	0-9	0	1
				20-29	0-9	0-99	0-99	0-99	0-9	0-9	0	3
						200-299	0-99	0-99	0-9	0-9	0	1
						300-399	0-99	0-99	0-9	0-9	0	1
					10-19	200-299	0-99	0-99	0-9	0-9	0	1
						300-399	0-99	0-99	0-9	0-9	0	1
				30-39	0-9	200-299	0-99	0-99	0-9	0-9	0	1
			100-199	0-9	0-9	0-99	0-99	0-99	0-9	0-9	7	12
					10-19	0-99	0-99	0-99	0-9	0-9	1	2
				10-19	0-9	0-99	0-99	0-99	0-9	0-9	0	1
		100-199	0-99	0-9	0-9	0-99	0-99	0-99	0-9	0-9	5	41
			100-199	0-9	0-9	0-99	0-99	0-99	0-9	0-9	0	1
		200-299	0-99	0-9	0-9	0-99	0-99	0-99	0-9	0-9	3	19
		300-399	0-99	0-9	0-9	0-99	0-99	0-99	0-9	0-9	2	14

Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7	Feature 8	Feature 9	Feature 10	Feature 11	Count of Pensioner	Count of All population
		400-499	0-99	0-9	0-9	0-99	0-99	0-99	0-9	0-9	0	4
		500-599	0-99	0-9	0-9	0-99	0-99	0-99	0-9	0-9	0	7
		600-699	0-99	0-9	0-9	0-99	0-99	0-99	0-9	0-9	0	4
		700-799	0-99	0-9	0-9	0-99	0-99	0-99	0-9	0-9	0	2
		800-899	0-99	0-9	0-9	0-99	0-99	0-99	0-9	0-9	0	1
		1100-1199	100-199	0-9	0-9	0-99	0-99	0-99	0-9	0-9	0	1
		1200-1299	0-99	0-9	0-9	0-99	0-99	0-99	0-9	0-9	0	1
	100-199	0-99	0-99	0-9	0-9	0-99	0-99	0-99	0-9	0-9	84	557
							100-199	0-99	0-9	0-9	1	4
						100-199	0-99	0-99	0-9	0-9	0	1
					10-19	0-99	0-99	0-99	0-9	0-9	0	1
				10-19	0-9	0-99	0-99	0-99	0-9	0-9	0	2
			100-199	0-9	0-9	0-99	0-99	0-99	0-9	0-9	4	17
					10-19	0-99	0-99	0-99	0-9	0-9	0	1
			200-299	0-9	0-9	0-99	0-99	0-99	0-9	0-9	0	1
		100-199	0-99	0-9	0-9	0-99	0-99	0-99	0-9	0-9	2	23

3.1.3 Skew in the dataset

How well the Logarithmic Transformation fixed skew on the Features (Target were left untransformed, as it is a class):

	Skew before	Skew after:
Feature1	25,200943	0,71661
Feature2	7,663886	1,003036
Feature3	7,883236	1,243534
Feature4	33,290134	0,598671
Feature5	6,485995	1,627191
Feature6	5,555848	1,175694
Feature7	11,636951	0,841028
Feature8	10,235551	0,825435
Feature9	36,913662	-0,041701
Feature10	-0,812758	-1,804033
Feature11	1,280796	2,244432
Pensioner	2,121877	2,121877

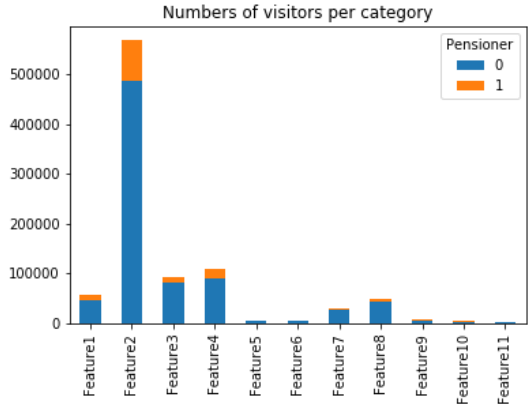
3.2 Exploratory Visualization

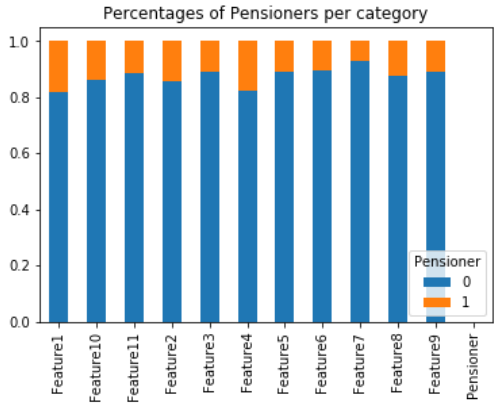
3.2.1 Imbalanced dataset: Target baseline

Illustrate the level of imbalance in the dataset:

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

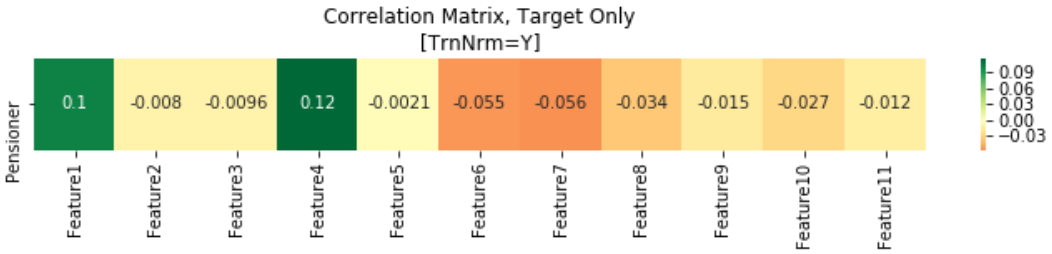
Target baseline for entire dataset is ~13%:





3.2.2 Correlation Matrix

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

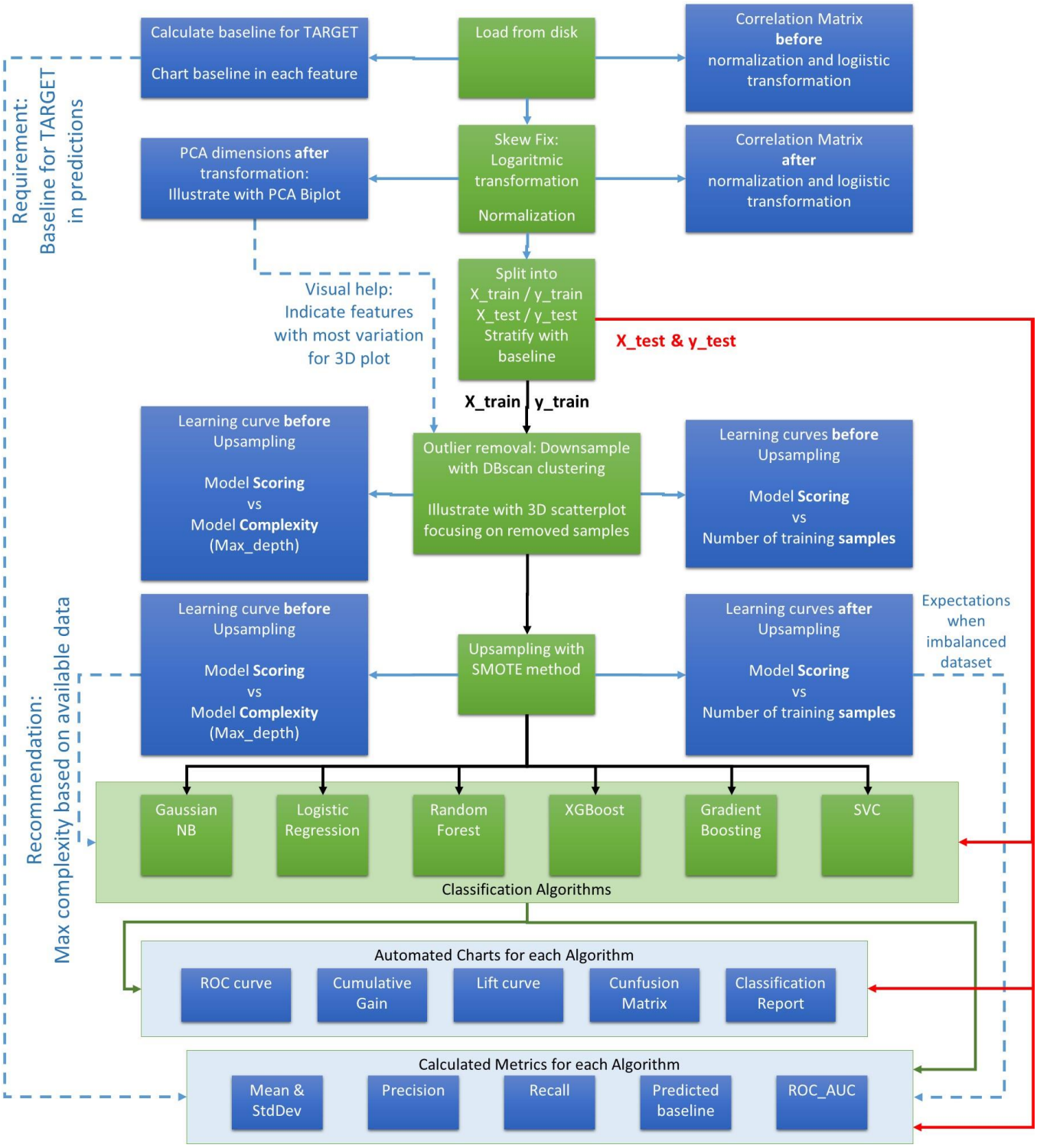
Learning curves
Sklearn methods used to display learning curves and model coplexity
- learning_curve. Used during ModelLearning
- validation_curve. Used during ModelComplexity

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.

This also means that a precision better than 13% must be considered positive, as a lift.

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.

Before fitting the selected Classifiers, there will still be performed some preparations:

This project will use the metrics after fitting to compare the effects from each of the methods designed for preparing the data:

- Transformation
- Downsampling
- Upsampling

Therefore multiple runs with different enabling of methods (arranged in different notebooks with the same random-state) will be performed and metrics being collected.

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:

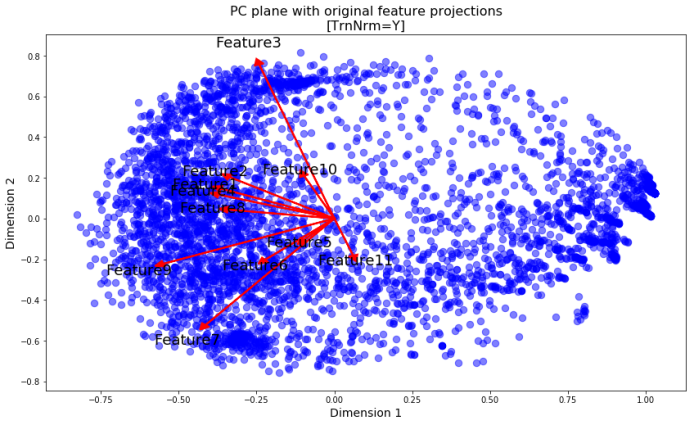
1. DBscan suggests samples not clustered
2. Samples that are not clustered may be of the rare Target. In these cases the sample must be preserved: The sample gets labeled with the first cluster made by DBscan.

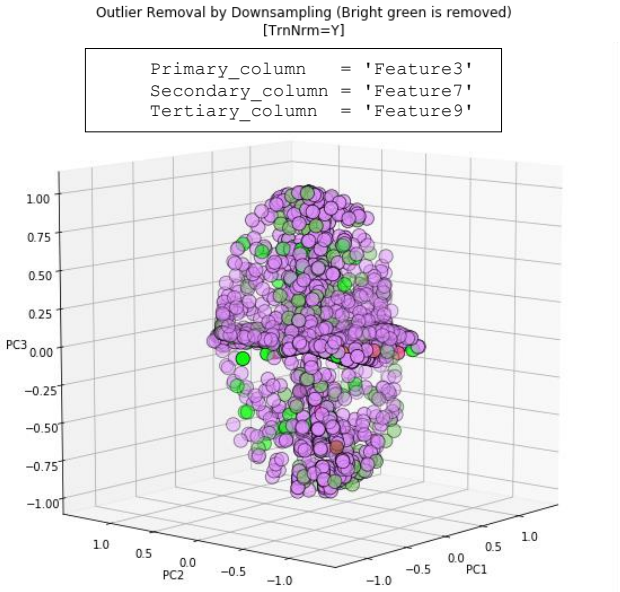
Another Downsampling technique described widely is to remove samples at random, as a contrast to clustering which is more controlled. This will not be performed in my project.

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:

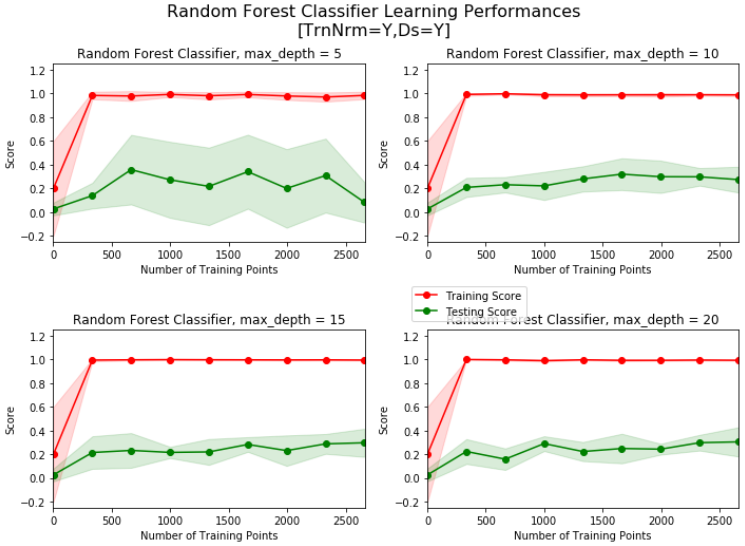




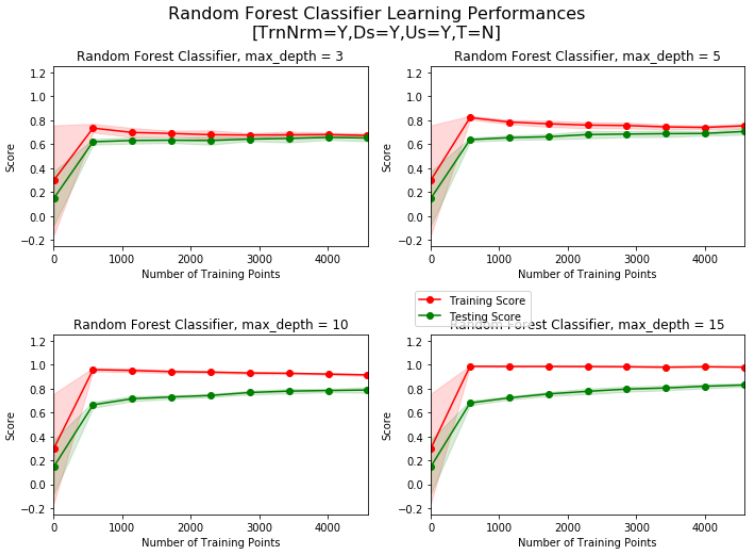
Learning curves

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.

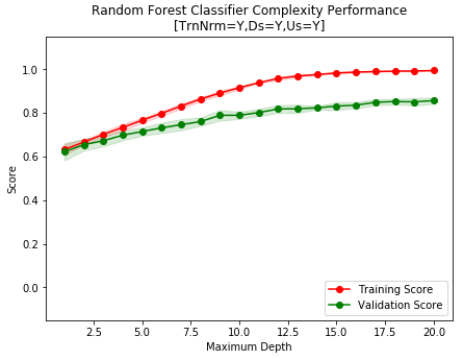


    2. After upsampling, a lot more of the rare Target was introduced, and I could get a sense of what to expect:



    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:

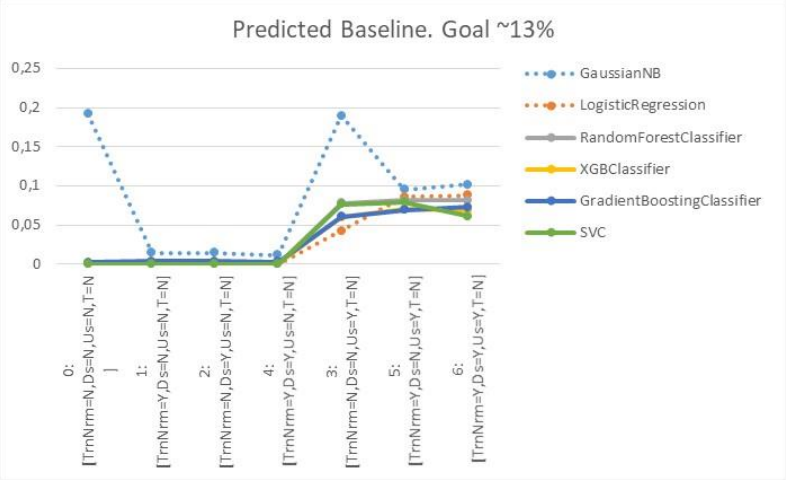
Run with different enabling of each of these techniques:

- 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

NB, for further reading: I discovered that the splitting routine did not have have the same random-state every time, this may influence a little bit even though I included a stratifying process.

Setups that comply with the benchmark baseline criterion:



Setups that provides ~ 0% in predicted baseline:

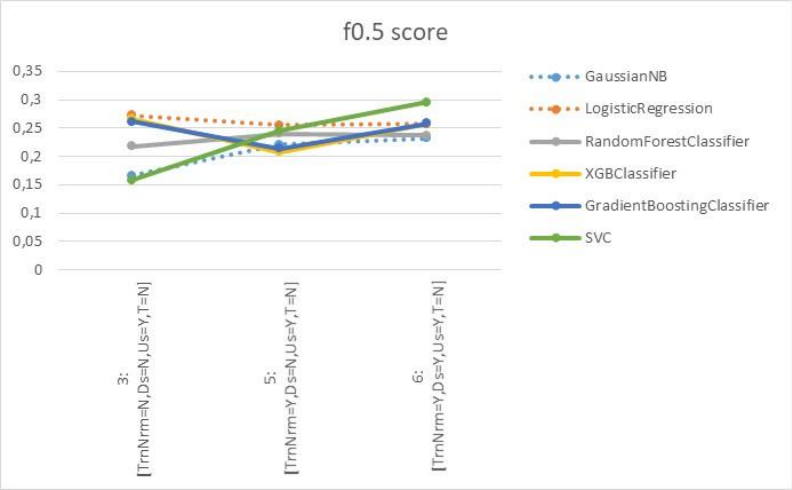
0: [TrnNrm=N,Ds=N,Us=N,T=N]
1: [TrnNrm=Y,Ds=N,Us=N,T=N]
2: [TrnNrm=N,Ds=Y,Us=N,T=N]
4: [TrnNrm=Y,Ds=Y,Us=N,T=N]

Required baseline is ~ 13%.

As one can see:
Upsampling is not member in these setups

These setups will be removed from further analysis

Setups with an acceptable f0.5 score, before hyperparameter tuning:



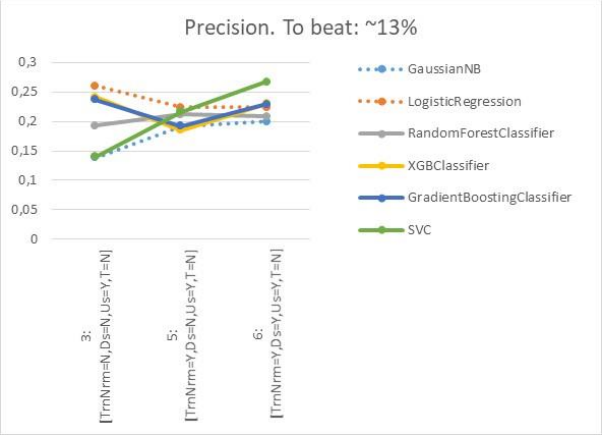
3: [TrnNrm=N,Ds=N,Us=Y,T=N]
5: [TrnNrm=Y,Ds=N,Us=Y,T=N]
6: [TrnNrm=Y,Ds=Y,Us=Y,T=N]

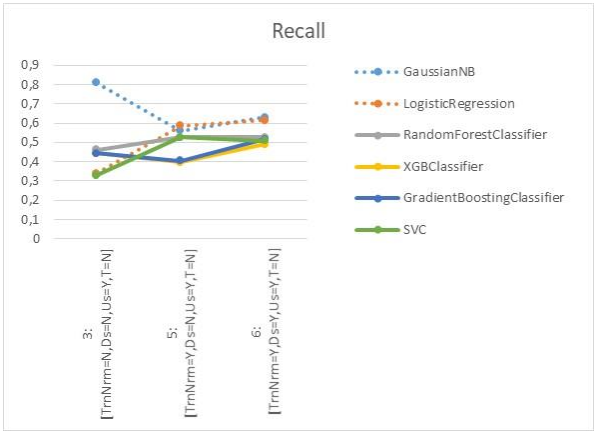
As one can see:
Upsampling is member
in these setups

Downsampling is member only of the most promising setup.

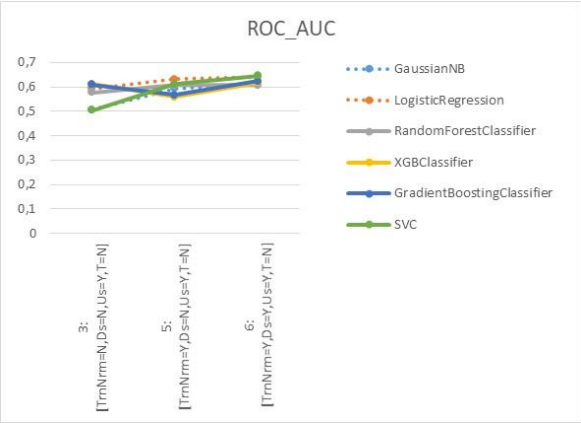
Some Classifiers perform better without TrnNrm: LogisticRegression and maybe GradientBoosting,

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.

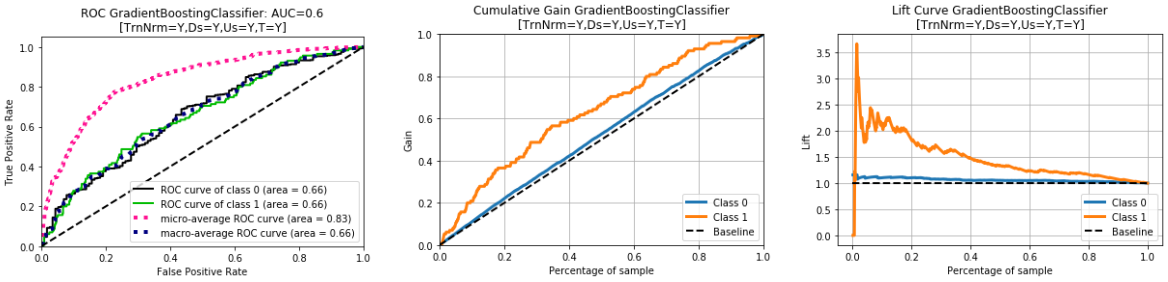
Achieved scores after GridSearch:

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

5.1.1 Robustness in order of randomness

Multiple runs of model with all methods enabled, except hyperparameter tuning

Run number 1

[TrnNrm=Y,Ds=Y,Us=Y,T=N]

	GaussianNB	LogisticRegression	RandomForestClassifier	XGBClassifier	GradientBoostingClassifier	SVC
acc_mean	0.832774	0.862396	0.862403	0.861233	0.84344	0.862403
acc_stddev	0.0138296	0.00439292	0.00181698	0.00843744	0.0140447	0.00181698
f_beta	0.247875	0.265487	0.228979	0.247253	0.251832	0.239448
precision	0.216049	0.232258	0.200658	0.221311	0.22541	0.21147
pred_base	0.0911905	0.0872502	0.0855615	0.0686744	0.0686744	0.0785252
recall	0.603448	0.62069	0.525862	0.465517	0.474138	0.508621
roc_auc	0.627034	0.646658	0.595806	0.602085	0.607083	0.603004

Run number 2

[TrnNrm=Y,Ds=Y,Us=Y,T=N]

	GaussianNB	LogisticRegression	RandomForestClassifier	XGBClassifier	GradientBoostingClassifier	SVC
acc_mean	0.83513	0.863581	0.863581	0.839877	0.844632	0.863581
acc_stddev	0.0129681	0.00039686	0.00039686	0.00958613	0.0139	0.00039686
f_beta	0.249458	0.263963	0.221122	0.269331	0.255411	0.270851
precision	0.217666	0.231544	0.191429	0.239382	0.226923	0.240458
pred_base	0.0897001	0.0843237	0.0990379	0.0732881	0.073571	0.074137
recall	0.6	0.6	0.582609	0.53913	0.513043	0.547826
roc_auc	0.62967	0.64272	0.596936	0.634263	0.618472	0.637237

Run number 3

[TrnNrm=Y,Ds=Y,Us=Y,T=N]

	GaussianNB	LogisticRegression	RandomForestClassifier	XGBClassifier	GradientBoostingClassifier	SVC
acc_mean	0.823371	0.86241	0.864777	0.861241	0.85059	0.864777
acc_stddev	0.0335169	0.00543812	0.00174326	0.0118472	0.018091	0.00174326
f_beta	0.231768	0.256498	0.233577	0.254919	0.255009	0.227273
precision	0.199468	0.222552	0.203822	0.227092	0.227642	0.19571
pred_base	0.106546	0.0954945	0.088977	0.071125	0.0697081	0.105696
recall	0.657895	0.657895	0.561404	0.5	0.491228	0.640351
roc_auc	0.6225	0.649249	0.609234	0.616941	0.615299	0.614414

Run number 4

[TrnNrm=Y,Ds=Y,Us=Y,T=N]

	GaussianNB	LogisticRegression	RandomForestClassifier	XGBClassifier	GradientBoostingClassifier	SVC
acc_mean	0.828076	0.86241	0.863594	0.845828	0.835149	0.864777
acc_stddev	0.0216208	0.00543812	0.0032717	0.0184114	0.0193003	0.00174326
f_beta	0.228385	0.284757	0.22655	0.248593	0.251479	0.261062
precision	0.201439	0.251852	0.199301	0.222689	0.226667	0.232283
pred_base	0.0786867	0.0764223	0.080951	0.0673648	0.0636853	0.0718936
recall	0.491228	0.596491	0.5	0.464912	0.447368	0.517544
roc_auc	0.593351	0.6597	0.592936	0.60557	0.604343	0.625027

One can see that the metric f_beta (f0.5) fluctuates during the different runs.

5.1.2 Champion Classifiers after Hyperparameter tuning:

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

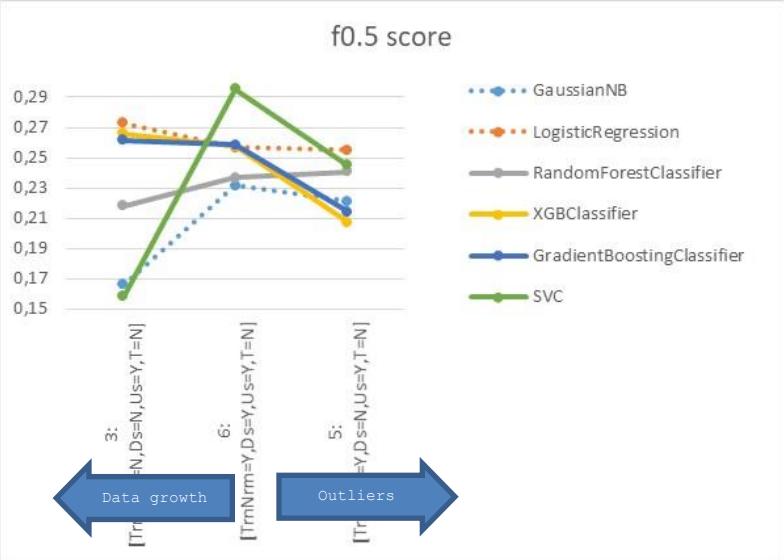
Confusion Matrix:

	false	true
negative	506	55
positive	221	61

5.1.3 Sensitivity to outliers and small changes in the data

In chapter **4.3.1 Compare the F_0.5 score between separate runs through the flowchart:** there is a section on 3 setups that provides acceptable f0.5 score. I will present the same information in a similar illustration, with a few changes:

- As configuration 6 is chosen as my champion configuration (before choosing any of the Classifiers), I put this in the middle of the sequence
- I'll also zoom in on the f0.5 scale
- Please be aware of the fluctuating nature in comparing different processes, as seen in section 5.1.1 Robustness in order of randomness.



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

Common features for these 3 sets:

- Downsampling(Ds)was performed
- No hyperparameter tuning (T)

Outliers:
Let lack of Downsampling (Ds) illustrate introduction of outliers.

Movement from configuration 6 to 5 can be compared to introducing outliers:

- ***SVC**, GaussianNB, **XGBClassifier** and **GradientBoostingClassifier** will perform slightly worse while introduced to outliers*
- RandomForestClassifier and LogisticRegression are almost indifferent

In this setting, outliers were entirely non-Targets, as unclustered Targets were forced into a cluster during Downsampling.

Changes in the data:
Let lack of Logarithmic Transformation (TrnNrm) illustrate a value growth in the data (All Features grow in the same order, like recording readers over a longer period that originally).

Movement from configuration 6 to 3 can be compared to data growth:

- ***SVC** and GaussianNB Classifiers will perform significantly worse while introduced to growth in data*
- RandomForestClassifier, LogisticRegression, **XGBClassifier** and **GradientBoostingClassifier** is almost indifferent

5.1.4 Does the model generalize well to unseen data?

As a proposal for demonstrating above statement that outliers will have significant effect:

1. Preserve the refused outliers. These data were only seen during preparations, and never shown to the champion Classifier.
2. Alternate Target class for a random selection corresponding to the original Target baseline.
3. Re-introduce and measure effects

5.1.5 Essentially, can we trust this model, and why or why not?

Before using this model, one must be aware of the general low f0.5 score, not even the precision is very high. The model in itself will save effort and bring a better reader experience for Pensioners.

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.

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

Even though results before GridSearch was promising, the results from the GridSerach was worse. This tells me that there are small margins, and I have not yet discovered local minima in my parameters or where to fine-tune my parameters.

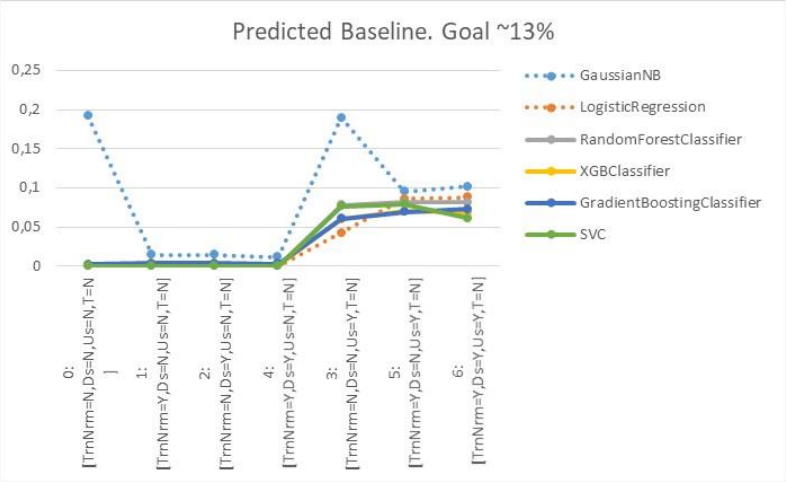
5.2.1 Are the improvements significant enough?

No, business-wise there may be some doubts:

The lift curves suggest that one can save effort in targeting readers with ads, but at the other hand there will be a more frequent need for retraining.

5.2.2 Is the model adequate for the problem?

Predicted Target baseline:
In chapter **4.3.1 Compare the F_0.5 score between separate runs through the flowchart:** there is a section on Setups that comply with the benchmark baseline criterion. I will here present the same illustration:



Compared to the original benchmark criterion in 3.4 Benchmark, **this model with none of its Classifiers are able to repeat the original Target baseline at 13%:**

GaussianNB	10 %
LogisticRegression	9 %
RandomForestClassifier	8 %
XGBClassifier	7 %
GradientBoostingClassifier	7 %
SVC	6 %

Precision:
The champion Classifiers have this precision:

XGBClassifier	18,5 %
GradientBoostingClassifier	19,2 %
SVC	21,6 %

Baseline benchmark is 13%. The achieved precision must be considered a large and positive lift.

Altogether, despite the predicted Target baseline requirement, I will therefore consider this model to be adequate.

5.2.3 Can the model solve the proposed problem?

The model itself contribute positive. But in practice, there is a need to retrain the models from time to time, as long as they are not 100% robust for changes in the data.

In other words: The training effort cannot be saved by a model's scores alone.

Imbalanced dataset leads to a very small set of Targets. Hence, changes in data will happen more often than with a well-balanced dataset, and therefore one have to record demographic data and retrain more often, than if the task was to analyze the gender of the reader. This may require more effort or introduce stress to customer relations.

I am not very sure if the publisher would invest in this simple solution alone, if not a prediction of Pensioners have extra high value.

6 Conclusion

I select GradientBoostingClassifier as my absolute 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 an imbalanced and small dataset, a good way of spending effort is to delve into learning curves.

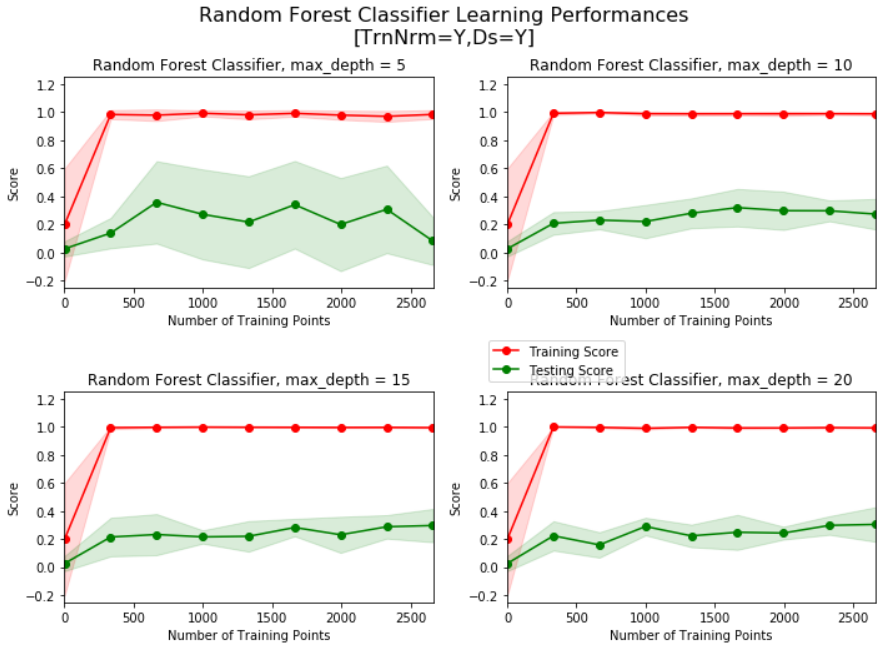
6.1 Free-Form Visualization

6.1.1 What I experienced: Learning curves together with Upsampling

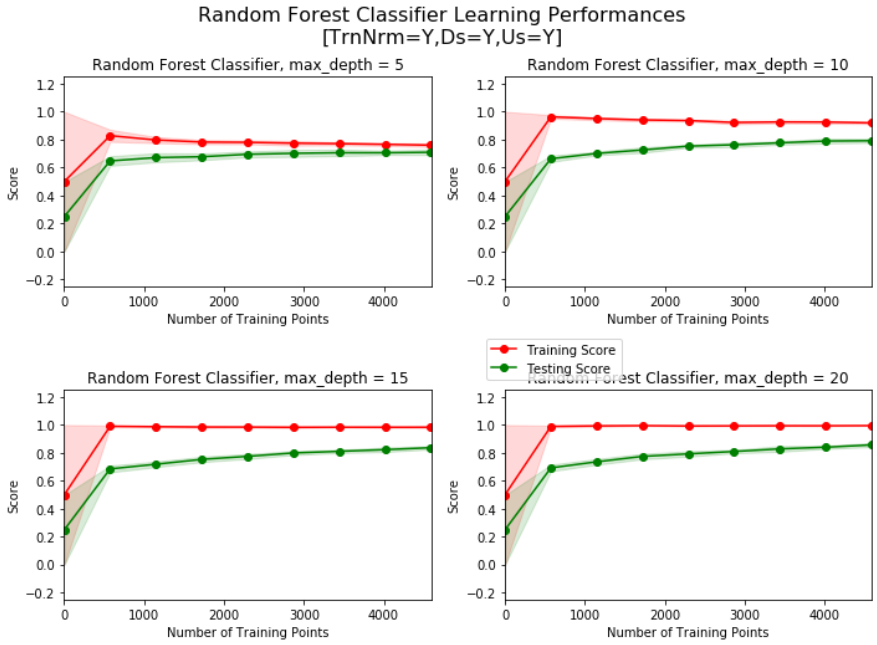
By using Learning Curves I found the optimal Max_Depth that is a parameter for several of my selected Classifiers. During this work I sometimes overshot these limitations and saw that performance dropped.

The learning curves before Upsampling had no focus. I then could not see:

- How important the model complexity is
- What level of precision I could expect from the Classifiers



After Upsampling (by SMOTE) the test set from 3328 rows to 5738 rows, the focus changed so that I could estimate max complexity and precision:



As SMOTE only upsamples on the rare Target class found in the Tran dataset, I could not expect the overall precision to be this good when testing on the Test dataset.

6.1.2 What I learned:

Upsampling is a good way to address imbalanced datasets

Using the learning curves on an imbalanced dataset

- The possibility for a high performance, achieved by high model complexity, may be assessed by studying the learning curves.
- If the scoring curves for train and test data converges before all available data is used, the correct level of model complexity is found.
- There is no use in aiming for a complexity that needs more data than available in order to achieve similar performance on both training and test sets.
- If model complexity is increased above recommendations, it is possible that GridSearch finds local minima while handling imbalanced datasets.

The final hyperparameter tuning during fitting of the selected Classifiers did not contribute so much as expected, further exploration is needed to reveal the ultimate parameter combination for each Classifier.

6.2 Reflection

6.2.1 Summary of end-to-end problem solution

A problem less discussed in the media, that is predicting Pensioners in order to customize reader experience and ad targeting, has been tried solved with several methods. Configurations that enabled upsampling with SMOTE had potential, all other combinations failed. Hyperparameter tuning at the end was not very successful.

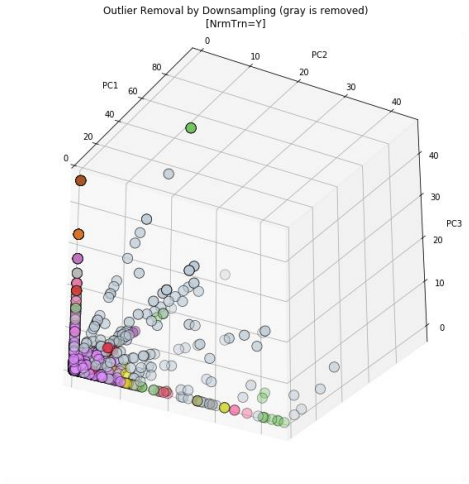
6.2.2 Challenges

A few experienced challenges, beyond the fact that I had to handle an imbalanced dataset:

Robust code for various Classifiers:
Make code robust for introducing various Classifiers. I did not make it for the learning curves, hence different code-cells was established for different Classifiers.

3D PCA Biplot:
A better understanding of the PCA Biplot could have introduced a 3D scatter and vector plot.

As I introduced 3D scatter plot for the DBscan clustering, the material became better explained. This became most clear while studying outliers in the un-transformed dataset. I could only show the 3 most important Features, but I knew that all 11 Features was used for the clustering. Therefore, samples that are far from the center in a 3D plot may be close connected via clusters not visible in only 3 dimensions. But still, in a 2D plot this distance would not be so visible.



6.3 Improvement

Proposed improvements on the models:

- Optimize number of outliers, by tuning the DBscan parameters
- Introduce ramdomness into the Downsampling, here I have removed unclustered samples (outliers)
- 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.
- When comparing setups, use the exact same split in Test and Train before comparing the later processes.

A good strategy if the publisher continuously repeats the customer surveys:

- Keep out a small set of the dataset - called the hold-out group.
- This group will act as a reference group, in order to see if a future recorded change in reader patterns is due to predictive publishing or a natural change due to external causes.

Especially if it is important to identify and influence the swing clients, one should keep a hold-out group for further analysis.

7 Appendix: Metrics

Metrics collected after different setup configurations.
Each setup had a single run.

	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