Machine Learning
MsC in Data Science and Business Analytics
NOVA IMS

Newland Project Analysis

Ana Marta Silva m20200971
Gustavo Ferreira Tourinho m20180846
Natalia Castañeda m20200575
María Luisa Noguera Lecompte m20201005

Predicting tax categories with Gradient Boosting Decision Tree Classifier

ABSTRACT

Objective: Newland, is a city in a fictitious future in an exoplanet. To ensure that the terraforming is sustainable, its government must tax its citizens with an income below average with a 15% rate, and those above average with a 30% rate. However, they need to predict newcomers' rate to keep their finances in check, thus putting together a competition so different data science consultants come up with predictive models for how each new citizen must be taxed. **Methodology** Our group of consultants performed a rigorous evaluation of 15 different algorithms, using different python libraries, and comparing their accuracy within and amongst them using three different scenarios.
Results The performance of the models (in terms of F1 score) ranged between 84 and 87% without any indication of overfitting. All models predicted the 0 category far better since the data was unbalanced, however used ensemble models in order to reduce the bias and search for better performance. **Conclusions** The stacking model with Gradient Boosting Decision Tree and Logistic Regression as the final predictive model since its designed to outperform individual individuals and decrease bias and proved the most accurate with a score of 87% after cross validation.

Keywords: Classification, Machine learning, Naïve Bayes, Linear Discriminant Analysis, Extra Tree Classifier, Voting Classifier, Gradient Boosting Decision Tree, Bagging, Stacking, Histogram Gradient Boosting, F1 Score.

I. INTRODUCTION

Newland is the city of the future where earthlings are migrating to after a considerable degradation of their own planet.
Since the very beginning, the selection of Newland's citizens has been marked as controversial, and they could be grouped in three different categories: Essential workers selected by the government, volunteers, and those who paid to be colonizers.
Considering the massive migration that will take place, and for the sake of the sustainability of the city, the government has decided to tax their current and future residents with a 15% or 30% rate, depending on their level of income (above or below the average) and other demographic conditions.

To determine how the incoming citizens should be taxed, the government has opened a contest for data scientist to **present a model to predict the rate for each new citizen (1 for 30% and. 0 for 15%) based on demographic and financial information**. It has only provided these consultants with a sample of little over 32 thousand residents, split into two sets of data (22400 and 10100): a first with the tax rate labeled, and a second with no tax rate defined.

Once the data was received, the team of consultants determined that several models should be tested under three main conditions: the cleanest dataset possible, comparing their performance based on accuracy, and combining their professional experience (know-how) with hard theoretical concepts.

II. BACKGROUND

Several models discussed during practical classes were put to the test in search of the best performing model. However, our group decided to search for other alternatives and try other models like the Linear Discriminant Analysis, Voting Classifier, Histogram Gradient boosting and Extra Tree classifier. Also decided to do bagging of Decision Tree, Linear Regression and Naïve Bayes.[1]

A first new individual model introduced is the Linear Discriminant Analysis, a linear classification algorithm, frequently used in multi-class classification problems. It can also be used in supervised dimensionality reduction problems. This algorithm works by calculating summary statistics for the input features by class model which will be used to make the predictions. Consequently, It is calculated the conditional probability of these observations belonging to each class and selecting the class with the highest probability. This model uses the Bayes Theorem to estimate the probabilities.
It assumes that the data is gaussian and that each attribute has the same variance, so the values of each variable vary around the mean by the same amount on average. It also assumes that the input variables are uncorrelated, which is why it is important to standardize the data and remove outliers before running it so it won't skew the statistics used in the predictions. This model has other extensions like the quadratic (each class uses its own estimate of variance, this classier is equivalent to the gaussian naïve bayes classier), in fact, LDA is a special case of quadratic discriminant analysis. Besides, we also have the flexible (uses non-linear combination of variables) and the regularized (introduces regularization into the estimation of variance). [2]

A first new ensemble model included in the project was Gradient Boosting Decision Tree classifier, a machine learning algorithm for regression and classification trees, that produces prediction models for the ensemble. The Gradient Boosting Classifier (GBC) sets a sequential learning procedure for each new tree, and it improves the accuracy of learning and corrects from the previously built trees. [3] In Gradient Boosting Decision Tree Classifier, we tuned the hyperparameters (n_estimators, learning_rate, max_depth, and min_samples_split.) for improving accuracy and reduce overfitting.

The Extra Tree Classifier is an ensemble model based on randomized decision trees, which average predictions in order to create a combined estimator with improved accuracy and less overfitting. In this model the features will be selected randomly as well as the threshold for each feature and from these random thresholds the best will be picked as a splitting rule. For this model some of the possible parameters to change are number of estimators (n_estimators), which consist on the number of trees, number of features used in the random subsets to split the nodes (max_features) and max_depth which defines until when the tree will grow, if they will reach full development.[4]
After applying the grid search method, the parameters max_depth, max_features and number of estimators in our own mode were tuned.

Another ensemble model the group decided to test was the Voting Classifier. This ensemble model is used to combine different machine learning models and vote (hard or soft voting) on which is the predicted label for a record. When using hard voting the model decides on the label for a record based on the decision taken from the majority of models. When voting is *soft*, (or weighted) then a percentage is attributed to each model and then to each record. Using the selected classifiers, is calculated a probability of belonging to each class which is then multiplied by the weight percentage and averaged. The final class label corresponds to the class with the highest weighted probability. The voting classifier method is used with high performing models in order to compensate and balance their weaknesses.[5]

Lastly, a model was built using a Histogram Gradient Boosting, an experimental classifier that aims to be a faster Gradient Boosting classifier. The data set will be binned into integer valued bins that decrease the number of splitting points when building the trees, thus running. faster. This speed is particularly visible for datasets with more than 10000 occurrences. The parameters of this model are equal to the ones of gradient boosting with the exception of the number of estimators. Instead, there will be a parameter of maximum iterations which corresponds to the maximum number of trees, being the default 100.[6]

III. METHODOLOGY

Design

Our team of consultants took a heuristic approach for rapidly defining a model to predict taxation rate following the usual steps of data preprocessing and the application of 15 classification algorithms. The data was initially cleaned, engineered and re-engineered after initial feature filtering, visualizations, and method combinations.

Instruments

Initial general libraries were Imported for exploration and cleaning, others were imported for specific purposes such as feature selection, modeling, and visualization. (Table 1 shows the libraries used).

General use	**Data preprocessing and feature selection**	**Modeling (classifiers)**	**Visualization**
Pandas Scipy Numpy Math Datetime Calendar Os Io Intertools	Sklearn	Sklearn	Matplotlib Seaborn Pandas Profiling iPython Graphbiz

Table 1. Python libraries

Procedure

Our group of consultants first reviewed the detailed document that explained each variable, their format, and if there were any irrelevant features in the original file.

Both datasets (Train and Test) were added to the work environment to keep consistency in the preprocessing stage. All variables were checked for missing values, data type and appropriate formatting.

The feature engineering process was performed after an initial data exploration where the consultants found that the variables Ticket Price, Money Received, Name, Birthday, and names could be used to identify their Group (A, B or C to which each resident belong to regarding how they were selected), Gender, and Age.

The initial visualization of the categorical variables showed that the frequencies in some of the categories within the variables Base Area, Employment Sector, and Marital Status were considerably low, therefore the group decided to merge these categories for further evaluation. As a result, the categories within Base Area were reduced to "Northbury" and "Other", Employment Sector to "Private", "Public", "Self-Employed" and "Unemployed", and in Marital status only the subcategories for "Married" were all merged under this generic status.

With these variables in place, missing values were filled for Role and Employment Sector only, using their mode, to then encode all categorical variables using the One-Hot encoding method.

Outliers for all numerical variables were removed using a combination of the IQR method and manual selection after box plot visualization, ensuring that 95% of the data was kept. Finally, all numerical variables were normalized with the Min-Max normalization method.

Once cleaned, feature selection was performed considering relevancy and redundancy, using appropriate methods depending on their type of variable.

In terms of redundancy, we applied four different methods: ANOVA score, Mutual information score, and Pearson correlation for the numerical variables, and Chi-square score for the categorical variables. After this evaluation, Education Level variable was discarded given that it had a significantly high correlation with the Years of Education variable.

Considering the results of these scores, the group decided to build three different scenarios, where two of them were built using different combination of numerical and categorical variables, and another using all the variables selected and engineered.

Name	Variables
Scenario 1	All variables except Education Level
Scenario 2	Without Working Hours per week, Chi-Sq = 10
Scenario 3	Without Working Hours per week, Chi-Sq = 20

Table 2. Variable based Scenarios

Once the preprocessing stage was completed, the first classifiers were trained with a split of the Train dataset ("Train", "Test" and "Validation") and then evaluated their performance with cross validation, while tuning the respective hyperparameters.
The first models were trained and validated using all three of the scenarios. However, with the exception of Naïve Bayes, there was no significant difference in the results comparing the scenarios, regarding their performance. After four trials, with the Naïve Bayes, Logistic Regression, Linear Discriminant Analysis and KNN, the subsequent models were only trained, validated and tested with the first and second scenarios.

Of all the different scenarios created, the best preforming one was selected for hyperparameter tuning in search of better performance, which was actually achieved[7]. These tuned models were furtherly compared using the AUC/ROC (which helps to identify the discrimination capacity of a model to distinguish between class 1 and 0); the higher the AUC the better the measure of separability[8] and used to create ensembled models (bagging, voting classifiers and stacking), each with several individual model combinations to be compared again within themselves.

Because of the imbalanced nature of the dataset, the Train set was treated with under and over sampling methods to then retrain one of the initial models, and compare it with its initial performance. However, the model's performance decreased, hence the over and under sampled set was not used any further.

IV. RESULTS

The decision of which model was the most appropriate for tax rate prediction, is the result of a continuous selection of the best performing models of each stage of the process.
After training, validating and testing several models with the preprocessed data sets, model was tuned, depending on its hyperparameters. The top performing ones were compared with each other using the Receiver Operating Characteristics chart. The preliminary results of each model are shown below.

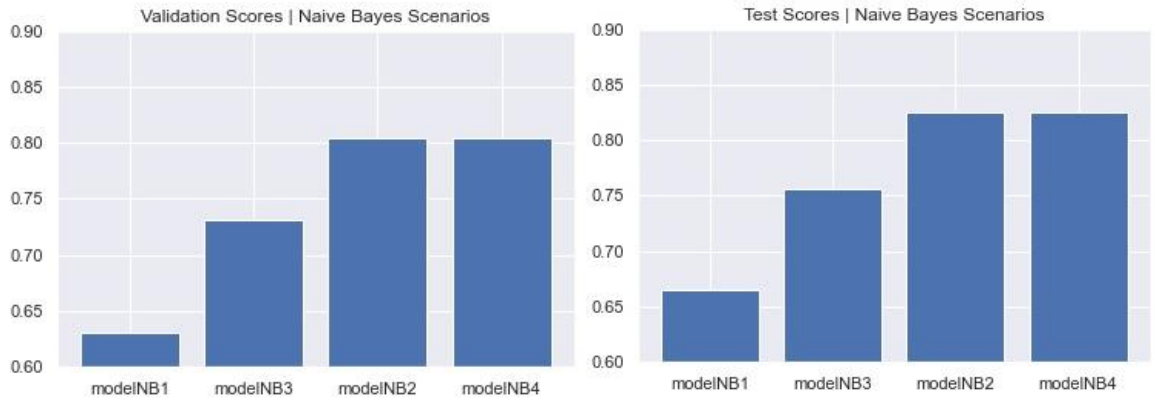

Chart.1 Naïve Bayes NB4

The best performing model using Naïve Bayes was ModelNB4, trained with the variables of Scenario 2 with the following parameters: Bar smoothing 5.33e-5, for a resulting F-1 of 0.803 for validation, and 0.825 test with cross validation. The grid search method was used to find the best bar smoothing parameter which allows the model to better detect variations in the data set which results in a better fit of the training data and consequently class predictions. This model improves its score with less number of categorical variables.

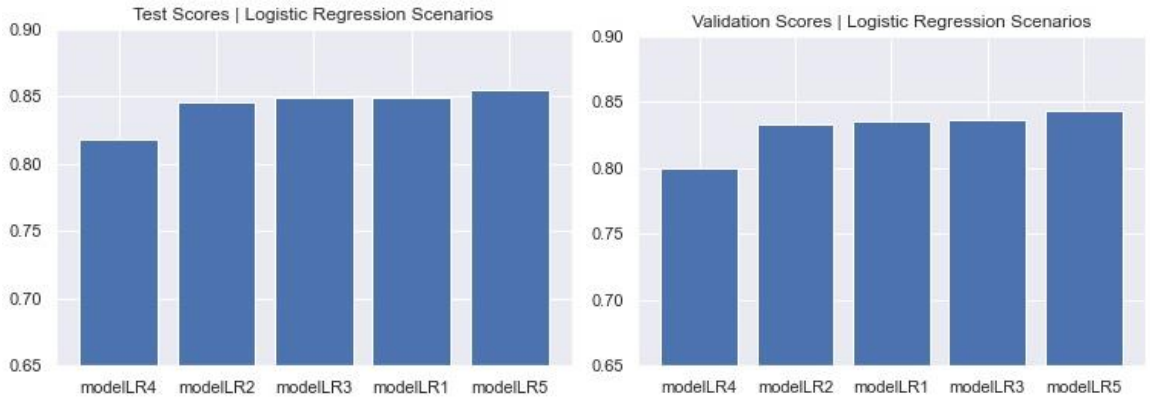

Chart 2. Logistic Regression model LR5

Model LR5 was the best performing using Logistic Regression. It was trained with the variables of Scenario 1 with a parameter C of 10.000 for a resulting F-1 of 0.844 for validation, and 0.852 for test after cross validation. Increasing C results in a more complex model, thus becoming a better classifier for diverse datasets.
ModelLR4 was trained with a balanced database and its score did not improve compared to its other scenarios.

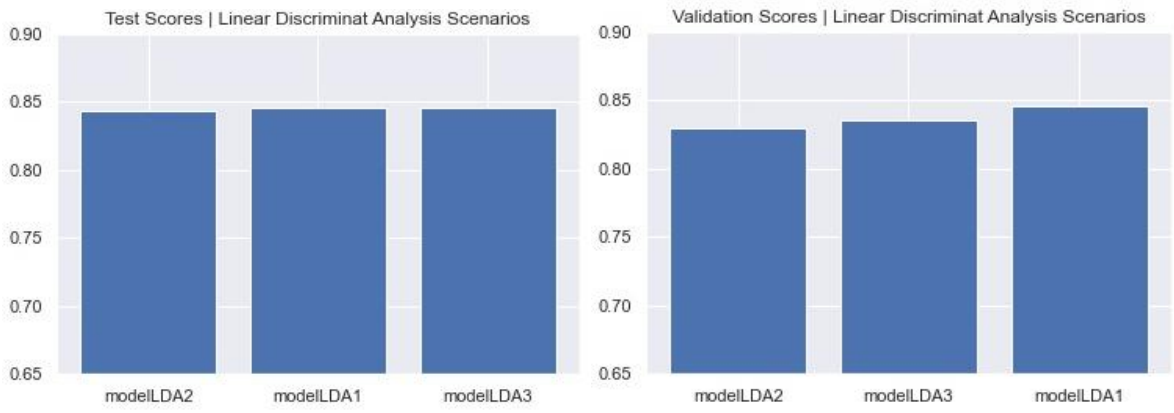

Chart 3. LDA model LDA3

Model LDA3 was the best performing using LDA. It was trained with the variables of Scenario 1, with default parameters, obtaining F1 Score of 0.832 for validation, and 0.843 for test after cross validation.

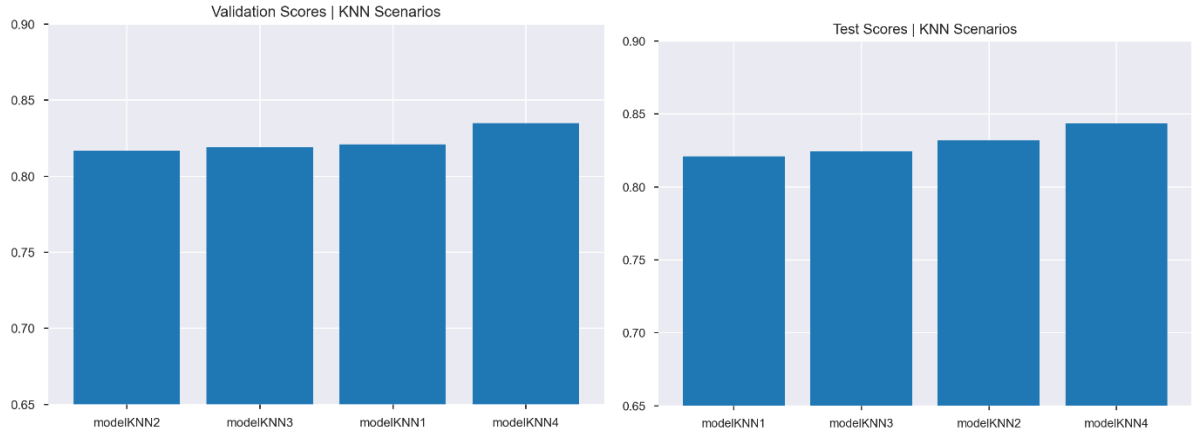

Chart 4. KNN model KNN4

Model KNN4 was the best performing using KNN and was trained with the variables of Scenario 2, with parameters: n_neighbors =15, leaf_size 40, power =1, algorithm='kd_tree', with a resulting F-1 score of 0.829 for validation, and 0.831 for test after cross validation, was the best performing of this set.

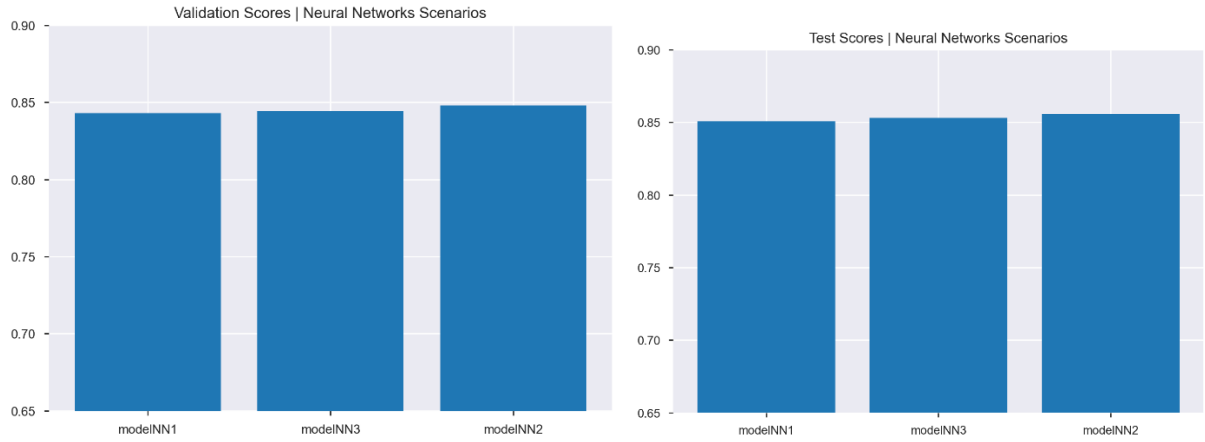

Chart 5. Neural Network model NN3

Model NN2 was the best performer using Neural Network. It was trained with the variables of scenario 2, with default parameters, for a resulting F-1 score of for 0.844 validation, and 0.848 for test after cross validation showed the best result.
We tried tuning the model by setting an adaptative learning rate schedule that keeps the learning rate constant as long as the training loss keeps decreasing in order to accelerate the training and remove the pression from choosing the learning rate. Also, the 3 hidden layers of 100 neurons each theoretically allows the model to better define classes when the distinction of these are not simple. However, the model did not improve.

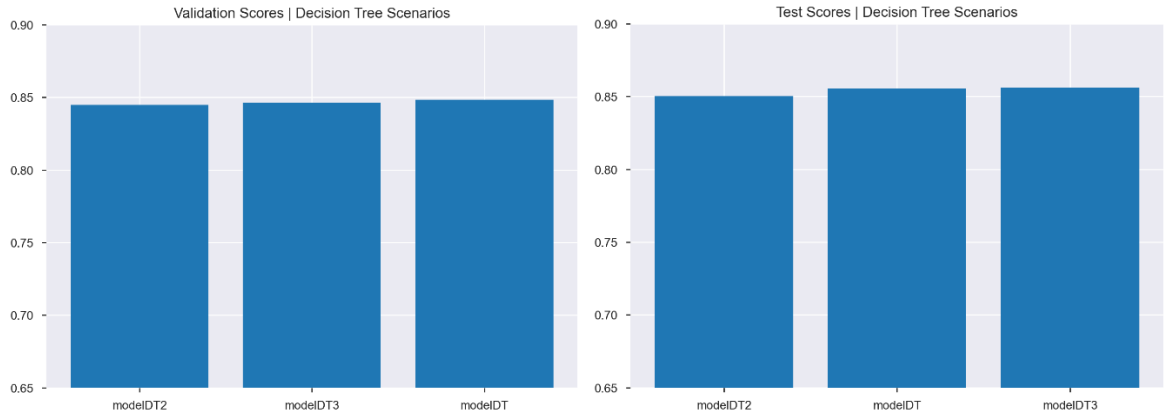

Chart 6. Decision Tree model DT3

Model DT3 was the best performing using Decision Trees. It was trained with the variables of scenario 2, with parameters: max_depth= 9, min_samples_split=20 and splitter='best' for a resulting F-1 score of 0.845 for validation, and 0.848 for test after cross validation.
Defining a maximum number of depth and a higher minimum number of samples, prevents the tree from growing to full development and resolves the problem of overfitting.

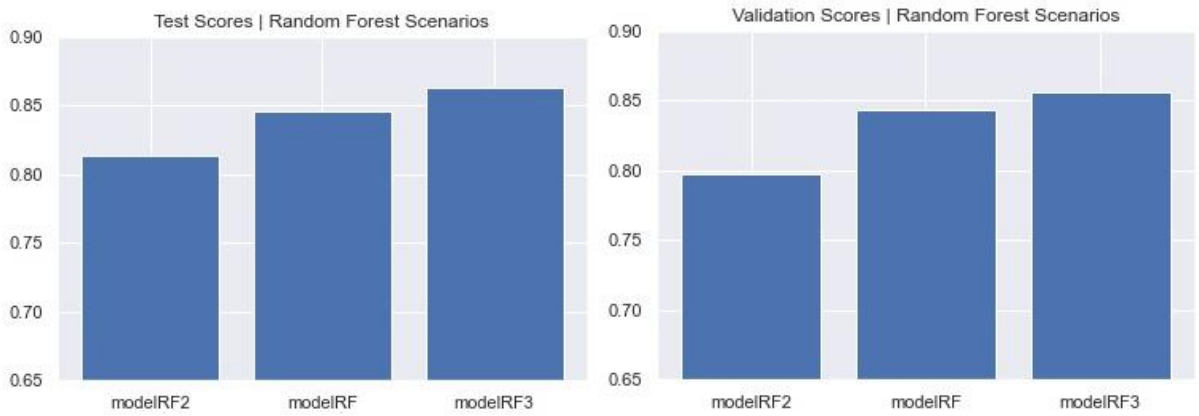

Chart 7. Random Forest model RF3

Model RF3 using Random Forest was the best performer. It was trained with the variables of scenario 1, using grid search to find the best hyperparameters: n_estimators=200, max_depth =20, and max_features =10, for a resulting F-1 score of 0.844 for validation, and 0.851 for test after cross validation.
Taking into account this is already an ensemble model, defining a maximin number of depth, a higher number of estimators and a lower number of features per partition, decreases the depth of each tree and resolves the problem of overfitting.

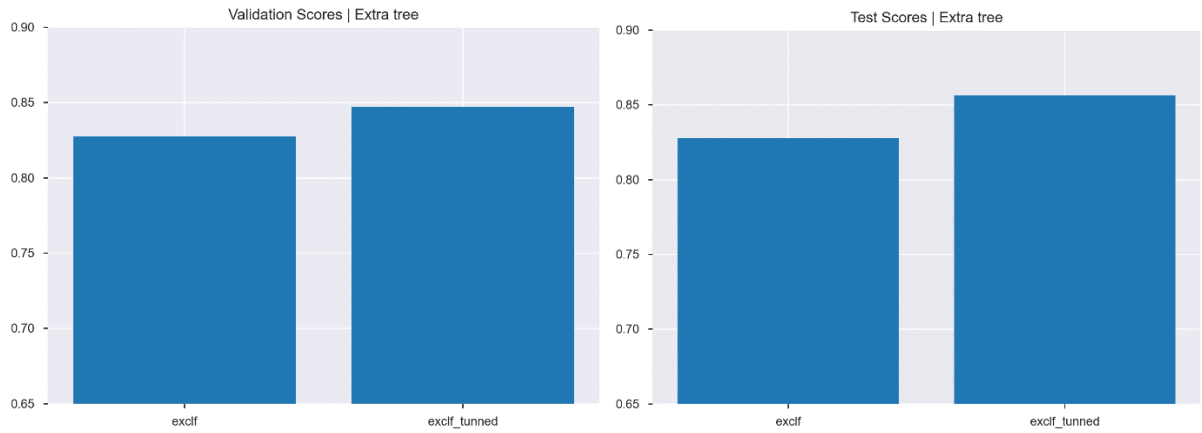

Chart 8. Extra Tree Classifier

Model Extra Tree exclf_tuned using ETC was trained with the variables of scenario 1, with parameter: max_depth= 10, max_features=20, n_estimators=200, for a resulting F-1 score of 0.843 for validation, and 0.850 for test after cross validation.

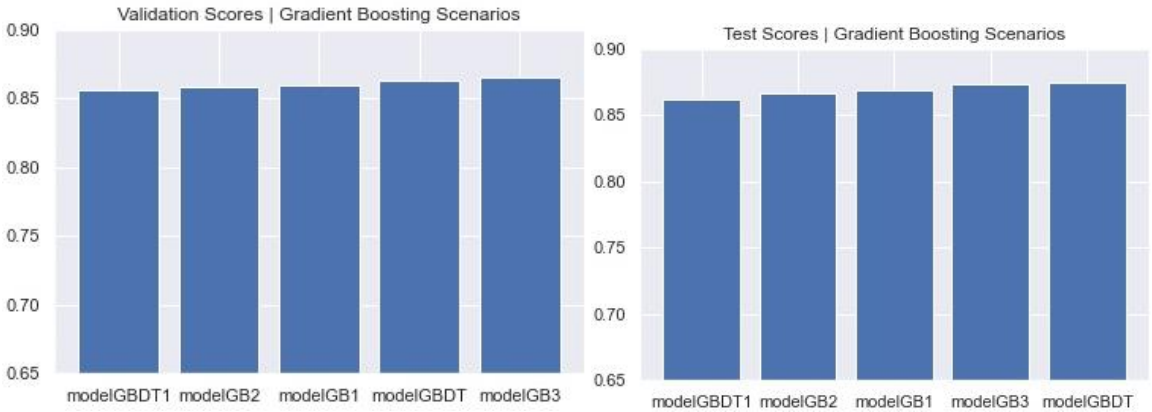

Chart 9. Gradient Boosting model GBDT and GB3

Model GB3 using Gradient Boosting was the best performing and was trained with the variables of scenario 1, with a n_estimators=200 and learning rate of 0.23, obtaining F-1 score of 0.857 for validation, and 0.863 for test after cross validation showed the best scoring.
After running GB3 we decided to improve our model adding more leaves for the decision tree classifier.

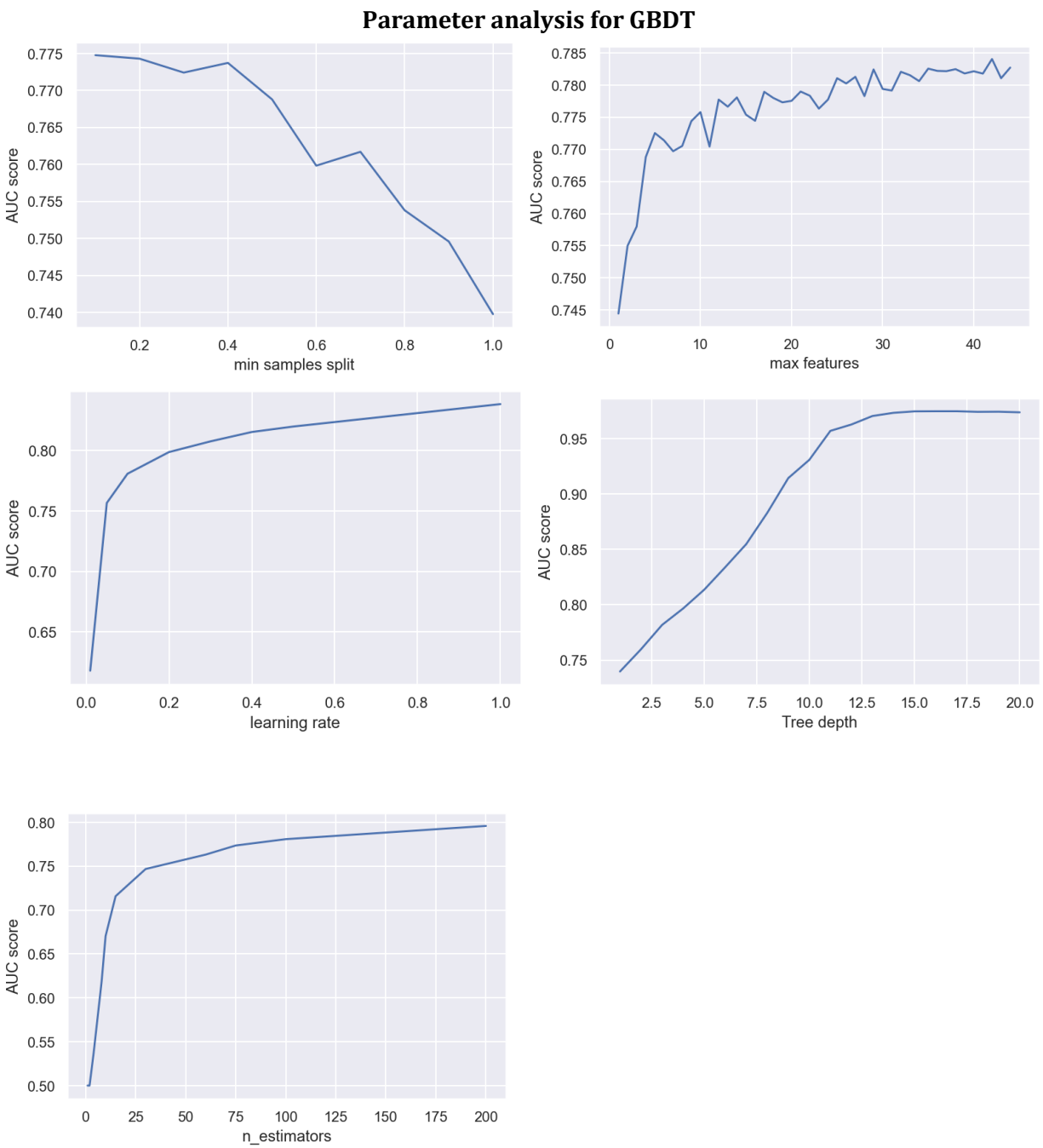

Chart 10. Parameter analysis for GBDT: minimum samples split, maximum number of features, learning rate, Tree depth and number of estimators

Considering the results of the parameter analysis (Chart 10) we set them considering the minimum inflection point, or the lower alpha, preventing overfitting values. Therefore, the parameters were defined as learning rate of 0.2, a max depth of 11, minimum samples split of 0.5, max_features=42, random_state=40 and a number of estimators of 75, for a resulting F-1 score of 0.859 for validation, and for test 0.869 after cross validation.

TRAIN

	precision	recall	f1-score	support
0	0.89	0.96	0.92	10253
1	0.82	0.60	0.69	3187
accuracy			0.87	13440
macro avg	0.85	0.78	0.81	13440
weighted avg	0.87	0.87	0.87	13440

```
[[9833  420]
 [1273 1914]]
```

VALIDATION

	precision	recall	f1-score	support
0	0.76	0.83	0.80	3418
1	0.24	0.17	0.20	1062
accuracy			0.68	4480
macro avg	0.50	0.50	0.50	4480
weighted avg	0.64	0.68	0.66	4480

```
[[2843  575]
 [ 877  185]]
```

TEST

	precision	recall	f1-score	support
0	0.76	0.82	0.79	3418
1	0.22	0.16	0.19	1062
accuracy			0.67	4480
macro avg	0.49	0.49	0.49	4480
weighted avg	0.63	0.67	0.65	4480

```
[[2812  606]
 [ 891  171]]
```

Table 3. Histogram Gradient Boosting results

A model with Histogram Gradient Boosting Classifier was also built, obtaining F-1 scores of 0.850 for validation, and 0.861 for test after cross validation.

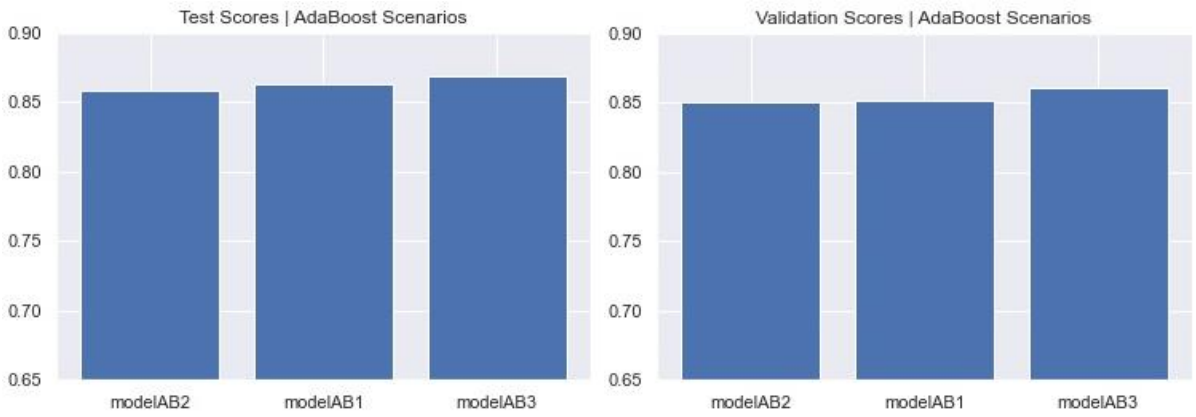

Chart 9. Ada Boost model AB3

The best performing model using AdaBoost was model AB3, trained with the variables of scenario 1, with parameters: n_estimators=350 and a learning rate=0.8, showed the best F1 scores of 0.857 for validation, and 0.864 for test after cross validation within its group. Increasing the number of estimators in order to tune our model improves the prediction of mis-classified samples. This required a lower the learning rate in order to provide a slower convergence and consequently a better performance.

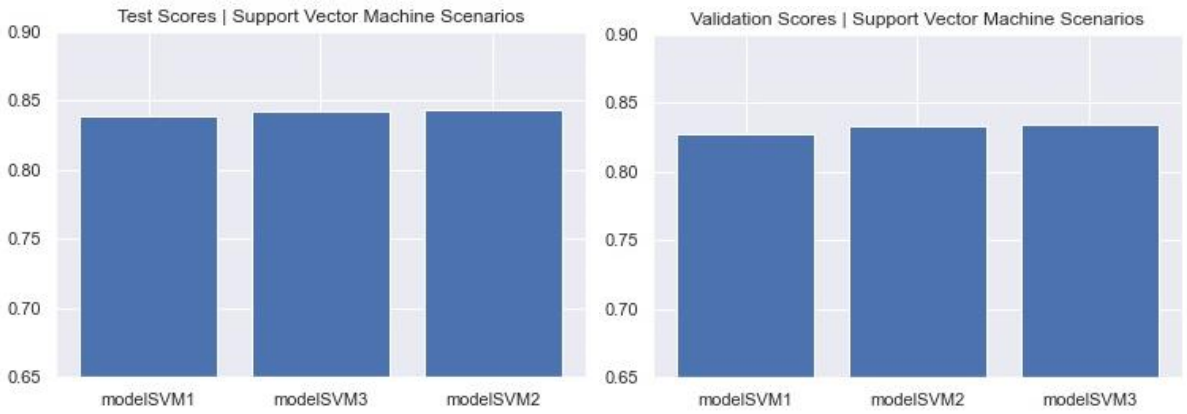

Chart 10. Support Vector Machine model SVM3

The best performing model using Support Vector Machine was model SVM3 trained with the variables of scenario 2, with a lineal kernel, showed a resulting F1 score of 0.829 for validation, and 0.841 for test after cross validation.
A linear kernel provides faster implementation and better performance for databases with high number of features.[9] The better scores obtained showed that our database's classes can be separated using a single line.

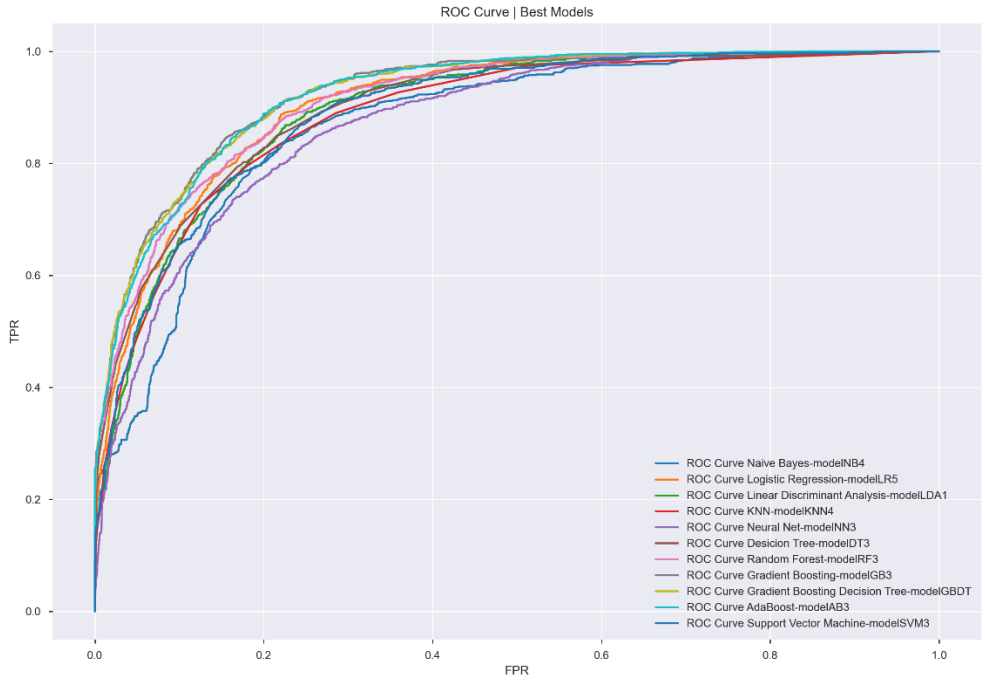

Chart 11. AUC - ROC Curve of Best Models

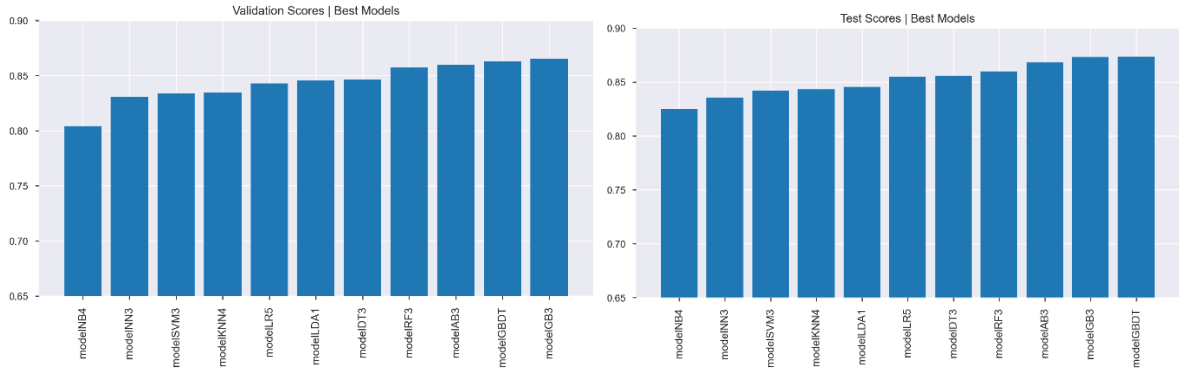

Chart 12. F1 scores of Best Models (validation and test)

After considering all the best preforming scenarios per algorithm, we decided on comparing them by their F1 Score and AUC- ROC curve method, to identify which model was more capable of distinguishing between classes (0 and 1).

1. Ensemble: Bagging, Stacking, Voting Classifiers, Extra Tree

Given the results of the curve, the best individual performing models in descending order were: Gradient Boosting Decision Tree, Gradient Boosting, AdaBoost, Random Forest, Decision Tree, Logistic regression, LDA, and Neural Network.
Because the preliminary results in terms of accuracy were between 0.84 and 0.87 we decided to combine them through bagging, voting classifier and stacking in search of even better performance.

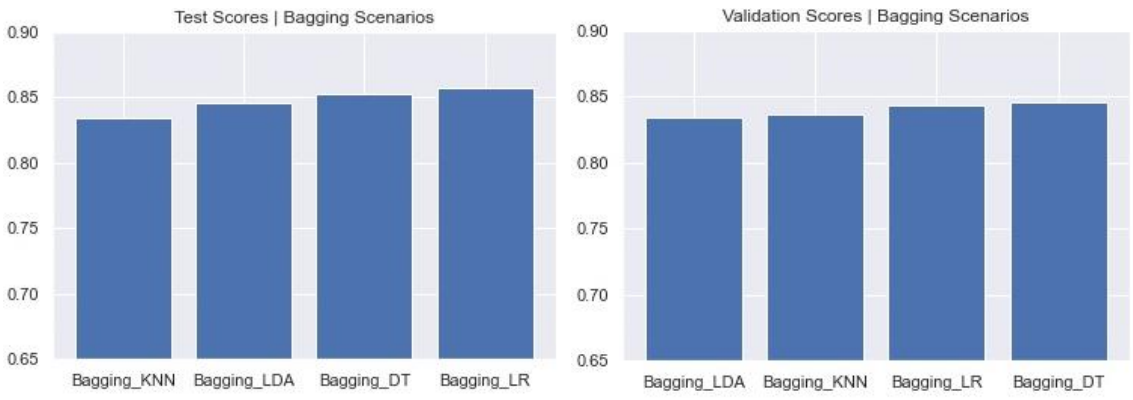

Chart 13. Bagging Scenarios comparison

We tried bagging with KNN, Linear Discriminant Analysis, Decision Tree and Logistic Regression.
Best bagging with Linear Regression using scenario 1, with F1 Score of 0.844 validation and 0.852 test after cross validation.

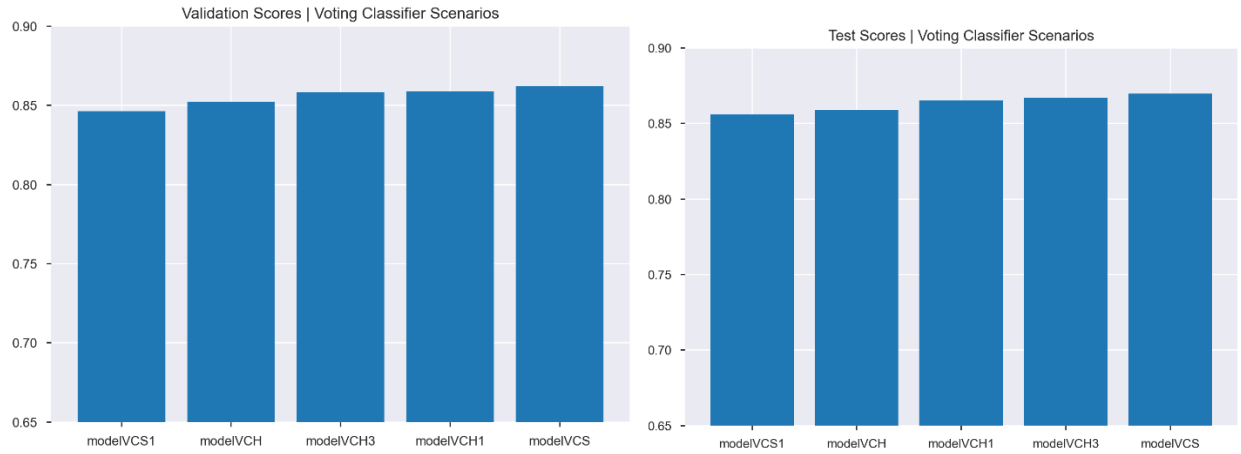

Chart 14. Voting Classifier scenarios comparison

We tried hard and soft voting classifier with three combinations of models as follows:

modelVCH1: Hard Voting | Gradient Boosting, Decision Tree, Logistic Regression
modelVCS1: Soft Voting | Gradient Boosting, Decision Tree, Logistic Regression
modelVCH: Hard Voting | Logistic Regression, Support Vector Machine and Decision Tree
modelVCS: Soft Voting | Logistic Regression, Support Vector Machine and Decision Tree
modelVCH3: Hard Voting | Gradient Boosting, AdaBoost, Logistic Regression

The best performing model using voting classifier was Soft Voting Classifier combining Logistic Regression, Support Vector Machine and Decision Tree (modelVCS) trained with the variables of scenario 1, with a resulting F1 score of 0.848 for validation, and 0.859 for test after cross validation.

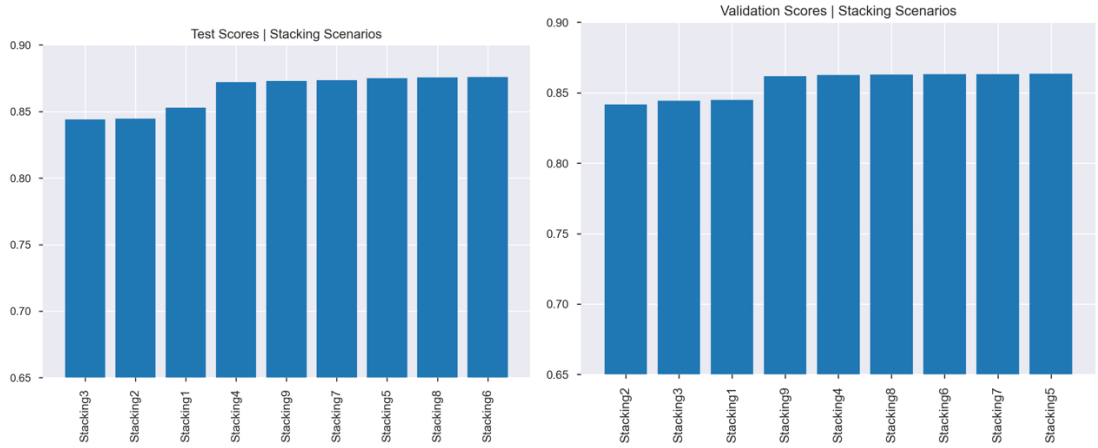

Chart 15. Stacking scenarios comparison

The following combinations were used to run the ensembles:
sc1: LDA, Neural Networks, Logistic Regression
sc2: LDA, Neural Networks, Logistic Regression (predict method)
sc3: LDA, Neural Networks, Gradient Boosting
sc4: Gradient Boosting Decision Tree, Neural Networks, Logistic Regression
sc5: Gradient Boosting Decision Tree, Random Forest, LDA
sc6: SVM, Gradient Boosting Decision Tree, LDA
sc7: Gradient Boosting Decision Tree, AdaBoost, Logistic Regression
sc8: Gradient Boosting Decision Tree, AdaBoost, LDA
sc9: Gradient Boosting Decision Tree, Logistic Regression

The best performing model using stacking was with Gradient Boosting Decision Tree, Logistic Regression (sc9) using scenario 1, resulting in F1 scores of 0.859 validation and 0.870 test after cross validation.

V. DISCUSSION AND CONCLUSIONS

The Newland challenge was the group's first experience applying Machine Learning concepts. As such, it tested and improved our understanding of the models and their limitations, and also the ability to structure and execute a classification project.
A first difficulty faced was the fact that our dataset only included 4 numerical features in a total of 13 features, since working with numerical features to train the models is simpler.
An additional setback resulted from our dataset being unbalanced in the number of occurrences for each label which we anticipated could affect/bias the predictive capacity of our models regarding the minority (0). As a possible solution we decided to apply a combination of oversampling and under-sampling methods, but proved to be unsuccessful as the results of the model trained after treating the imbalance didn't improve.

Trying several models to find the best fit resulted in a very complicated task, mostly because the scores with which we were evaluating performance (accuracy) were very close to each other.
Because of those results, we decided to try to identify other criteria that could rule out any model, such as overfitting (high variance) but the best performing models had a precision score during training of under 90% so it could not be considered as a discriminatory criterion between the models.

After trying several individual models, we decided to try ensemble models in order to find in the combination of weaker models a stronger one and after also trying several combinations the model with most accuracy was the Gradient Boosting Decision Tree with a score of 86,8%.

The next best models resultant from experimental stacking, ranged from 84.1% to 87%, checked by 10-fold cross validation. It is also important to mention that our strategy in creating the stacking models was by using our best performing model, Gradient Boosting Decision Tree, in all our stacking experiments as either a base model or final estimator.

Different performance measures were calculated for these models in order to better understand their predictive potential. Specifically, we calculated the recall, ROC_AUC, the accuracy and the F1 score before cross validation. All these measures showed was the continued similarities in performance of our models, thus forcing us to choose one final model with the highest performance.

The stacking model number 9, is a combination of Gradient Boosting Decision Tree as base estimators, and Logistic Regression as the final estimator. These two models allow for two different perspectives on the dataset which is the key for the ensemble model's good performance (Table 4).

	Gradient Boosting Decision Tree	Stacking 4	Stacking 5	Stacking 7	Stacking 9
Recall	0.652	0.636	0.675	0.643	0.650
Accuracy (Mean F-1)	0.873	0.872	0.875	0.873	0.873
ROC_AUC	0.797	0.791	0.806	0.793	0.796
Accuracy (Mean F-1) (cross_validation)	0.869	0.869	0.868	0.870	0.870
Kaggle Score	0.8624	0.865	0.864	0.863	0.863

Table 4. Performance Score Comparison

Our team firmly believes that because our training dataset was originally imbalanced with higher occurrences of citizens with income below average (those with label 0), it is understandable that all our models showed higher capacity in predicting the majority class. Keeping this in mind, our chosen model – Stacking Gradient Boosting Decision Tree & Logistic Regression – is not only a logical option, since boosting type of ensemble models are built with the objective of outperforming individual classifiers, but are also recommended for decreasing model bias. As a result, combining a complex classifier model as Gradient Boosting with a simple model as logistic regression allowed us to achieve a better performance.

VI. REFFERENCES

[1] Friedman, J.H. (2002) Stochastic gradient boosting. Computational statistics & data analysis

[2] Machine Learning Masteriy, September 28, 2020, Linear Discriminant Analysis with Python https://machinelearningmastery.com/linear-discriminant-analysis-with-python/

[3]Friedman, J.H. (2001) Greedy Function Approximation A Gradient Boosting Machine. Annals of Statistics

[4] Scikit Learn, Sklearn.Ensemble.ExtraTreesClassifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

[5] Eric Bauer, Ron Kohavi (1998). An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants [Kluwer Academic Publishers, Boston. Manufactured in The Netherlands]. https://ai.stanford.edu/~ronnyk/vote.pdf

[6] Github, Juliano García, August 26, 2020, Histograms for efficient Gradient Boosting https://robotenique.github.io/posts/gbm-histogram/

Neptune, August 24, 2020, Hyperparameter Tuning in Python: a Complete Guide https://neptune.ai/blog/hyperparameter-tuning-in-python-a-complete-guide-2020

[8] Towards Data Science, June 26, 2018, Unnderstanding AUC-ROC Curve https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

9 Towards Data Science, june 7th of 2018, Support Vector Machine — Introduction to Machine Learning Algorithms, https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47