Machine Learning Project Report

Income Prediction for the Newland citizens using Machine Learning

Authors (Group 27):

Gabriel Cardoso (m20201027@novaims.unl.pt)
João Chaves (m20200627@novaims.unl.pt)
Nguyen Huy Phuc (m20200566@novaims.unl.pt)
Anastasiia Tagiltseva (m20200041@novaims.unl.pt)

December 2020

Table of Content

Abstract

The project presents the results of applying machine learning methods suitable for predicting income from the proposed dataset using supervised machine learning algorithms: Logistic Regression, Stochastic Gradient Descent classifier, K-Neighbors, Decision Tree, Gaussian Naive Bayes, Random Forest, Support Vector, Gradient Boosting, Ada Boost algorithm and Multi-layer Perceptron classifier, and compare their performances to obtain the best performing classifier and then use Stacking to increasing the predictive force of the classifier.

The result is that the best model is Gradient Boosting with average F1 Score of 86.96% using 10-fold cross-validation over 22,400 observations in the training dataset

Keywords: Project, Machine Learning, Predictive Modelling, Phi_K Correlation, Min Max Scaler, One Hot Encoder, Logistic Regression, Random Forest, Neural Network, Gradient Boosting, Grid Search Algorithm, Stacking Classifier

I. INTRODUCTION

The research client - the government of the Newland - wishes to create a predictive model of income for people on their way to Newland to apply a binary tax rate (15 or 30%) and make the new city more financially sustainable.

The group was given a data set of 22,400 observations to create a predictive income model below or above average (0 or 1).

This model was applied to 10,100 new observations (test dataset) and uploaded to Kaggle. Since, in this study, it was implemented a feature importance analysis, this report also aims to conduct a comprehensive analysis to highlight the key factors that have a higher chance of influence the income, and therefore, the tax payment for citizens and tax revenue for the state, one of the most important sources of revenue for any country or state.

This paper has been structured as an introduction, background, proposed methodology, results, discussion and conclusion.

II. BACKGROUND

It is a common assumption to test multicollinearity before selecting the variables into regression model. Multicollinearity happens when independent variables in the regression model are highly correlated to each other. It makes it hard for interpretation of model and also creates overfitting problem. To check whether Multi-Collinearity occurs was plotted the correlation matrix of all the independent variables. And because we have both categorical and interval variables was calculated φK correlation coefficient, based on several refinements to Pearson's hypothesis test of independence of two variables. The combined features of φK form an advantage over existing coefficients. First, it follows a uniform treatment for interval, ordinal and categorical variables Second, it captures non-linear dependency. Third, it reverts to the Pearson correlation coefficient in case of a bi-variate normal input distribution [1].

For the purpose of having a solid consolidation of the weight of each feature to the dependent variable, Weight of Evidence encoding technique fundamentals were applied together with Ordinal Encoding.

Since, in train data, dependent variable is binary and is available, a function was created to obtain the following weight,

$$Weight(Y = 1|Xj = 1) = \frac{Nºobs\ (Y = 1 \cap Xj = 1)}{Nºobs\ (Xj = 1)}$$

which represents the weight of each attribute (Xj) of each categorical variable(X) assign to an income taxpayer(Y=1) (that could be calculate by the sum of its occurrences divided by the number of observations of that specific sample). In order to avoid a possible creation of an overfitted model, ordinal encoding was used to make it easier to show the discrimination of the similarity/distance between each feature. This method would allow us to reduce the number of features by grouping them by approximate probability, preventing us to face the curse of dimensionality that it is always associated when the number of elements of each feature is high. This technique would allow us to implement any model learnt with some kind of relevance since the data points were reflecting the somewhat accurate distance. In other words, this led the study to explore models like k-nearest neighbors' algorithm with a more concise data, before using other approaches (encoding and modelling wise).

To identify the best combination of used models has applied the Super Learner algorithm (also known as a Stacking Ensemble) from the ML-Ensemble (mlens) python library.

The super learner algorithm is a supervised ensemble algorithm that uses K-fold estimation to map a training set (X,y) into a prediction set (Z,y), where the predictions in Z are constructed using K-Fold splits of X to ensure Z reflects test errors, and that applies a user-specified meta learner to predict y from Z.

In other words, the super learner is an ensemble machine learning algorithm that combines all the models and model configurations that might be investigated for a predictive modeling problem and uses them to make a prediction as-good-as or better than any single model that may have been investigated [2].

III. METHODOLOGY

The Data set

The train data set includes figures on 22400 different records and 14 attributes, which consist of 9 categorical and 5 continuous attributes as shown in Table 1. The binomial label in the data set is the income level that predicts whether a person earns more than average income or not.

Variable	Type	Data type	Description
Citizen_ID	continuous	int64	Unique identifier of the citizen
Name	categorical	object	Name of the citizen (First name and surname)
Birthday	categorical	object	The date of Birth
Native Continent	categorical	object	The continent where the citizen belongs in the planet Earth
Marital Status	categorical	object	The marital status of the citizen
Lives with	categorical	object	The household environment of the citizen
Base Area	categorical	object	The neighborhood of the citizen in Newland
Education Level	categorical	object	The education level of the citizen
Years of Education	continuous	int64	The number of years of education of the citizen

Employment Sector	categorical	object	The employment sector of the citizen
Role	categorical	object	The job role of the citizen
Working Hours per week	continuous	int64	The number of working hours per week of the citizen
Money Received	continuous	int64	The money paid to the elements of Group B
Ticket Price	continuous	int64	The money received by the elements of Group C
Income	binary	int64	The dependent variable (Where 1 is Income higher than the average and 0 Income Lower or equal to the average)

Table 1. – Features name and respective type, data type and description.

After the exploration and understanding of data, fixing possible problems on data like missing values or outliers is mandatory, so it was created new variables in order to get features with higher predictive power.

After investigating the data, it was carried out some early analysis, like the following:

- Gender could be generated using the title stated in 'Name' column;

It was handled by checking the first string before '.' in Title Name: if it was 'Mrs' or 'Miss', that would be changed it to 'Female' or else would return 'Male'.

- Age of the citizen could be generated using the Birthday information;

Converting Birthday to age using current year, according to description (2048). In the description of the data set, it is written that the variable Age of citizens (Birthday) only assumes values from 17. This condition was checked it out, to be sure that the data set didn’t have any inconsistencies.

- Missing values in categorical features are represented under the value '?';

Handling missing data is important, as many machine-learning algorithms do not support data with missing values. It was replaced '?' with NaN value to detect easily and the number of missing values in each feature like figures below demonstrate:

----------- Train dataset % missing -------------	
Role	5.67
Employment_Sector	5.64
Base_Area	1.76

----------- Test dataset % missing -------------	
Role	2.54
Employment_Sector	2.54
Base_Area	0.83

Table 2 - Missing values in Train and Test Data as % of the respective total of observations of each data set.

Data cleaning was required so it could be fitted to any future model. In other to accomplish that, missing values were treated with the mode relative to each categorical feature; in addition to that, outliers were explored, but the removal of them would mean that a big amount of data would be dropped. Since the data is census data, which has been verified by the Government, therefore the outliers are kept and used in this study.

As we can see (in the table above) the data set contains a certain set of missing values for categorical features: Role, Employment Sector, Base Area which has been dealt with mode transformations applied to the data. However, in our final model, it was decided to handle the missing values for every attribute by setting a default marker called ‘Unknown’ and assigning a unique category for negating information loss.

Coherence check:

This section is necessary to make sure the data makes logical sense. Below it's possible to see what were the logical rules that we looked to verify and confirm irrelevant or partially relevant features that could negatively impact the model performance.

Besides that, having correlated independent data can also harm the model. In this way, it was checked the correlation matrix in this step using phi k described in Background part.

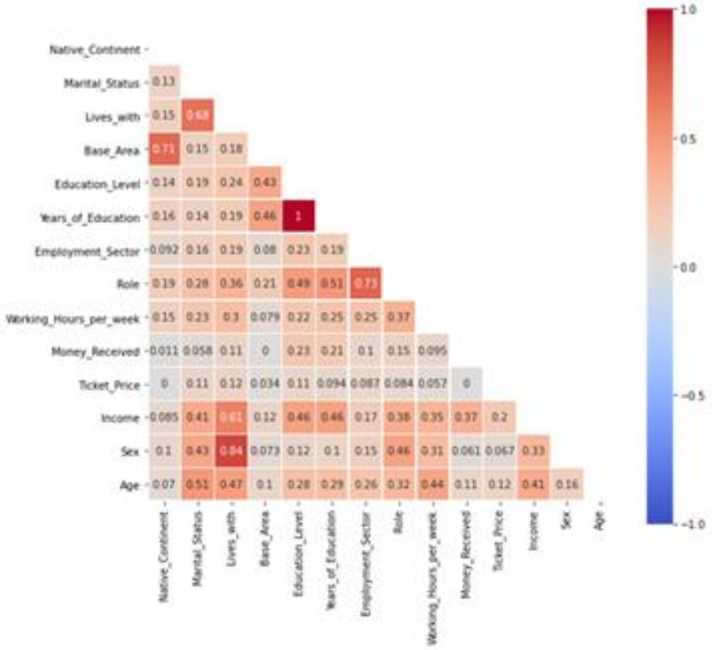


Fig.1 – Training data’s correlation matrix heatmap.

Because of the nature of the first encoding technique (prone to overfitting), it was necessary to check if there were any features that were highly correlated with the dependent variable. In the heatmap above, it can be observed that some of the independent variables are highly correlated to each other.

When independent variables are highly correlated, change in one variable would cause change to another and so the model results fluctuate significantly. The model results will be unstable and vary a lot given a small change in the data or model. This will create the following problems: It would be hard to choose the list of significant variables for the model if the model gives different results every time. Coefficient Estimates would not be stable, and it would be hard to interpret the model [3].

From the heatmap above we can see strong correlation (>0.8) between features Years_of_Education and Education_level and between features Sex and Lives_with. The most straight-forward method is to remove some variables that are highly correlated to others (Education_level and Sex) and leave the more significant ones in the set.

To fix multi-collinearity issue was decided transform some of the variables to make them less correlated but still maintain their feature and one hot encoding. But even after this the correlation between Years_of_Education and Education_level was significant. The feature ‘Education_level’ was removed because according to the matrix relation between ‘Income’ and ‘Years_of_Education’ is stronger.

The correlation results after this became much more acceptable and we were able to include all other variables as model features.

Feature importance was deducted using the first method described in Background Section that led us to understand better each feature's category relationship with the dependent variable (Income) as shown in the following figures below.

It will serve as a reference for categorical encoding, using two different types of techniques that can provide an encoded value that could translate better the distance/similarity between categories and the Income feature. Those techniques are:

1) *Ordinal Encoding*: following the definition, "the integer values have a natural ordered relationship between each other and the machine learning algorithms may be able to understand and harness this relationship", although, in this study, that ordered relationship was created based on the probability referred above. This way, they were converted into new categories, however, the number of those was so reduced, that it enables us to use the technique below, in order to avoid the curse of dimensionality without losing important correlations.

2) *One-Hot Encoding*: This method involves splitting off different categorical features into its own categories where each and every category assumes a binary value.

3) *Weight of Evidence Encoding:* This method encodes categorical features replacing labels by a binary classification based on the weight of evidence or ratio of probabilities.

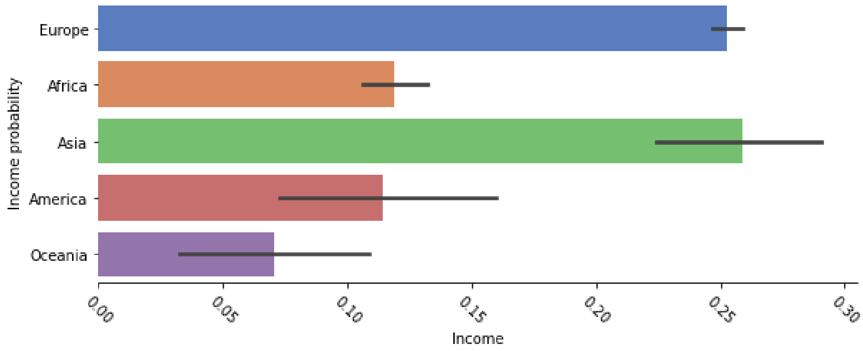


Fig.2- Native Continent:

For this feature, we will use OneHotEncoding technique since there is a lower number of features and their weight does not have a large variation (See Encoding).

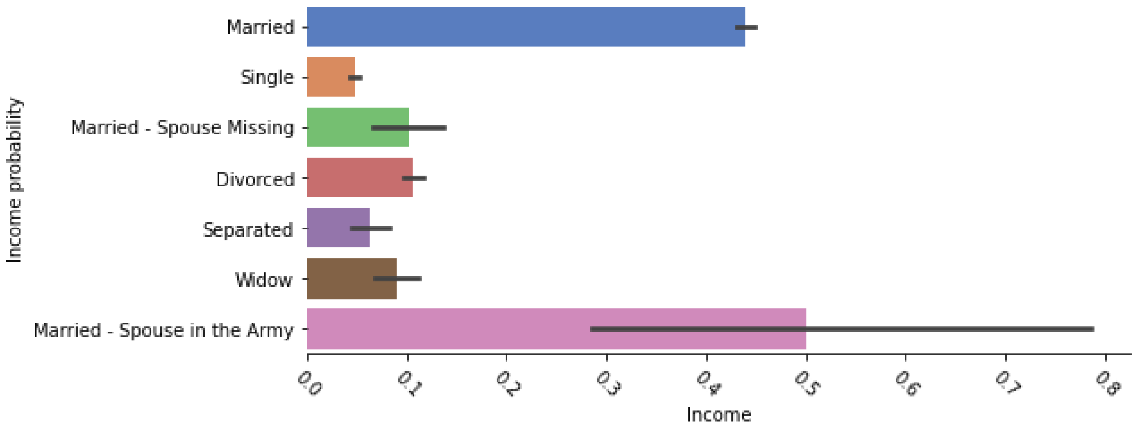


Fig.3 - Marital status

Since 50% of the “Married” people are going to pay higher taxes, we grouped 'Married - Spouse in the Army' and 'Married' into one status, 'Married' people. The next group, according to weight in the target variable, are statuses 'Married - Spouse Missing,' 'Separated' and 'Widow' that were added in a new feature called 'Married but Single.' Status 'Single' was left as it is, and for futures unknown values, it was decided to have 'Other marital' status.

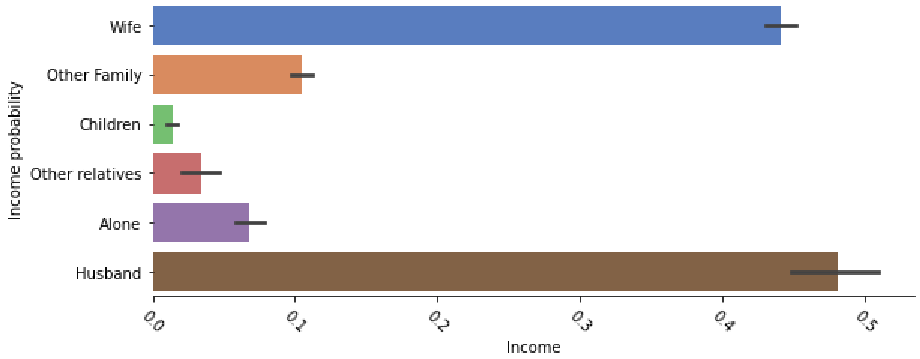


Fig.4- Lives With

Based on the weight analysis, it can be easily observed that people who live with wife or husband have a significantly high percentage of having a high income. A possible reason to that would be because they have developed a good career and settled down, thus their income status should be better. Followed by people who are living with other family or single, this could be young people who have just gotten a job and still not able to afford their own accommodation yet. Finally, people who live with children can highly be a widow or broken marriage, thus their status of the economy could be harsher. Therefore, it was decided to divide the variable into 3 groups as discussed above.

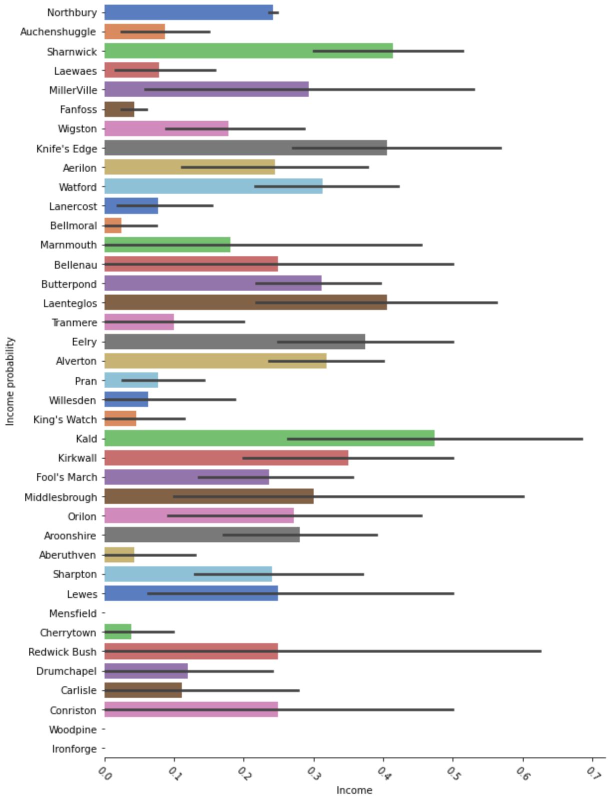


Fig.5 - Base area

By exploring the data, there's something interesting in Base Area - people from space expedition are mostly based in Northbury. To deal with this situation, WoE encoding was applied and the feature Base Area was renamed and reconverted to a binomial feature where Northbury residents would be given a value of 1 and all non-resident of Northbury would return 0.

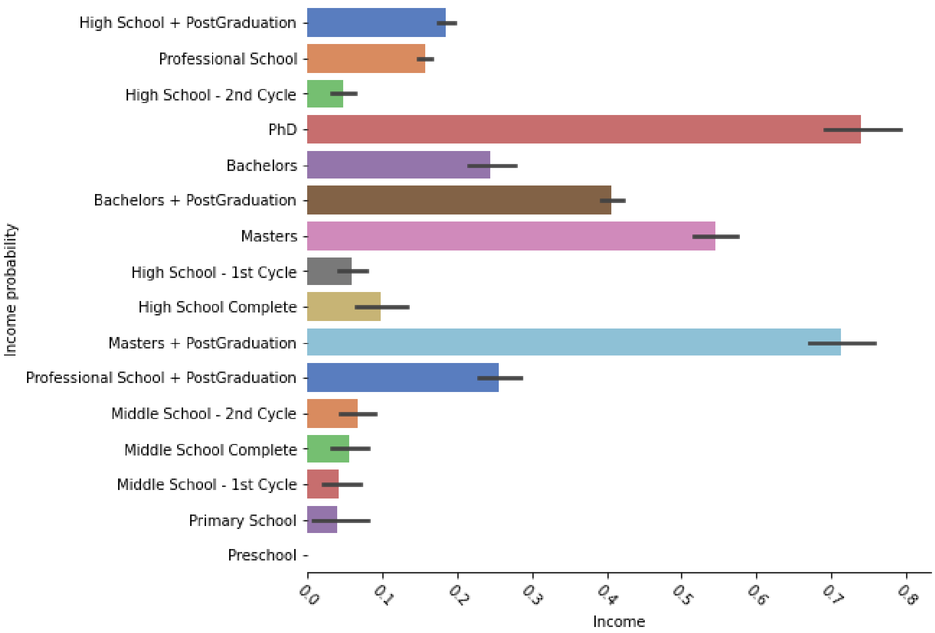


Fig.6 - Education Level

This feature was also grouped by the importance of each category to the dependent variable.

- 'Post_Masters': (Phd,Masters + PostGraduation)
- 'Masters': (Masters)
- 'Post_Grad': (Bachelors + PostGraduation)
- 'Bachelor': (Bachelors,Professional School + PostGraduation)
- 'No_Bachelor': (High School + PostGraduation,Professional School);
- 'Without_high_education': otherwise.

It was decided to drop this feature because of strong correlation between it and 'Years_of_education' even after transformation (Fig.1).

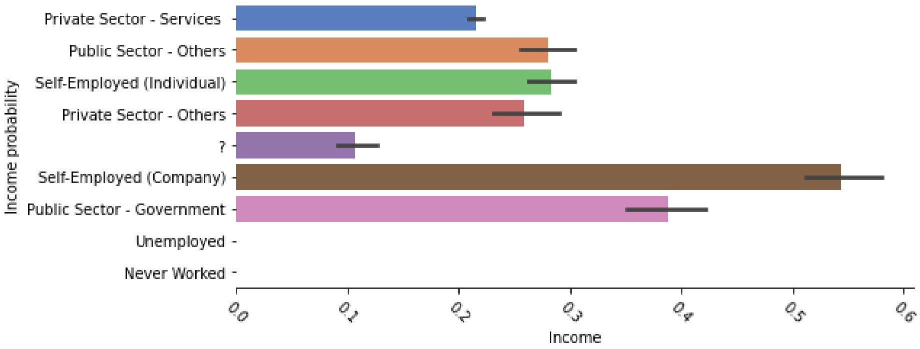


Fig.7 - Employment Sector

To reduce the ‘Employment sector’ number of features, it was aggregated the combination of sectors “'Private Sector - Services” and 'Private Sector - Others' into a new category called “Private Sector”. Sectors “Unemployed,” “Never Worked,” and “Unknown” were concatenated in 'Other employment’.

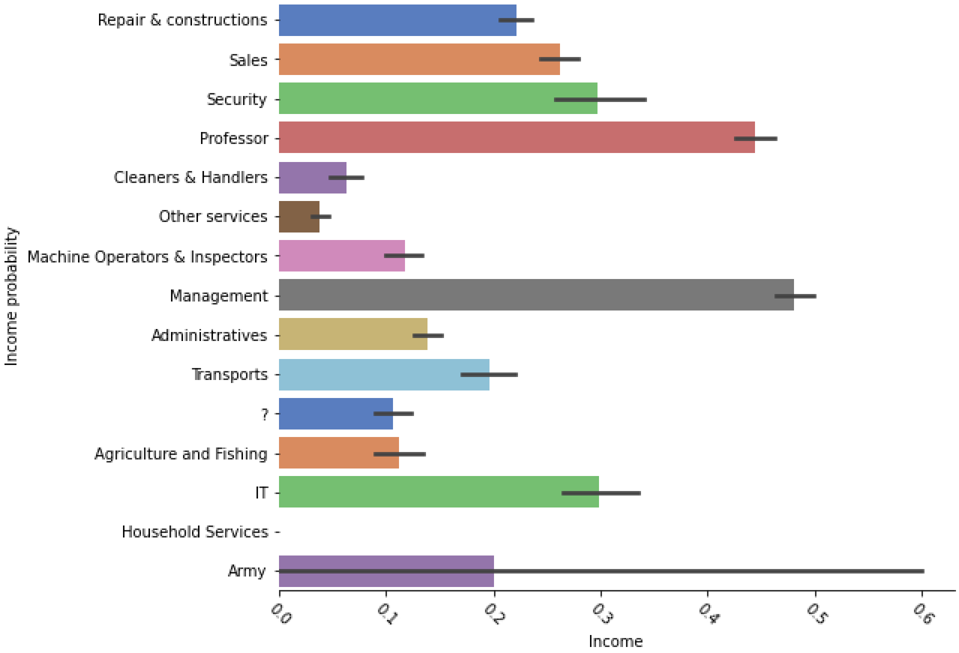


Fig.8 - Role

The same thought process was applied in this feature. It was grouped by the importance of each category to the dependent variable.

- 'Very High': Management, Professor;
- ‘High': IT, Security, Sales;
- 'Medium': Repair & constructions, Army, Transports;
- 'Low': Administratives, Machine Operators & Inspectors, Agriculture and Fishing;
- 'Other role': otherwise

Data Normalization

It is a good practice to normalize the data by putting its mean to zero and its variance to one (StandardScaler by scikit-learn), or to rescale it by fixing the minimum and the maximum between -1 and 1 or 0 and +1 (MixMaxScaler by scikit-learn) [4].

The transformation is given by:

X_std = (X - X.min) / (X.max-X.min)

*X_scaled = X_std * (max - min) + min*

where min, max = feature_range.

Model selection

Ten popular classifiers are initially evaluated on the suitability with the dataset. It was identified which classifiers that perform better with this data set and continue to tune the model in the later section. All ten classifiers are developed by scikit-learn [5]: Logistic Regression, Stochastic Gradient Descent, K-Nearest Neighbors, Decision Tree Classifier, Gaussian Naïve Bayes, Random Forest, Linear Support Vector Machine, Gradient Boosting, AdaBoost, Multi-layer Perception classifier (Neural Network Classifier).

Detailed Model selection

Since some models were having better results, it was decided to explain briefly, each one of the best.

Logistic regression is a statistical model that uses a logistic function to model a binary dependent variable. Since our dependent variable is binomial and logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables [6].

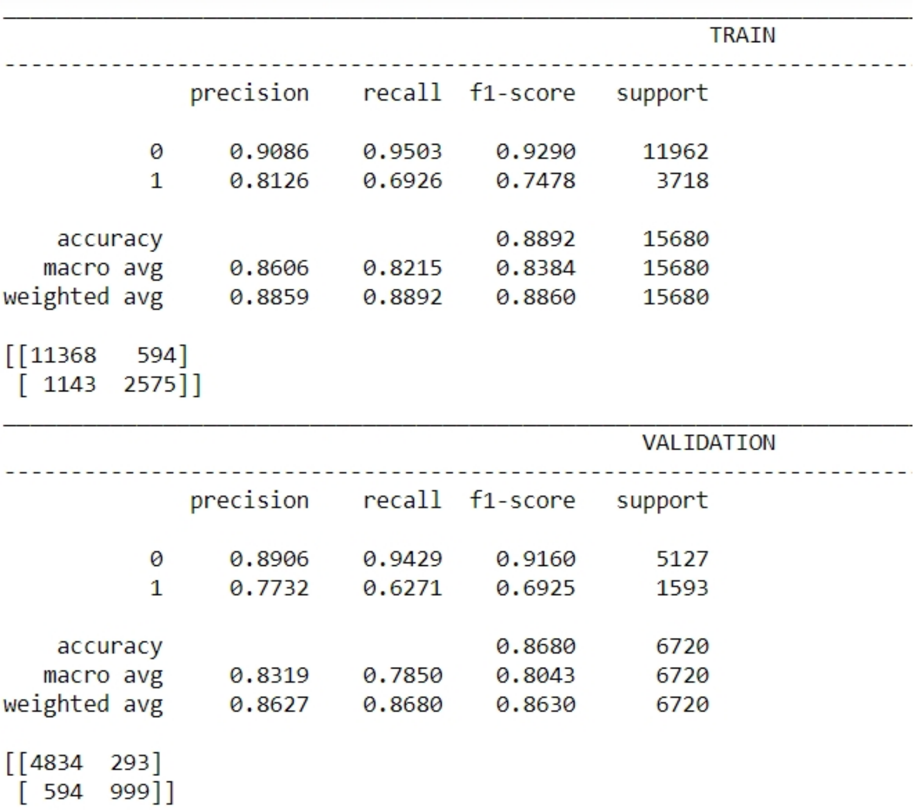
Random Forest: First, the Random Forest(RF) is a classifier that evolves from decision trees. It consists of many decision trees. To classify a new instance, each decision tree provides a classification for input data; RF collects the classifications and chooses the most voted prediction as the result. The input of each tree is data from the original dataset. In addition, a subset of features is randomly selected from the optional features to grow the tree at each node. Each tree is grown without pruning. Essentially, random forest enables many weak or weakly-correlated classifiers to form a strong classifier [7].

Boosting: This classifier goes into the category of the previous one, using trees it allows for the optimization of arbitrary differentiable loss functions. They often consider homogeneous weak learners, learns them sequentially in a very adaptative way (a base model depends on the previous ones) and combines them following a deterministic strategy [8].

Stacking Model: the idea of stacking is to use several different weak learners and combine them by training a meta-model to output predictions based on the multiple predictions returned by these weak models. So, in order to build a stacking model, two things must be defined: The learners we want to fit and the meta-model that combines them. In contrast with boosting, stacking consider heterogeneous weak learners, in other words, weak learners are fitted independently from each other’s [9].

Training the model

The Gradient Boosting Classifier Model is tuned using Grid Search Algorithm for getting the best set of hyper-parameters. After training the model with Grid-Search applied on GBC, learning rate of 0.1, maximum depth of 4, max features of 7, min samples leaf of 15, min samples split of 120, subsample of 0.8, 500 estimators and random state of 10 are obtained. The summary of Grid-Search Tuning of GBC model on the basis of the Mean Score is shown in Table below.



```
________________________________________________________________________________
                                                            TRAIN
--------------------------------------------------------------------------------
                precision    recall  f1-score   support

           0       0.9086    0.9503    0.9290     11962
           1       0.8126    0.6926    0.7478      3718

    accuracy                           0.8892     15680
   macro avg       0.8606    0.8215    0.8384     15680
weighted avg       0.8859    0.8892    0.8860     15680

[[11368   594]
 [ 1143  2575]]
________________________________________________________________________________
                                                       VALIDATION
--------------------------------------------------------------------------------
                precision    recall  f1-score   support

           0       0.8906    0.9429    0.9160      5127
           1       0.7732    0.6271    0.6925      1593

    accuracy                           0.8680      6720
   macro avg       0.8319    0.7850    0.8043      6720
weighted avg       0.8627    0.8680    0.8630      6720

[[4834  293]
 [ 594  999]]
```

Table 3. The summary of Grid-Search Tuning of GBC model

IV. RESULTS

Preprocessing techniques which were used for the feature engineering original dataset: One-Hot Encoding, Weight of Evidence Encoding, grouping features by the importance of each category to the dependent variable.

Model selection:

Ten models initially chosen are trained using the whole dataset. To prevent overfitting by using the whole dataset, we used stratified 10-fold cross-validation (CV). Mean score, standard deviation, minimum, maximum and average training time are recorded for comparison (as shown in table below).

Model	Average Score	Standard Deviation	Minimum	Maximum	Average training time (in seconds)
Logistic Regression	**0.8473**	+/-0.006	0.8348	0.8584	0.197
SGDC	**0.8438**	+/-0.006	0.8348	0.8540	0.060
KNN	**0.8291**	+/-0.011	0.8004	0.8418	0.717
CART	**0.8157**	+/-0.01	0.8004	0.8342	0.048
GaussianNB	**0.7755**	+/-0.011	0.7487	0.7883	0.012
Random Forest	**0.8447**	+/-0.007	0.8335	0.8571	0.753

LinearSVC	**0.8492**	+/-0.007	0.8393	0.8648	0.102
Gradient Boosting	**0.8652**	+/-0.006	0.8546	0.8776	0.978
AdaBoost	**0.8594**	+/-0.007	0.8463	0.8686	0.380
Neural Network	**0.8516**	+/-0.008	0.8361	0.8661	13.184

Note on abbreviation: SGDC: Stochastic Gradient Descent Classifier; KNN: K-Nearest neighbors; CART: Decision Tree Classifier

Table 4. 10-fold cross-validation results for each model used.

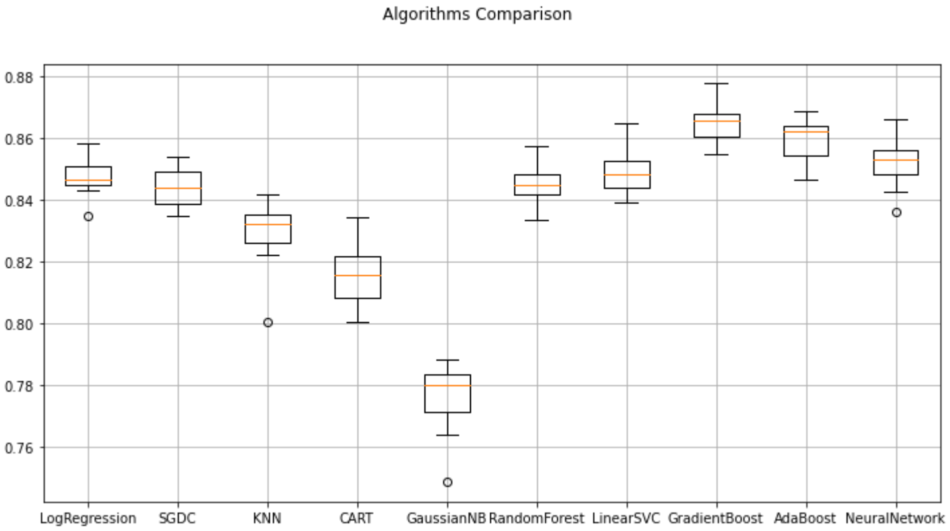


Fig.9. Box plot of the CV scores

From the result of 10-fold cross-validation, it was chosen Logistic Regression, one neural network model (MLPClassifier) and two ensemble models (Random Forest and Gradient Boosting). In the next section, further tunning in order to optimize the hyperparameter were carried out on these models to find the good fit for this data set.

Learning curves

Learning curves are a good way to see the overfitting effect on the training set and the effect of the training size on the accuracy. Reviewing learning curves of models during training can be used to diagnose problems with learning, such as an underfit or overfit model, as well as whether the training and validation datasets are suitably representative [10].

In the model selection state, it was used the learning curves plot to evaluate different models in term of diagnosing (1) whether the train or validation datasets are not relatively representative of the problem domain; (2) whether the model is underfit, overfit, or well-fit [8]; (3) variation of each model's score.

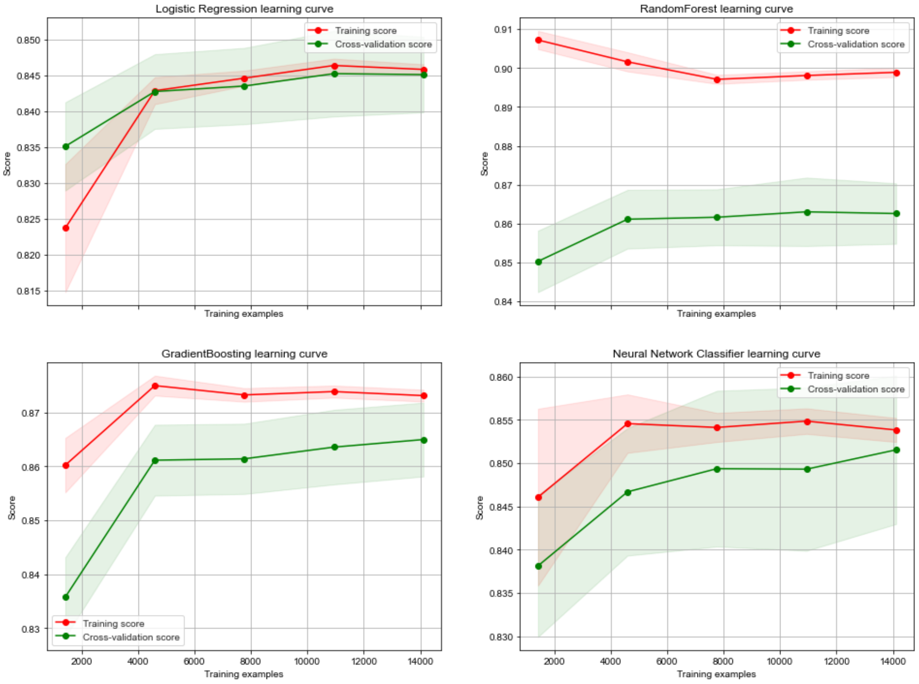


Fig. 10: Scores mean of different models by training examples in ten k-fold cross-validation.

From the plots above, it can be concluded that after 8000 training examples, more training won’t be as meaningful for the score in all models, except for Logistic Regression that it excels with only half of the training examples previously needed. Also, it can be noticed a little bit of overfitting in the Random Forest model since the gap between training data and validation data is a bit significative (4% difference in score). However, all other models have a difference observed not meaningful to be assume that the overfitting problem is present in each one of them.

Hyperparameter Tuning

GridSearchCV was used on the training dataset to develop the best hyperparameter to accurately predict the income. Hyperparameter grid for searching were designed based on the basic attributes of the data set such as number of categorical and numeric features, number of records, data type and further research. The model with the best hyperparameter where then trained and tested again on the split datasets and another 10-fold cross-validation.

Model	**Logistic Regression**	**Neural Network**	**Random Forest**	**Gradient Boosting**

Hyperparameter	solver=' lbfgs ',
	C = 300,
	tol=0.0001,
	max_iter=100,
	random_state=2
		hidden_layer_sizes = (100,25),
		activation = ' tanh ',
		solver = 'sgd',
		learning_rate_init = 0.1,
		learning_rate = 'adaptive',
		batch_size = 'auto',
		max_iter = 300,
		momentum=0.7
			bootstrap=False,
			max_depth=9,
			max_features=10,
			min_samples_leaf=3,
			min_samples_split=10,
			n_estimators=300,
				loss = ' deviance ',
				max_depth=4,
				max_features=7,
				min_samples_leaf=15,
				min_samples_split=120,
				n_estimators=500,
				subsample=0.8,
				warm_start = False,
				random_state = 10

Train data score	0.8496	0.8677	0.8712	0.8892
Test data score	0.8519	0.8546	0.8612	0.8680
CV - Score	0.8495	0.8552	0.8615	0.8696

Table 5. GridSearchCV hyperparameter tuning result

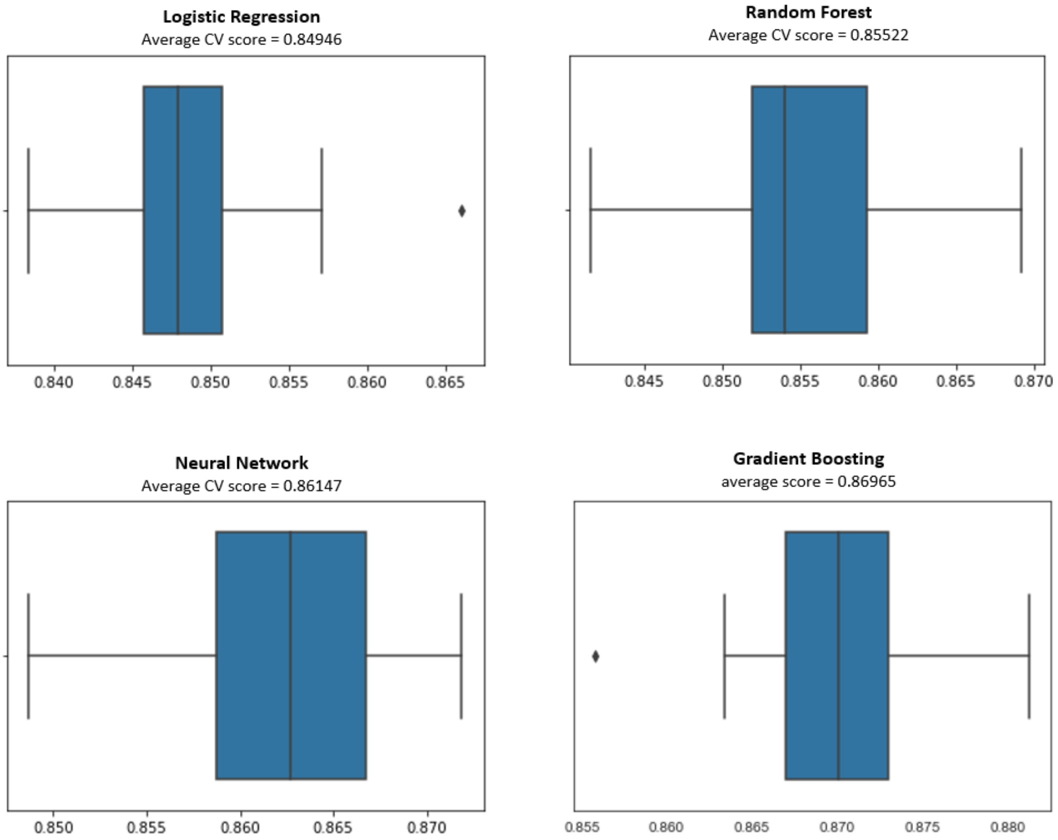


Fig.11 Box plot of cross-validation scores of each model relative to the hyperparameter tunning.

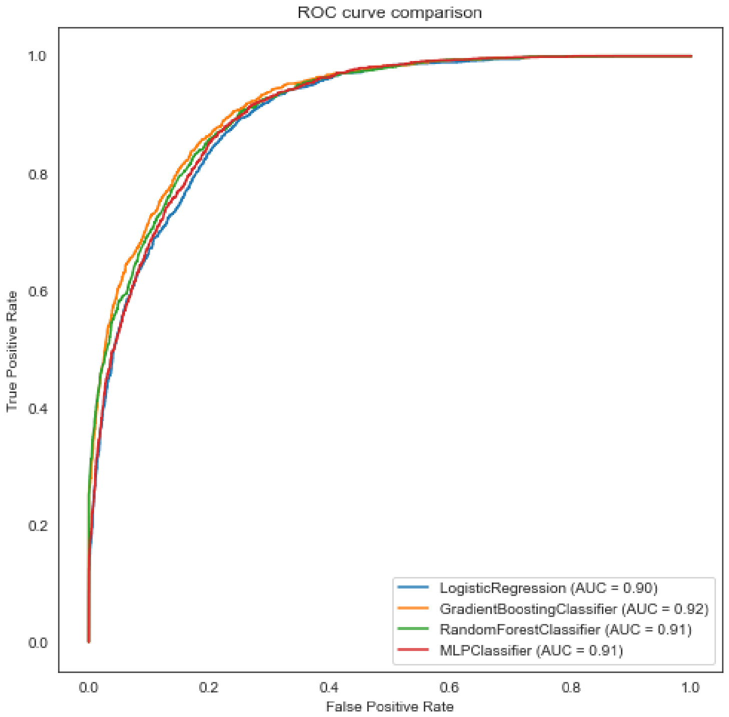


Fig. 12. ROC curves for the models: Logistic Regression, Neural Network, Random forest, Gradient boosting classifier after hyperparameter tuning

The box plots and ROC curves revealed that Gradient Boosting is always slightly better than the remaining models. All classifiers were applied after normalizing all the features of the data set and it can be concluded that they are not overfitting models given the training accuracy don't exceed the validation accuracy beyond all limits. The confusion matrices of all models given are shown in Table 6.

			Predict							
			Logistic Regression		Neural Network		Random Forest		Gradient Boosting	
			0	1	0	1	0	1	0	1
Actual	Train data	0	11181	781	11222	740	11496	466	11368	594
		1	1578	2140	1335	2383	1553	2165	1143	2575
	Validation data	0	4815	312	4804	323	4915	212	4834	293
		1	683	910	654	939	721	872	594	999

Table 6. Confusion matrix of each chosen model

All three models - Random Forest, GBC and Logistic Regression - when used as the algorithm to run the predictive model to classify whether the income above average or not, resulted in ‘good fit’ models, where the training accuracy exceeds the testing accuracy within allowed limits.

Stacking

Despite Gradient Boosting outperformed the other 3 models, the average accuracies of them showed not too much difference. To identify whether it is possible for ensemble or stacking these models, it was decided to calculate the correlation of the prediction of each model.

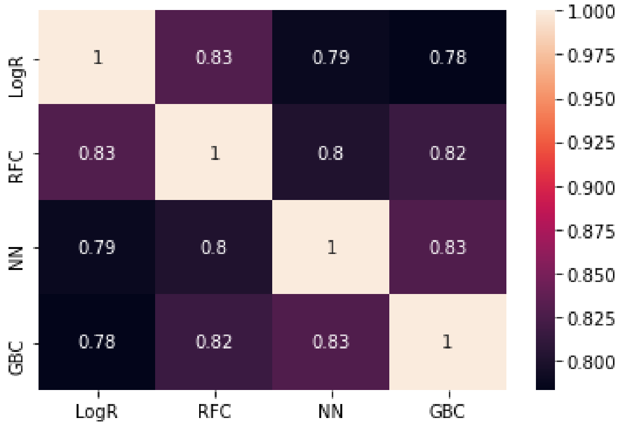


Fig. 13. – Correlation heatmap among the prediction result of each model.

There is a possibility that the prediction combination of these models would generate a better result as they are highly correlated but not absolutely. In the next step, two ensemble methods, Voting Classifier and Stacking Classifier from scikit-learn, would be applied. Despite the previous result showed some relevance, the result of voting ensemble method indicated a worse score, while the stacking method showed a similar one in comparison with the Gradient Boosting, although, its performance was more stable and consistent.

In order to decide the best combination of stacking models, it was tried to run all the possible combinations of four models from two-model stacked to four-model stacked. Applying the ensemble model called Super Learner from ml-ensemble [2, 9] where a 10-fold cross-validation was executed for each of the stacking models.

	Logistic regression	Neural network	Random forest	Gradient boosting	10-fold cross-validation score
Stacked	X	X			0.8536
	X		X		0.8613
	X			X	0.8680
		X	X		0.8580
		X		X	0.8680
			X	X	0.8680
	X	X	X		0.8594
	X		X	X	**0.8688**
	X	X		X	0.8680
		X	X	X	0.8685
	X	X	X	X	**0.8688**

Table 7. Stacking models combination scores where X means that model was inserted on the stacking model.

From the result, it is clear that stacked model with the presence of Gradient Boosting again showed consistently better performance. Moreover, from the result, the best stacked model is the Logistic Regression, Random Forest and Gradient Boosting. Thus, these three models were selected for the final stacking algorithm.

After looking at the results, it was decided to implement the Gradient Boosting. The results given showed that the difference between the training data and validation score were not that different from the previous model (Table7).

Model	Average Score	Standard Deviation	Minimum	Maximum	Average training time (in seconds)
Gradient Boosting	**0.8696**	+/-0.007	0.8558	0.8812	**3.556**
Voting Ensemble	**0.8629**	+/-0.008	0.8513	0.8812	17.693
Stacking LR & GB & RF	**0.8687**	+/-0.007	0.8540	0.8844	10.511

Note on abbreviation: SGDC: Stochastic Gradient Descent Classifier; KNN: K-Nearest neighbors; CART: Decision Tree Classifier

Table 8. 10-fold cross-validation results for each model used.

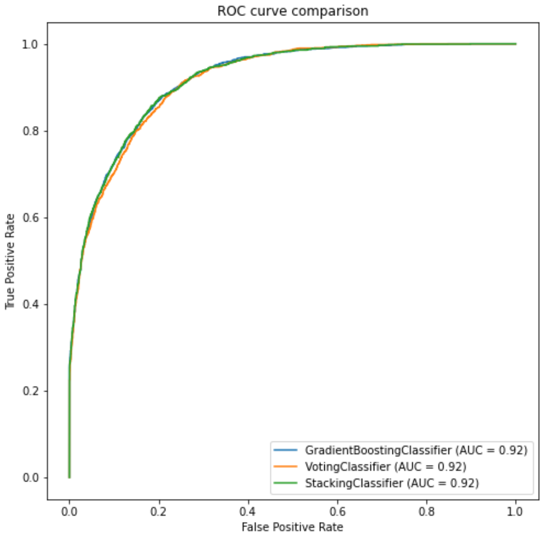
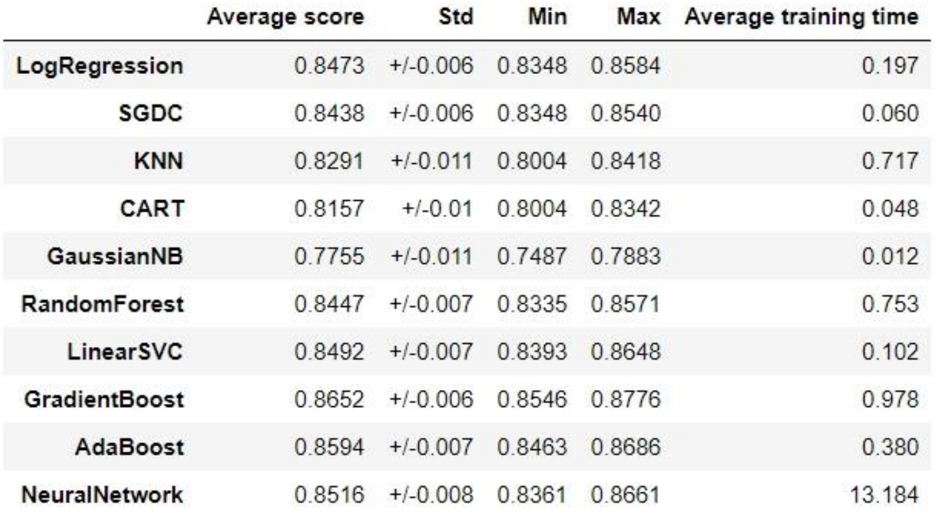


Fig. 14. ROC curves for the final models

Therefore, since both of confusion matrix and the difference between training and validation accuracy were similar, the stacking model was the chosen one to be applied on test data since it was the one which provided the best average F1-score using 10-fold cross-validation of 86.96423% (for Kaggle 30% test data, it got 86.732%)

V. DISCUSSION

For future references, could be used stacking model, but it is important to notice one of the limitations that the stacking model has is relative to the computation demand wise. In other words, this model should not be applied in machines that low specifications, otherwise the training time (as can be shown in the picture below) can be not sustainable in a shorter deadline's projects.



	Average score	Std	Min	Max	Average training time
LogRegression	0.8473	+/-0.006	0.8348	0.8584	0.197
SGDC	0.8438	+/-0.006	0.8348	0.8540	0.060
KNN	0.8291	+/-0.011	0.8004	0.8418	0.717
CART	0.8157	+/-0.01	0.8004	0.8342	0.048
GaussianNB	0.7755	+/-0.011	0.7487	0.7883	0.012
RandomForest	0.8447	+/-0.007	0.8335	0.8571	0.753
LinearSVC	0.8492	+/-0.007	0.8393	0.8648	0.102
GradientBoost	0.8652	+/-0.006	0.8546	0.8776	0.978
AdaBoost	0.8594	+/-0.007	0.8463	0.8686	0.380
NeuralNetwork	0.8516	+/-0.008	0.8361	0.8661	13.184

Table 9. - Average time for each of model in the validation process (in second).

For future approaches could be useful to check the statistical significance of the variables and global correlation of variables with phi_k correlation. The global correlation coefficient defines how well each variable can be modeled in terms of another variable, so this can help in feature engineering. Phi_k library also measures the statistical significance of variables [1].

During the experimental phase, it was noticed that some of the categorical features actually have high cardinality (i.e., unordered categorical predictor variables with a high number of levels). Two different methods of encoding the categorical variables are evaluated. The first is mentioned in the methodology, in which the categorical features' classes were grouped based on their weight of probability in result with the dependent variable. The second method has the same logic with the first one, except that the grouped classes were encoded ordinally by their weight with dependent variable. However, after the model having the best result with Gradient Boosting, in the effort of improving the encoding method by looking for earlier research, one was found that regularized versions of target encoding (i.e., using target predictions based on the feature levels in the training set as a new numerical feature) performed best for all machine learning algorithms (Florian Pargent, 2019) [11]. Thus, an implementation of a regularized target encoding method which is an extension from Generalized Linear Mixed Models (GLMM) called GLMM encoder [12] (Will McGinnis, 2016) were investigated. The results are:

Encoding method	Dataset (rows, cols)	Logistic Regression	SGDC	KNN	CART	Gaussian NB	Random Forest	Linear SVC	Gradient Boost	AdaBoost	Neural Network
Base + One Hot Encoding	**(15680, 59)**	**0.8485**	0.8436	0.8213	0.8142	0.5944	0.843	**0.8497**	0.8661	**0.8596**	0.8438
Category grouping + One Hot Encoding	**(15680, 32)**	0.8473	**0.8438**	**0.8292**	**0.8157**	0.7755	0.8447	0.8492	0.8652	0.8594	0.8516
Category grouping + Ordinal Encoding	**(15680, 13)**	0.8465	0.8363	0.8283	0.8136	0.8332	0.847	0.8467	0.867	0.8578	**0.852**
GLMM encoding	**(15680, 13)**	0.8467	0.8385	0.8291	0.8129	**0.8374**	**0.8492**	0.8481	**0.8671**	0.8591	0.8515

Note on abbreviation: SGDC: Stochastic Gradient Descent Classifier; KNN: K-Nearest neighbors; CART: Decision Tree Classifier

Table 10. Average score cross-validation (k-fold 10 splits)

Encoding method	Dataset (rows, cols)	Logistic Regression	SGDC	KNN	CART	Gaussian NB	Random Forest	Linear SVC	Gradient Boost	AdaBoost	Neural Network
Base + One Hot Encoding	**(15680, 59)**	0.501	0.093	1.494	0.067	0.021	0.81	0.147	1.398	0.556	15.785
Category grouping + One Hot Encoding	**(15680, 32)**	0.197	0.060	0.717	0.048	0.012	**0.753**	**0.102**	0.978	0.380	13.184
Category grouping + Ordinal Encoding	**(15680, 13)**	0.066	**0.037**	**0.25**	0.032	0.008	0.879	0.149	0.882	0.332	10.979
GLMM encoding	**(15680, 13)**	**0.062**	0.04	0.275	**0.038**	**0.008**	0.784	0.124	**0.769**	**0.277**	**7.444**

Table 11. Average training time (In seconds)

The result of regularized target encoding clearly outperformed other encoding methods regarding to both average cross-validation score and training time. However, since target encoders tends to overfit as they use the target variable to encode and it was only tested this result using training dataset, these results might not be similar on the test dataset (Kaggle submission showed considerably lower score then training set). Thus, it was decided not to use this method in our final model. Nonetheless, for future improvement, the overfitting problem can be addressed using cross-fold target encoding to prevent as much as possible data leakage.

Predicting a class label in the imbalanced dataset (the high difference between the positive and negative values) in this study gives us an opportunity to improve the performance of a classifier that predicts probabilities by tuning the threshold used to map probabilities to class labels. [13] The default threshold for interpreting probabilities to class labels is 0.5, and tuning this hyperparameter is called threshold moving. The tune of the optimal threshold when converting probabilities to crisp class labels for imbalanced classification applied to the GB model gave the result in test-data of 87,0623%. The stochastic nature of the algorithm didn't let us use this result in the current work, but it can be useful for further approach with cross-validation for threshold moving.

VI. CONCLUSION

This study provides a comparison of performance between different classification methods in classifying using Newland dataset. The classification accuracy of all methods is compared to each other. Prior to performance comparison, several preprocessing techniques such as data cleaning, feature engineering, feature selection, sampling, and parameter tuning were first conducted. After obtaining optimal values of each classifier, a series of experiments were carried out using 10-fold cross validation.

The experimental results show that the best classifier was the Gradient Boosting, considering, as the most important factor in deciding the best predictive model, the average F1-score (0.86964). This predictive model based on the GB promises to play an important role in the Newland financial stability and identify factors that could be improved for increasing citizen's income. This research article showcase that supervised machine-learning predictive models can be used to develop intelligent systems by the government in their quest to offer to ensure financial stability and the possibility of implementing a long-term strategy of the state through more accurate forecasting of incoming taxes from the population.

VII. REFERENCES

[1] Baak , M., Koopman, R., Snoek, H., Klousa, S., A new correlation coefficient between categorical, ordinal and interval variables with Pearson characteristics. March 12, 2019
https://arxiv.org/pdf/1811.11440.pdf

[2] J. Ferré, Regression Diagnostics in Comprehensive Chemometrics, 2009

https://www.sciencedirect.com/science/article/pii/B9780444527011000764?via%3Dihub

[3] SongHao Wu (2019, May) Multicollinearity in Regression
https://towardsdatascience.com/multi-collinearity-in-regression-fe7a2c1467ea

[4] About Feature Scaling and Normalization. Sebastian Raschka'sWebsite. July 11, 2014
https://sebastianraschka.com/Articles/2014_about_feature_scaling.html

[5] Supervised Learning in SkLearn.
https://scikit-learn.org/stable/supervised_learning.html

[6] What is Logistic Regression?
https://www.statisticssolutions.com/what-is-logistic-regression/

[7] Siddharth Misra, Hao Li, in Machine Learning for Subsurface Characterization (2020)
https://www.sciencedirect.com/topics/engineering/random-forest

[8] Mohtadi Ben Fraj (2017, December), In Depth: Parameter tuning for Gradient Boosting
https://maviator.github.io/2017/12/24/InDepth-Gradient-Boosting/

[9] Rocca, Joseph (2019, April), Ensemble methods: bagging, boosting and stacking
https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

[10] Brownlee, J. (2019, August 06). How to use Learning Curves to Diagnose Machine Learning Model Performance.
https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

[11] Pargent, F., Bischl, B. and Thomas, J., 2019. A Benchmark Experiment on How to Encode Categorical Features in Predictive Modeling.
https://files.de1.osf.io/v1/resources/6fstx/providers/osfstorage/5c97e0b374462c0018ae66a8?action=download&direct&version=1

[12] McGinnis, W.D., Siu, C., Andre, S. and Huang, H., 2018. Category encoders: a scikit-learn-contrib package of transformers for encoding categorical data. Journal of Open-Source Software, 3(21), p.501.
https://contrib.scikit-learn.org/category_encoders/glmm.html

[13] Jason Brownlee on February 10, 2020 in Imbalanced Classification A Gentle Introduction to Threshold-Moving for Imbalanced Classification
https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/