ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Machine Learning Applications in Credit Scoring

Evgenios Karezos – M.Sc. in Data Science thesis

Supervisor: Dr. V. Vassalos

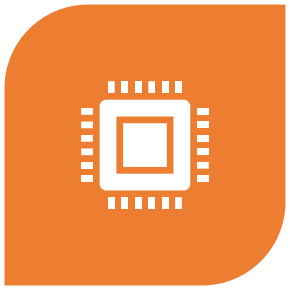
Who we are?

Channel VAS works with Mobile Telephone Providers

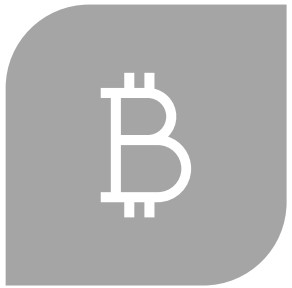
Provides their customers with extra airtime or bundles when they do not have money available on their account

Assigns every customer with a *credit limit* calculated after estimating their *Probability to Default (PD)*

Objective



USE *MACHINE LEARNING ALGORITHMS* TO PREDICT THE CUSTOMERS' *PROBABILITY TO DEFAULT*.



TRY TO OUTPERFORM THE ALGORITHM BEING USED UP UNTIL NOW.

NOISE AND DEFAULT RATIO FREE METHODOLOGY, ABLE TO SCORE CUSTOMERS IN SUBSEQUENT PERIOD.

Keynotes

- The data were highly imbalanced (e.g. 5% defaulters)
- Different Sampling techniques were tested
- Multiple Baseline & Ensemble Machine Learning Algorithms examined
- Tuning performed based on different metrics (AUC & Average Precision Score)
- AUC was selected for this industry, since PD gets used for Credit Limit Assignment

The Dataset

Customers split into four business-wise formerly formulated segments

The 167 features describe the customers' behaviour for a 15-month period and model their *Recharges*, *Advances* and *Recoveries*

Recharge: A customer credits their account with money

Advance: A customer borrows airtime or buys a bundle

Recovery: A customer repays their debt

The Dataset

- A customer is considered to have defaulted on their loan when they have not repaid after 6 months

Size & default ratio for the four segments:

	Patterns	Default Ratio
Segment 1	52.053	4,75%
Segment 2	113.263	4,32%
Segment 3	150.945	4,81%
Segment 4	101.904	5,46%

Data Preprocessing

Null values due to queries output replaced with 0

e.g. the average recharge value of a customer in a specific time period was *Null,* if there were no recharges during this period.

Variables describing dates were transformed to the equivalent tenures and the initial variables were dropped.

Modelling

- Scaled each feature separately before fed on algorithms like LASSO, SVM & ANN

- Specific algorithms might behave badly if the individual features do not look like standard normally distributed data

- Risk of a feature's variance to dominate the objective function

Sampling

Segment sizes and percent of the initial instances finally used

	Under-Sampled Patterns	Percentage
Segment 1	9.892	19,00%
Segment 2	19.592	17,30%
Segment 3	29.072	19,26%
Segment 4	22.244	21,82%

- Highly imbalanced data
- Used Under-Sampling to change the percentage of the bad cases and let the models learn more effectively

Probability Calibration

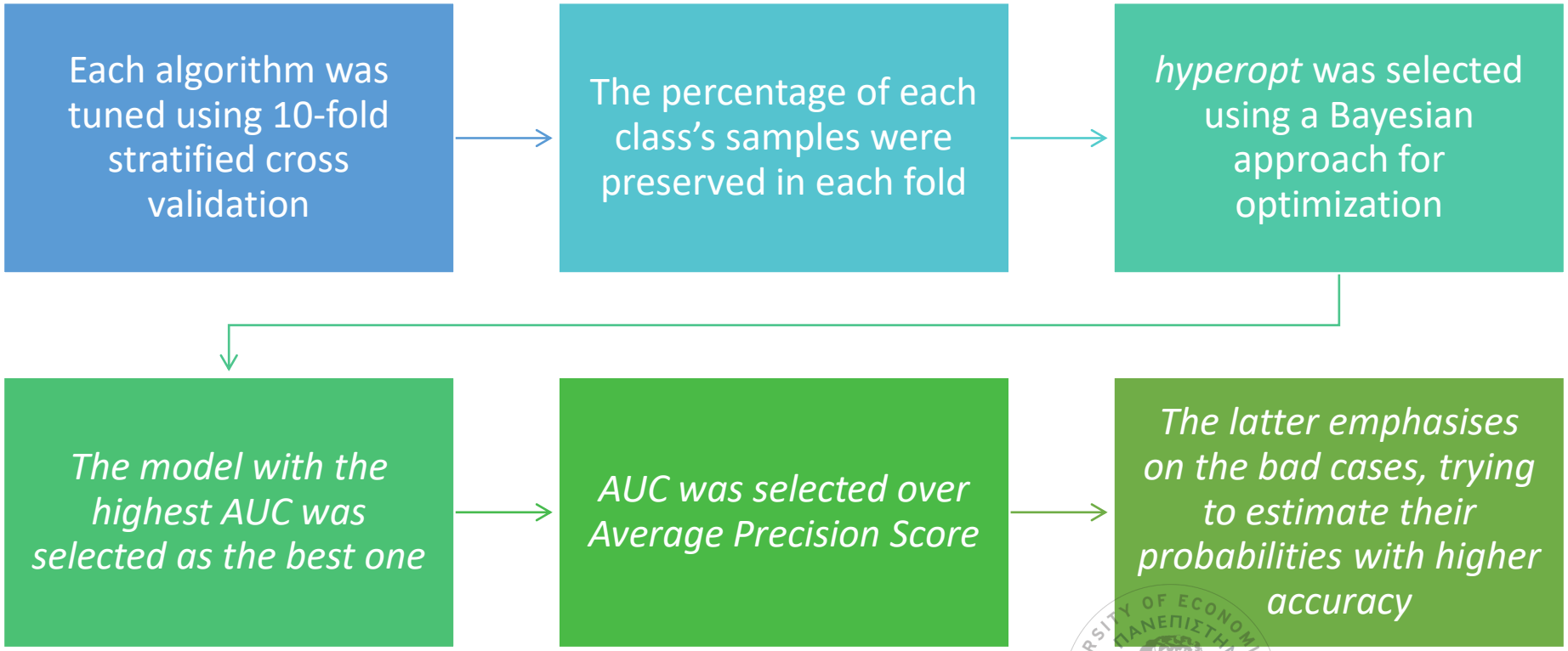
$$p = \frac{beta * p_s}{(beta - 1) * p_s + 1}$$

$$\text{beta} = \frac{\#\ \text{majority after under\_sampling}}{\#\ \text{original majority}}$$

p_s : predicted probability by the classifier

- Target class manipulation changes the distribution of prediction scores
- Sampling of non-target class leads to increased probability predictions
- If the probability score order is needed, this is not a problem
- The actual probability is used for the credit limit assignment
- The probabilities were calibrated

Tuning


Each algorithm was tuned using 10-fold stratified cross validation
The percentage of each class's samples were preserved in each fold
hyperopt was selected using a Bayesian approach for optimization
The model with the highest AUC was selected as the best one
AUC was selected over Average Precision Score
The latter emphasises on the bad cases, trying to estimate their probabilities with higher accuracy

Current Methodology

- The *Information Value* of each variable was calculated
- *Those with the highest values were kept*
- *Selected variables were fed into a genetically optimized penalized logistic regression algorithm, for the best variables to be selected*

Disadvantages

1. Weak learner not taking into account nonlinear terms and interactions
2. Time consuming procedure performed for every segment

Logistic Regression

- Tried to compare simple full logistic regression model to current method

AUC on the validation set

Segment	Logistic	Current	% Diff
1	0.62089	0.62995	-1.44%
2	0.63934	0.63135	1.27%
3	0.64221	0.63845	0.59%
4	0.65405	0.65555	-0.23%

LASSO & Ridge

- LASSO performs subset selection by setting coefficients of not important variables equal to zero
- Ridge Regression addresses multicollinearity – The features are highly correlated

AUC on the validation set

Segments	LASSO	Ridge	Current	% Diff
1	0,62056	0,61912	0,62995	-1,49% & -1,72%
2	0,63949	0,63783	0,63135	1,29% & 1,03%
3	0,64484	0,64341	0,63845	1,00% & 0,78%
4	0,65560	0,65507	0,65555	0,01% & -0,07%

Naïve Bayes & SVM

Naïve Bayes

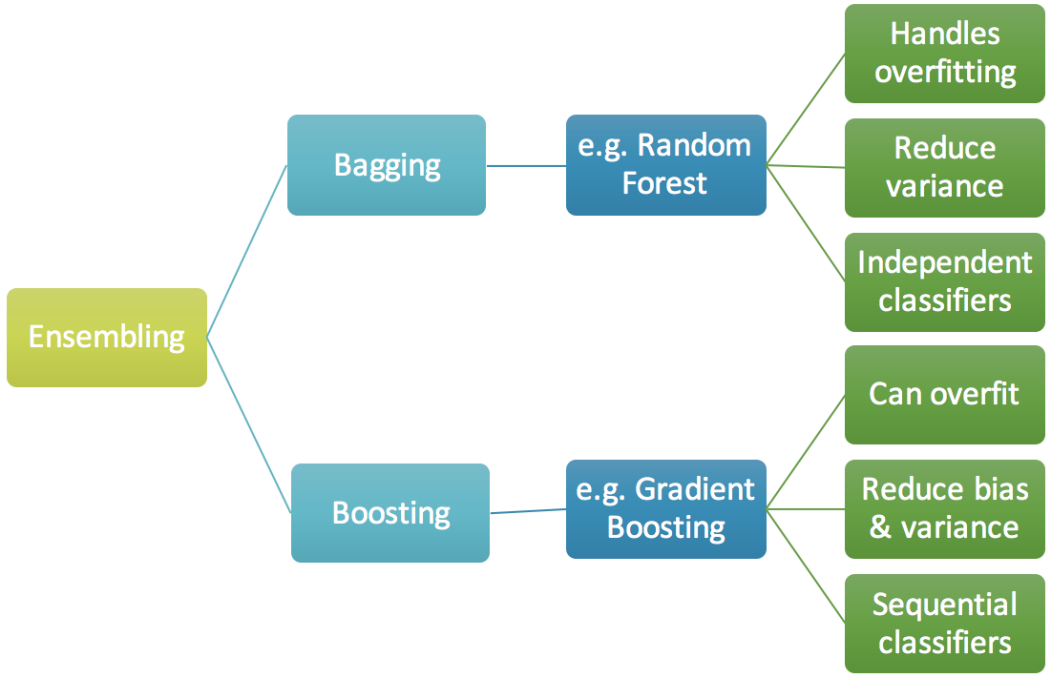
- Assume features to be statistically independent
- Possibly surpass multicollinearity by looking at every variable separately

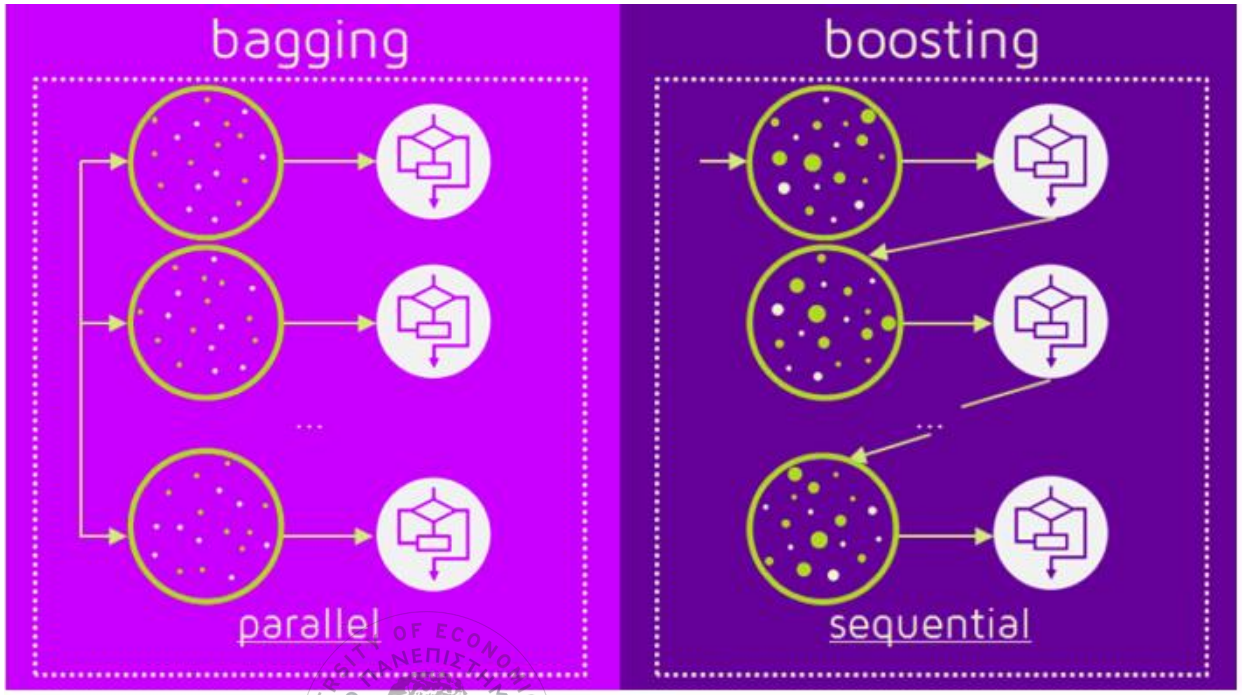
SVM

- Map inputs into high-dimensional feature spaces using non-linear transformation
- The classes become linearly separable, but the generalization error increases

Both algorithms performed even worse than the current one

Ensemble Methods





Bagging

- Applied at its most usual case, using a Decision Tree as *base estimator*
- Slightly better predictive ability than LASSO

AUC on the validation set

Segment	Bagging	Current	% Diff
1	0,62125	0,62995	-1,38%
2	0,63772	0,63135	1,01%
3	0,64544	0,63845	1,09%
4	0,65883	0,65555	0,50%

AdaBoost

- Applied at its most usual case, using a Decision Tree as *base estimator*
- Each model focuses on where the previous performed poorly
- First Decision Tree assigns equal weights on every instance
- Next ones increase the weights of difficultly classified observations and lower those of easier ones

AUC on the validation set

Segment	AdaBoost	Current	% Diff
1	0,62398	0,62955	-0,95%
2	0,64155	0,63135	1,62%
3	0,65022	0,63845	1,84%
4	0,65954	0,65555	0,61%

Gradient Boosting Machine (GBM)

- Instead of using weights like AdaBoost, it uses gradients in the loss function and takes steps into the direction of the negative gradient
- Approximates gradient with a weak learner to avoid over-fitting on the gradient
- It allows to optimize a user-defined cost function, suitable for the problem at hand

AUC on the validation set

Segment	GBM	Current	% Diff
1	0,62864	0,62995	-0,21%
2	0,64204	0,63135	1,69%
3	0,65095	0,63845	1,96%
4	0,66203	0,65555	0,99%

XGBoost

- Advanced Implementation of GBM
- Uses the second gradient to build the trees
- Makes fewer, but more accurate steps towards the minimum
- Keen on over-fitting
- Dropout, like in ANN, can be used to randomly drop boosting tree members (DART)

AUC on the validation set

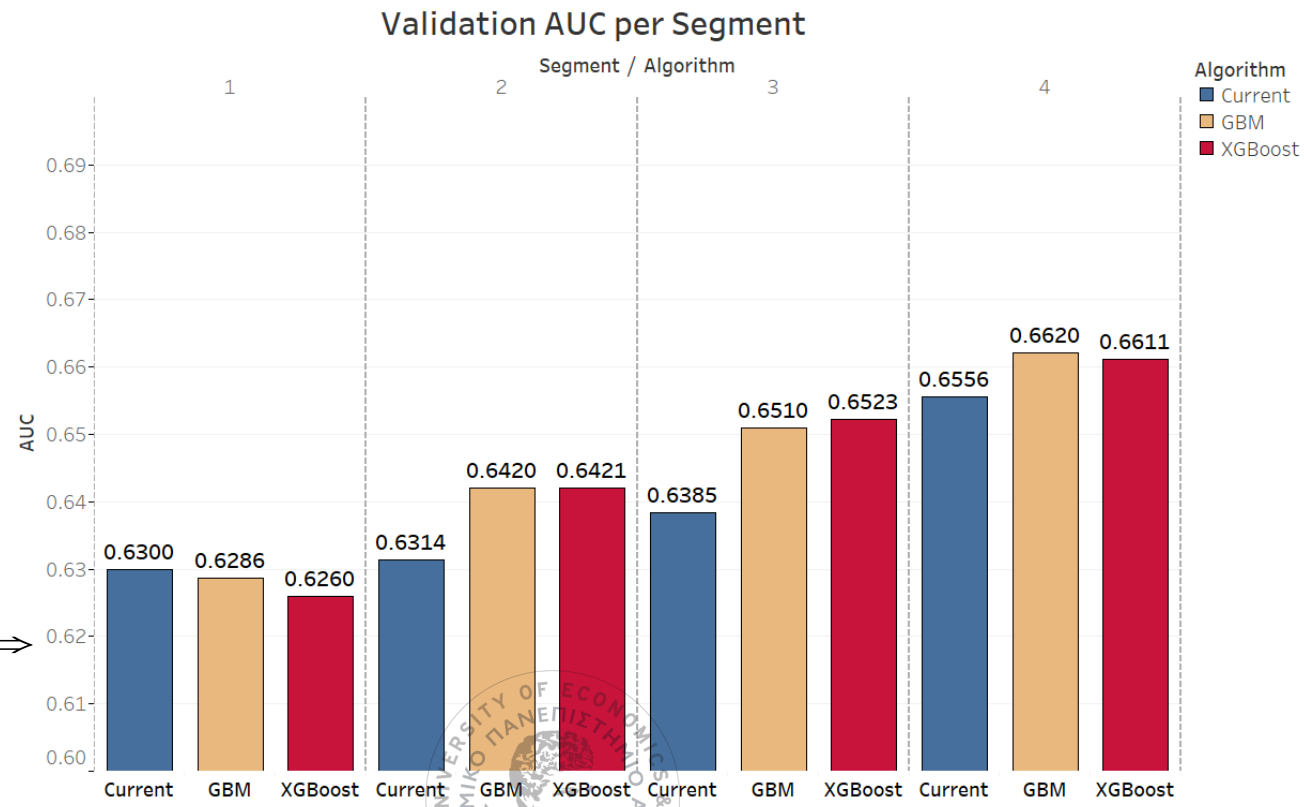
Segment	GBtree	DART	Current	% Diff
1	0,62595	0,62170	0,62995	-0,63% & -1,31
2	0,64208	0,64208	0,63135	1,70% & 1,70%
3	0,65227	0,65227	0,63845	2,16% & 2,16%
4	0,66111	0,66111	0,65555	0,85% & 0,85

Sum up

- XGBoost better on the third segment
- GBM better on the first and the fourth
- Fourth segment brings up to 50% of the revenues
- Current algorithm better on the first noisy segment

Algorithmically: XGBoost best model

Business-wise: GBM leads to higher revenues ⟹ better model



Overall model

- Tested if a catholic model, fitted across all segments, would be more predictive
- Only GBM and ANN were tested
  - GBM as the best model per segment
  - ANN because they learn better when fed with more observations

Deployment size and percent of the initial instances finally used

	Size
Initial Patterns	418.165
Under-sampled Patterns	61.820
Percentage	14,78%

Artificial Neural Networks

- ANN were tuned one time for each optimizer used

  - **SGD**: Performs frequent updates with high variance causing the objective function to fluctuate, potentially better local minima, complicates convergence to exact minimum. Used Nesterov momentum (add part of the previous step to the current update vector) to accelerate SGD and let it know where it is going, in order to slow down before another increment

  - **Adagrad**: Adapts *learning rate* to the parameters. Smaller updates for parameters associated with frequent features and inversely for infrequent features by accumulating all past squared gradients

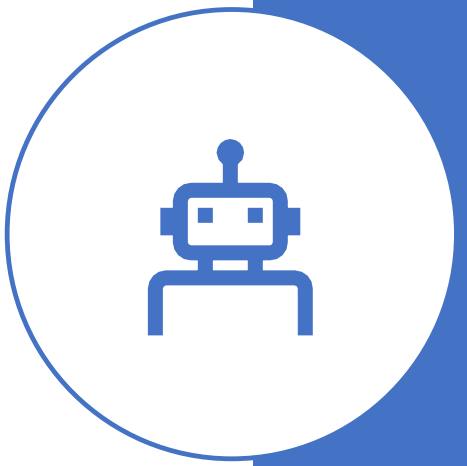
Artificial Neural Networks

- **Adadelta**: Adagrad's extension, reduces its monotonically decreasing learning rate. Restricts the window of accumulated past gradients to some fixed size. Learning rate absent from the update rule
- **RMSprop**: Identical to *Adadelta. Uses learning rate in the update rule*. *Specific parameter values are suggested*
- ***Adam****: Similar to previous two, but uses a momentum-like term*
  Momentum is like a ball running down a slope, Adam behaves like a ball with friction.

Exponential averages result to poor generalization ability of adaptive learning rate methods diminishing the influence of rare informative mini batches

Artificial Neural Networks

- **AMSGrad**: Uses the maximum of past squared gradients, results in a non-increasing step size and avoids problems suffered by Adam
- **Adamax**: Like Adam uses ℓ_2 norm, Adamax uses ℓ_∞ norm and does not lead to bias towards zero
- **Nadam**: Adam is a combination of RMSprop and momentum. Nadam combines Adam with Nesterov



Overall Models' Performance

- GBM dominates again
- The model using *AdaGrad* performed better, followed by *SGD*
- The rest optimisers did not face the deficiencies of those two
- AdaGrad will represent the ANN from here on

Algorithm	AUC	% Diff
Current	0,65695	-
GBM	0,67396	2,59%
SGD	0,66373	1,03%
AdaGrad	0,66548	1,30%
Adadelta	0,65410	-0,43%
RMSprop	0,66074	0,58%
Adam	0,66091	0,60%
Adamax	0,66191	0,76%
Nadam	0,66197	0,76%

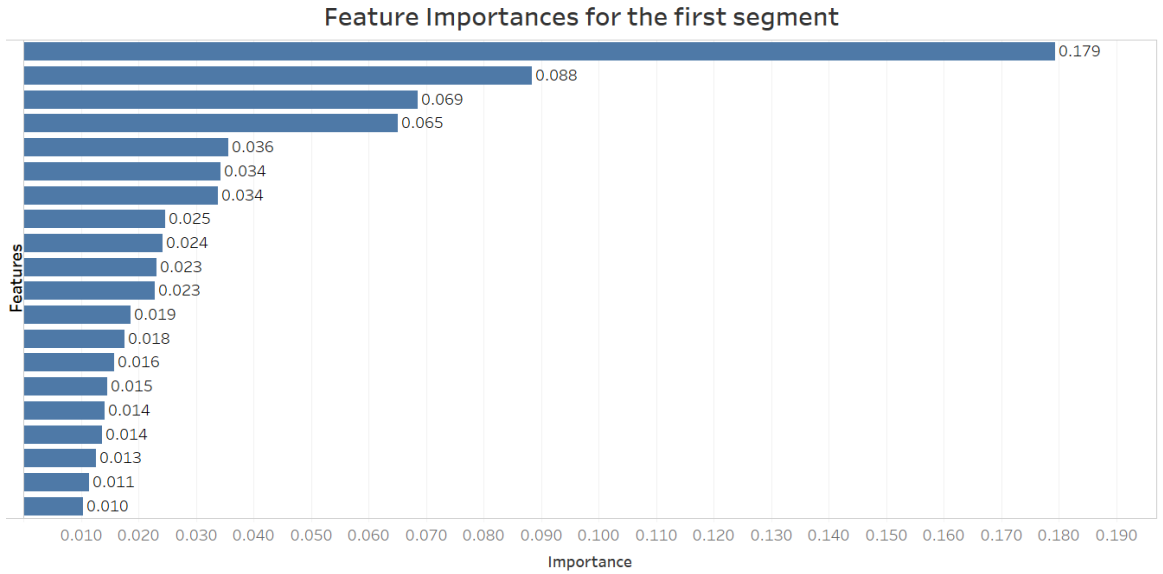
Performance on Unseen Data

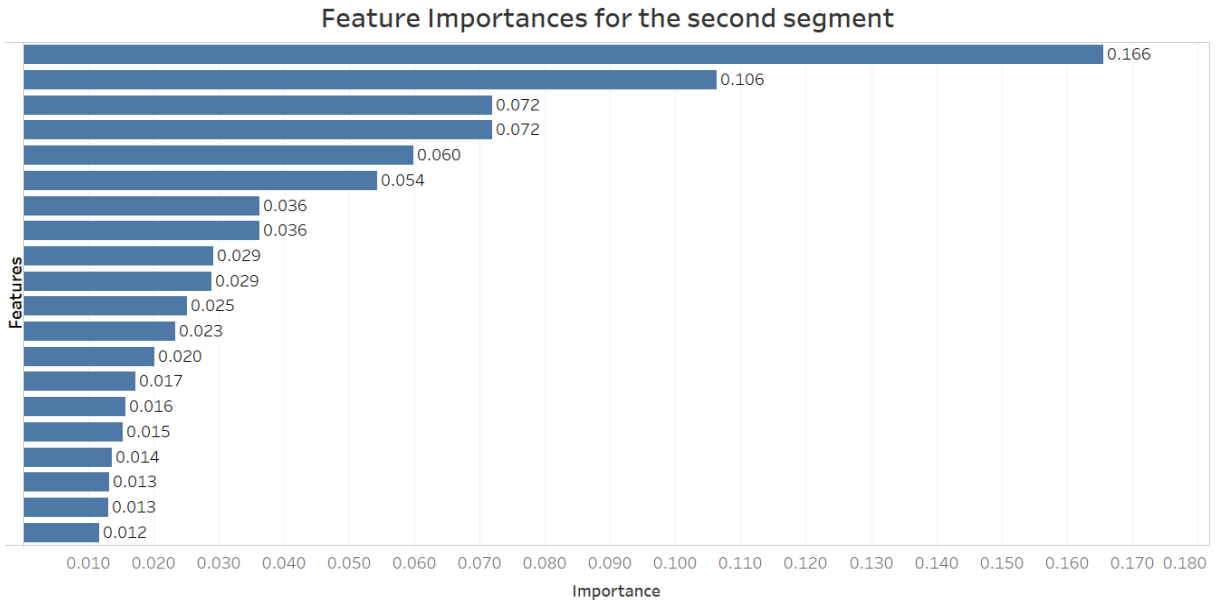
- Produced models need to be robust & perform well on data from a subsequent period of time
- Per segment models
  - GBM followed by XGBoost & Bagging outperform current methodology to the same extend as on the validation data
- Overall models
  - Consistent behaviour with GBM, AdaGrad & SGD having the best performance
  - Overall AUC is higher than its per segment values

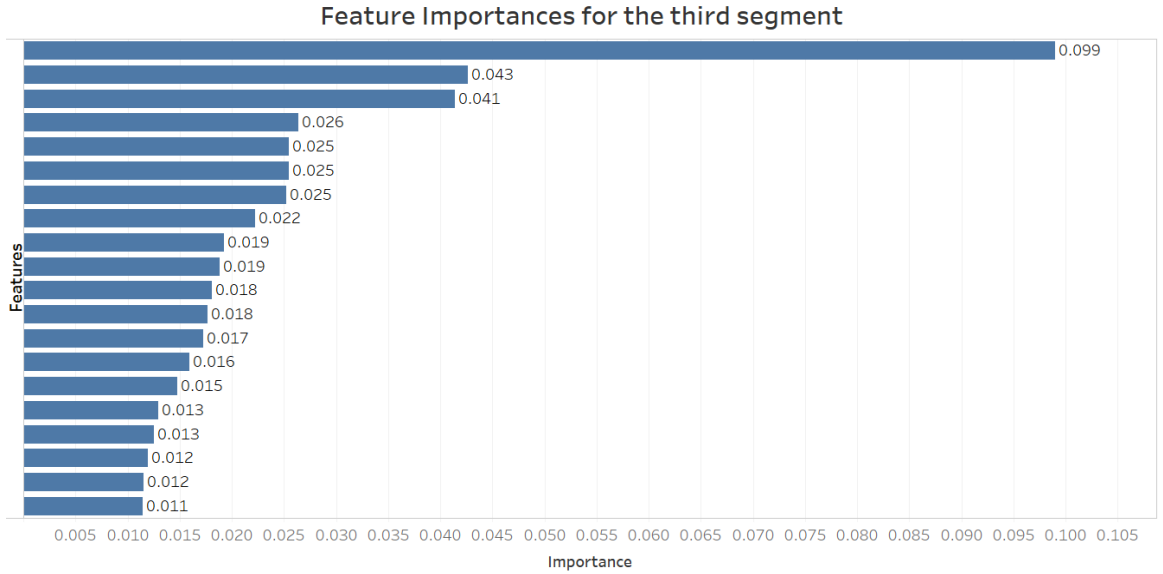
Overall GBM: 0,66493

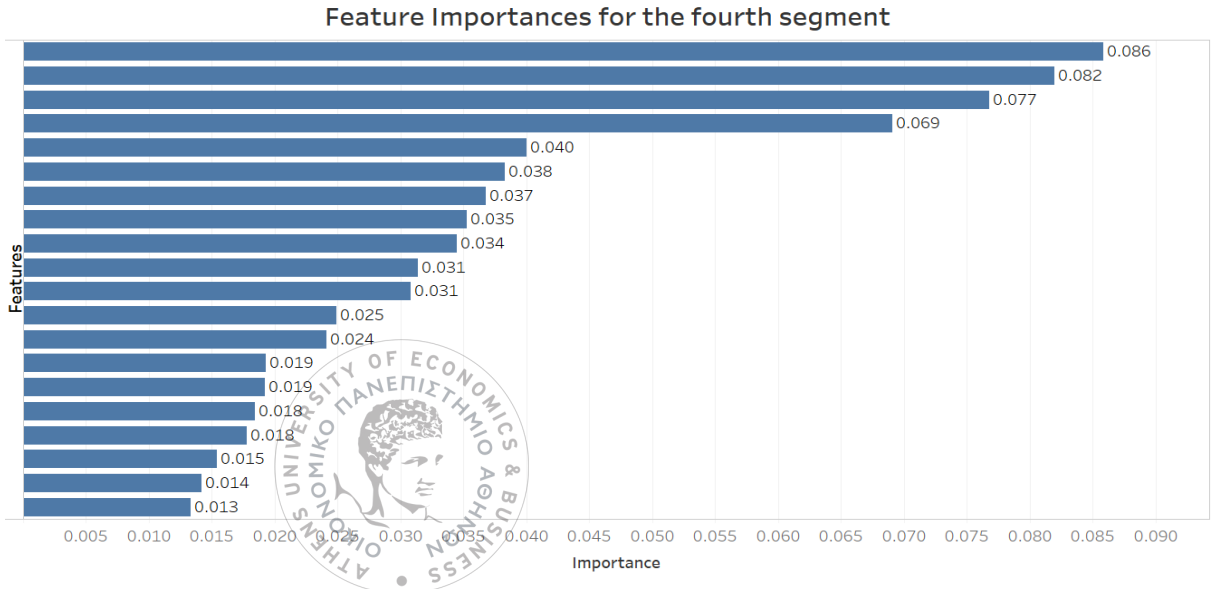
Highest per segment GBM: 0,65560

Feature Importances

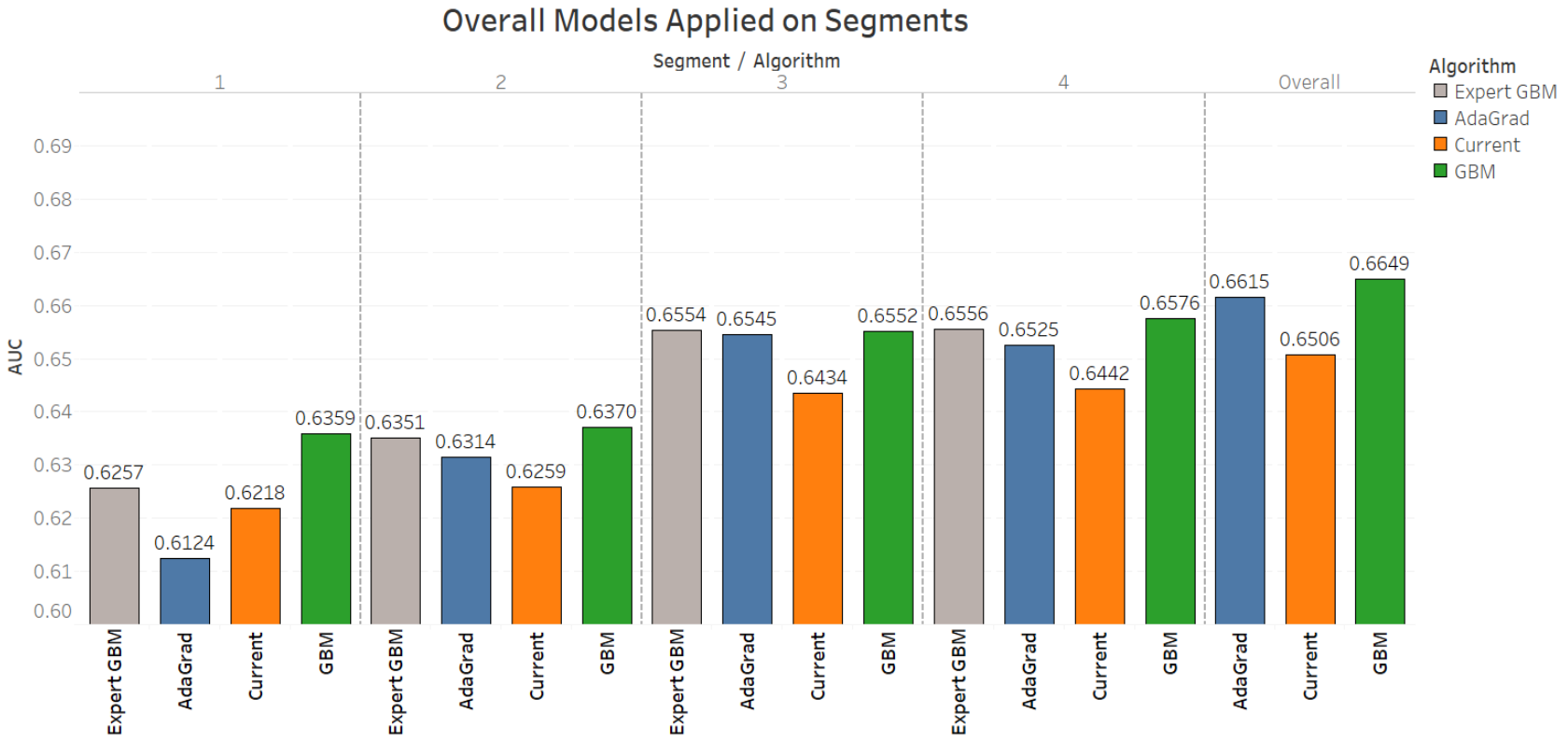








Overall Model Applied on Segments



- Each overall model was applied on each segment's out-of-time data
- They perform better than the segment experts
- Indicates to use only one model and score all segments' customers with this

Aggregate Experts

ALGORITHM	AGGREGATE	OVERALL	% DIFF
Current	0,6407	0,6506	-1,52%
GBM	0,6480	0,6649	-2,54%

- Calculate each out-of-time instance's PD, using its segment's expert model
- Compare their predictive ability to the corresponding catholic model on the same data
- An overall model performs better
- Another indication to use this instead of the segment experts

Conclusions

- Ensemble techniques & ANN provide the highest uplift in terms of AUC
- Each model was mainly based on 3-5 features
- The overall models discriminated more effectively
- All models were robust, since they performed equally well on totally unseen data, retrieved from a different period of time

Suggestions & Future Work

Suggestions

- Use existing GA to select most informative features & speed up performance
- New feature categories, learn different subscribers' aspects
- Substitute the per segment models on every deployment with an overall model

Future Work

- Parallelize the procedure rewriting the code in *PySpark*
- Test more extensive grids for the tuning of the hyper-parameters
- Implement *Stacking* & learn from the predictions of a bunch of classifiers

Thank you!!