Project Report for

Machine Learning Group Project

Master's in Data Science and Advanced Analytics at NOVA IMS, Lisbon

Group Information

Group:
MAA_202021_50

Group members:
Md. Shawkatul Islam Aziz
Md. Tahir Hossain
Philipp Metzger
Ali Sabbir
Henrique Vaz

Link to GitHub repository:
https://github.com/ph1001/Group-Project-Machine-Learning-Group-50

Abstract

This project's goal is to implement machine learning methods and algorithms for classifying individuals as having an income below or above average. For this, with a training dataset that was provided, multiple methods are implemented and tested.

In order to complete the task, an exploratory analysis of the training dataset is conducted, several transformations are applied to certain variables, the existence of missing values is assessed, the discriminatory power of categorical features is analysed, the categorical variables are encoded, an outlier detection is performed, the features are scaled, and an appropriate subset of the resulting features is selected for further processing. Then, several machine learning algorithms are tested for their fitness for this particular application. This is done by first finding optimal or near optimal parameter values by using Grid Search and then comparing the algorithms using these best parameters to each other. In the end, a small number of algorithms is selected and combined in an ensemble classifier.

By using several Feature Selection techniques, twenty features are selected. On the basis of this data, optimal or near optimal parameters for the models assessed are found and these models are compared to each other. The final selection of models to combine in an ensemble classifier are a Gradient Boosting classifier, an AdaBoost classifier and a Random Forest classifier.

The model we generated is an ensemble of three classifiers: A Gradient Boosting classifier, an AdaBoost classifier and a Random Forest classifier. Those three models were stacked using an instance of the type 'StackingClassifierCV'. The result obtained in the process are described in detail in the results section of this report.

I. Introduction

The goal of this project is to implement Machine Learning algorithms that given the suitable input data are capable of predicting if an individual has an income lower or higher than the average income in a group of citizens.

For this, a dataset of 22400 observations serves as training data. This dataset includes amongst others the variables 'Birthday', 'Marital Status' and 'Education Level' and the binary target variable 'Income'. A full list of the variables contained in the dataset will be presented in Chapter III. The input data and target vector are used to train several predictive models of which one is chosen in the end to be the best suited model for the task.

The project is related to the Kaggle competition 'Newland' in which several groups of students of the course Data Science and Advanced Analytics at the Lisbon based university NOVA IMS compete in a contest that is embedded in the fictitious scenario of the colonisation of a newly discovered habitable planet. In this Kaggle competition, each group uploads a vector of predictions computed with their best model, based on the input data of a test dataset provided in the project materials. The vector of predictions serves as the quality measure to assess which group designed and implemented the best predictive model.

II. Background

In this chapter, the theoretical background of the techniques or algorithms who have not been explored during the practical classes but are applied in this project are explained.

Support Vector Classification:
"Support vector machine" (SVM) belongs to the supervised machine learning algorithms. It can be used for regression and classification. The latter is the case in this project. Thus, the algorithm used here is called support vector classifier. The goal of support vector machines is to define a hyperplane in the n-dimensional space containing the training data points[1], that separates the different classes in the target variable[2]. In case of a linearly separable dataset, meaning that a hyperplane can be defined such that if all observations from one class are on one side and all observations from the other class are on the other side of the hyperplane, the hyperplane simply represents the border that separates the two classes and that has the maximum distance to each and every point in the dataset (measured individually), which means that only the distances to the points on each side that are closest to the hyperplane are relevant. These closest points are referred to as the "support vectors" since they support the position of the hyperplane. Because the area around the hyperplane could be imagined as a channel or street separating the dataset, this is also called the "widest street approach". In case of a non-linearly separable dataset, a penalty can be added during the calculation of the optimal solution, such that points that are "on the wrong side" of the hyperplane are penalized in a way that is adequate for the application. This is called the "Soft Margin" approach (see chapter 7.1.1 in

[1] Or, if the "Kernel Trick" is used, a higher dimensional space of which said n-dimensional space is a subspace.
[2] Typically, two classes as in this application, but adaptations for multiclass problems also exist (See chapter 7.1.3 in [3]).

[3]). In applications, where the dataset contains patterns where a hyperplane is not able to create a satisfactory division, such as when a ring of points of one class surrounds a cluster of points of the other class, the n-dimensional space of the datapoints can be mapped to a higher-dimensional space in order to achieve a better separability. This is called the "Kernel Trick" (see chapter 6 in [3]).

Random Oversampling:
When one group of a categorical or binary target variable has more observations than the others (in the categorical case) or the other (in the binary case), it can be sensible to use random oversampling to harmonise the number of observations associated to each category of the variable. In random oversampling this is achieved by duplicating observations that are associated to the category or the categories that originally have less observations associated to them. The observations to duplicate are selected randomly and with repetition. The goal of random oversampling can be that after the process, all categories have the same number of observations associated with them. Or in the binary case, it is also be an option to define a factor, by which the numbers of observations of the two classes differ from each other such that n_small = a * n_large, $\in$ IR, a $\in$ (0,1], where n_small is the number of observations associated with the class that originally had less observations and n_large is the number of observations associated with the other class. Random oversampling is described in Chapter 5 of [1].

III. Methodology

III.1 Materials and Software

The materials used in this project are the training dataset and the test dataset provided. The original training dataset consists of 22400 observations and 15 variables (including the target variable). These variables are shown in Table 1, which is originated in the PDF file 'Project Presentation.pdf' which serves for defining the fictitious background and the goal of the project and the kaggle competition.

Variable	Description
Citizen_ID	Unique identifier of the citizen
Name	Name of the citizen (First name and surname)
Birthday	The date of Birth
Native Continent	Home continent of the citizen on planet Earth
Marital Status	The marital status of the citizen
Lives with	The household environment of the citizen
Base Area	The neighborhood of the citizen in Newland
Education Level	The education level of the citizen
Years of Education	The number of years of education of the citizen
Employment Sector	The employment sector of the citizen
Role	The job role of the citizen
Working Hours per week	The number of working hours per week of the citizen
Money Received	The money payed to the elements of Group B
Ticket Price	The money received by the elements of Group C
Income	The dependent variable (Where 1 is Income higher than the average and 0 Income Lower or equal to the average)

Table 1: The original dataset's variables adopted from "Project Presentation.pdf"

The software that is used in order to complete the project is Python and more precisely Anaconda and Jupyter Notebook. In the latter, the code for this project is created.

In the following part, the steps conducted in our Jupyter notebook are described.

III.2 Loading of Data, Data Exploration and Data Pre-processing

First, all packages and libraries that are used are imported. These include, amongst others, libraries from the standard Python library such as 'os' and packages from 'sklearn' which are used for the predictive models but also the library 'xgboost' which we used for feature selection, but not for classification.

In the second step the data described above is loaded into main memory and a first data exploration is conducted through which it becomes clear, that the training dataset contains more observations with the target variable 'Income' equal to 0 than observations with the target variable 'Income' equal to 1, meaning that in the training dataset, there are more observations that are classified as having an income lower than the average.[3] This serves as the motivation for applying Random Oversampling in our application.

In the following step the variable 'Birthday' is transformed to obtain the age of each individual. As the unit for this newly created variable *days* is chosen, and the old variable is replaced by the new one.

In the next step, the existence of cells containing empty strings, spaces or NaN values[4] is assessed.

In a further exploratory analysis, the discriminatory power of the categorical features in the training dataset is assessed by plotting bar charts for each category in which the affiliation the either target value is represented in either grey or green.

The following step's purpose is to one-hot encode most of these categorical features in order to make it possible to process them in the algorithms used later on. Only the feature 'Education Level' is not one hot encoded. For this one, a different encoding method is designed and applied, which assigns a numerical rank to each original value of this variable.

Next, outlier detection is performed. For this, the minima and maxima of features that could potentially have outliers are computed and an assessment is made on whether or not these values are realistic. The result of this assessment is described in subchapter IV.2.

The following step is the procedure of feature scaling. For this, a standard scaler is used, which standardises every feature by removing its mean and scaling it to unit variance [2].

III.3 Feature Selection

III.3.i Correlation Analysis

Next, as a first feature selection step, correlations between the metric variables (and the target

[3] Income = 1: 76.29 %, Income = 1: 23.71 %

[4] NaN stands for Not a Number, meaning that the respective value is non-existent.

variable 'Income') as well as between all variables (and the target variable 'Income') are assessed in order to decide, which features are contributing only redundant information to the dataset and which features can be discarded due to a very low correlation with the target variable. For this purpose, the Pearson and Spearman correlations between all these features are computed and visualized in correlation matrix plots. Two distinct masks are used for both pairs:

1. In order to find redundancies, only the correlations higher than 0.3 are displayed. Based on this, of feature pairs with very high correlations, one is discarded.

2. Secondly, to identify the features that are most correlated with the target variable, only the correlations lower than 0.08 are displayed. Based on this, most features with a lower correlation to the target than 0.08 are removed.

for now, it is relevant to know that some features are discarded and a list of features 'features_to_keep_1' is defined, which contains the features that are kept after the correlation analysis. The variables of the resulting training dataset are presented in Table 2.

Variable	Description
Age_days_rel_to_2020	Age of the citizen relative to the most recent day the code has been run
Marital_Status_Married	1: Citizen is married, 0 otherwise
Marital_Status_Single	1: Citizen is single, 0 otherwise
Marital_Status_Divorced	1: Citizen is divorced, 0 otherwise
Lives_with_Children	1: Citizen lives with children, 0 otherwise
Lives_with_Husband	1: Citizen lives with husband, 0 otherwise
Lives_with_Alone	1: Citizen lives alone, 0 otherwise
Lives_with_Other Family	1: Citizen lives with other family, 0 otherwise
Role_Management	1: Citizen's role is in management, 0 otherwise
Role_Other services	1: Citizen's role is in "other services", 0 otherwise
Role_?	1: Citizen's role is unknown, 0 otherwise
Role_Administratives	1: Citizen's role is of administrative nature, 0 otherwise
Role_Cleaners & Handlers	1: Citizen's role is described as "Cleaners & Handlers", 0 otherwise
Role_Professor	1: Citizen is a professor, 0 otherwise
Working Hours per week	The number of working hours per week of the citizen
Money Received	The money payed to the elements of Group B
Ticket Price	The money received by the elements of Group C
Education_Level_Classified	The education level of the citizen in respect to the classification implemented in this project
Employment_Sector_Self-Employed (Company)	1: Self-employed (Company), 0 otherwise
Employment_Sector_Private Sector - Services	1: Employed in private sector, 0 otherwise
Income	The dependent variable (Where 1 is Income higher than the average and 0 Income Lower or equal to the average)

Table 2: 'features_to_keep_1' - The training dataset's variables after the pre-processing steps

III.3.ii Application of further Feature Selection methods

In a further feature selection step, RFE[5], a Ridge classifier and XGBoost are applied in order to identify the most important features. For RFE, a Gradient Boosting Classifier was used.

The result of this further limitation of the feature space is saved in the list 'features_to_keep_2'.

[5] RFE, short for Recursive Feature Elimination

III.3.iii Verification of feature selection for MLP and AdaBoost classifier

This part contains an analysis whose goal it is to verify whether or not the feature selection defined in the list 'features_to_keep_2' is an adequate choice for applying it in combination with an MLP[6] and with an AdaBoost[7] classifier. For this purpose, two steps are done:

1. The two algorithms were applied with four different subsets of the training dataset. As a quality measuring technique, k-fold cross validation with k = 10 was used.

2. Starting from 'features_to_keep_2', features are randomly removed, one feature at a time and the validation score is visualised in a multiline plot.

In the next section, the process of selecting the appropriate models and their parameters is described.

III.4 Model Selection and Parameter Tuning

The purpose of the following part of the code is to assess the fitness of different models for dealing with the application at hand. Grid Search[8] and k-fold cross validation with k = 10 are used to find optimal or near optimal combinations of parameter values for each algorithm. After this, the best models are selected based on their accuracy. This selection is also conducted using k-fold cross validation with k = 10. The models that are considered in this part are MLP classifier, AdaBoost, Gradient Boosting Classification[9], Random Forest Classification[10], Support Vector Classification[11] and K-nearest Neighbours Classification[12].

IV. Results

IV.1 Results in Data Exploration and Data Pre-processing

IV.1.i Assessment of discriminatory power

For the categorical variable 'Base Area', most cases are in one category and very few observations are in the other categories. Due to the small sample sizes in the low cardinality categories, it is probable that the variation between the categories is only of random nature and doesn't represent any type of pattern. The visualization for this analysis is shown in Figure 1.

---

[6] Multilayer Perceptron (for a general explanation see chapter 5 in [3]; for the documentation of the implementation used for this project see [4])
[7] AdaBoost, short for Adaptive Boosting (for a general explanation see chapter 14.3. in [3]; for the documentation of the implementation used for this project see [5])
[8] For the documentation of the implementation used in this project see [6]
[9] For a general explanation see chapter 12 of [11]; for the documentation of the implementation used in this project see [7]
[10] For a general explanation see [12]; for the documentation of the implementation used in this project see [8]
[11] Explanation of the method in chapter II of this report; for the documentation of the implementation used in this project see [9]
[12] For a general explanation see chapter 2.5.2 of [3]; for the documentation of the implementation used in this project see [10]

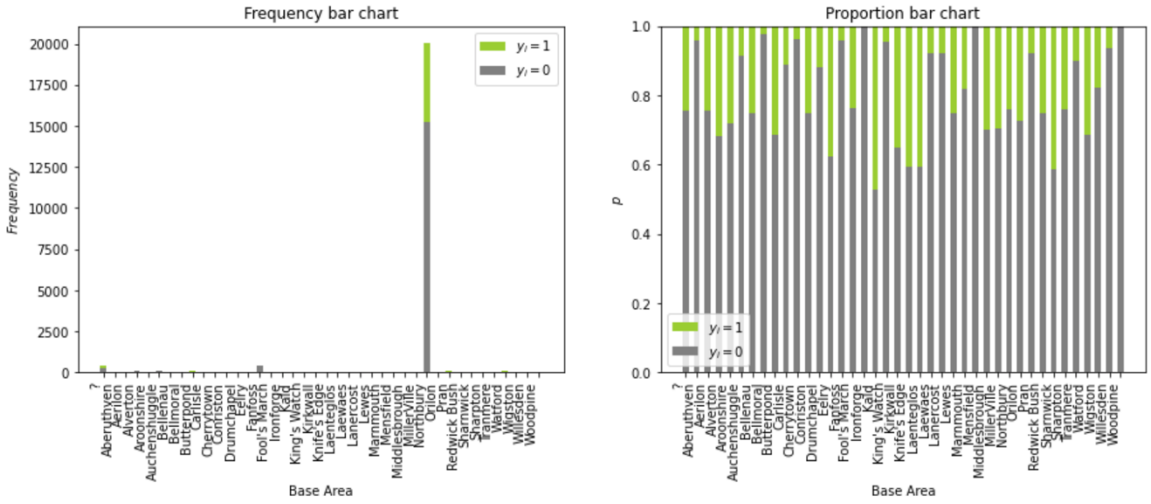

Figure 1: Discriminatory power of 'Base Area'

Following the same logic, the categorical feature 'Native Continent' was also discarded. Due to the low variation in the categories, this feature is considered as not relevant for the classification. The visualization for this analysis is shown in Figure 2.

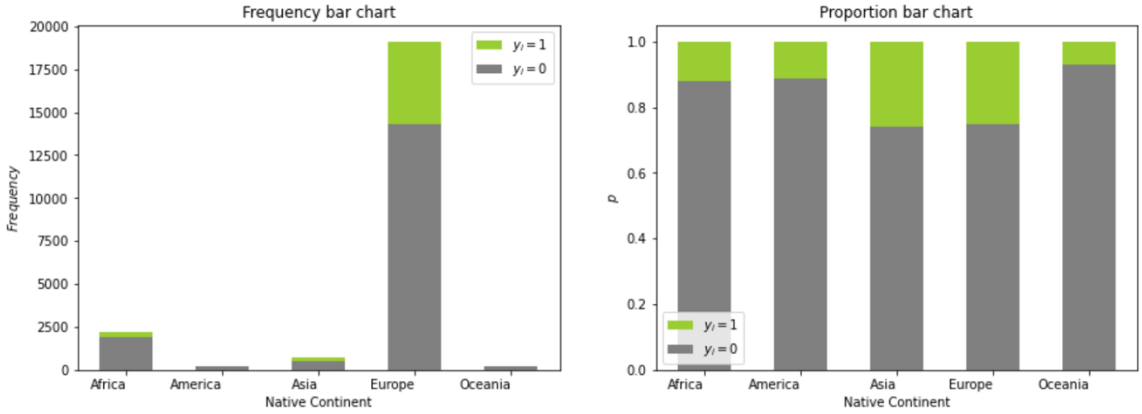

Figure 2: Discriminatory power of 'Native Continent'

The remaining categorical features are kept and one-hot encoded. They are further processed in the next steps.

IV.2 Results of the Detection of Outliers

As a first step in the process of outlier detection, the features that could potentially have outliers are identified. These are: 'Age_days_rel_to_2020' (which is a transformation of 'birthday'), 'Working hours per week', 'Money Received', 'Ticket Price' and 'Years of Education'. The detection of outliers is conducted by computing the minima and maxima of each of these features and assessing, whether or not theses values are realistic or not:

1. 'Age_days_rel_to_2020': For the assessment of this feature, the variable that it is computed from ('birthday) is looked at: The earliest birthday is on 1. Jan. 1958 and the latest one is on 1. Jan. 2031. Since all values of this variable range in a reasonable interval, no observations are removed.

2. ‘Working hours per week’: Here, the maximum is 99 and the minimum is 1. A quick assessment of the likelihood of these values is conducted: Regarding the maximum of 99, it can be assessed that this person would work 14 hours per day on average if he or she worked every day. Since it is logically possible to work this much, we conclude that all values of this variable are reasonable, and no observations have to be removed.

3. ‘Money Received’: The minimum of this variable is 0 and the maximum is 122999. This means that there are no negative value, which would be inconsistent with the variable’s name and that there are no extraordinary high values either.

4. ‘Ticket Price’: With a maximum of 5358 and a miminum of 0, the same as for ‘Money Received’ is valid for ‘Ticket Price’.

5. ‘Years of Education’: With a maximum of 21 and a minimum of 2, all value are not far away from the mean which is 13.17. Concluding from this, all values seem to be reasonable and no observations need to be removed.

IV.3 Results of the selection of features

IV.3.i Correlation Analysis

This part describes the result of the correlation analysis conducted for the purpose of further feature selection. The training dataset at this point is stored in the variable ‘train_data_scaled’. The features stored in this variable plus the target variable income make up the basis for the correlation analysis. The decisions made and results of this analysis are as follows:

No high correlations are detected between the original metric features. But the newly created feature ‘Education_Level_Classified’ is highly correlated with ‘Years of Education’ which was expected. Both of these two features have roughly the same correlation to the target variable ‘Income’. The decision is made to keep ‘Education_Level_Classified’ which is identified as the more informative feature overall and to drop ‘Years of Education’.

Next, the correlations between all the features mentioned above is assessed. As a result of this, the following decisions are made:

‘Role_?’ and ‘Employement_Sector_?’ have a correlation of 1. Furthermore, out of the two, ‘Role_?’ has a higher correlation to the target variable. Also, the correlation of ‘Employement_Sector_?’ to the target variable is lower than the limit of 0.08 which was described in III.3. Because of this, ‘Employement_Sector_?’ is removed.

The result after discarding most of the features that have a correlation to the target variable smaller than 0.08, a set of features remains. The names of these features are saved in the list ‘features_to_keep_1’, which is visualized in Table 3.

Variable
Age_days_rel_to_2020
Marital_Status_Married
Marital_Status_Single
Marital_Status_Divorced

Lives_with_Children
Lives_with_Husband
Lives_with_Alone
Lives_with_Other Family
Role_Management
Role_Other services
Role_?
Role_Administratives
Role_Cleaners & Handlers
Role_Professor
Working Hours per week
Money Received
Ticket Price
Education_Level_Classified
Employment_Sector_Self-Employed (Company)
Employment_Sector_Private Sector - Services

Table 3: 'features_to_keep_1'

IV.3.ii Results of the application of further Feature Selection methods

As described in III.3.ii, this set of features, further features selection techniques are applied.

Recursive Feature Elimination (RFE):
The result of the RFE is stored in a data frame indicating which features are probable to be relevant ("True") and which are not ("False"). This result is shown in Figure 3.

	Important
Age_days_rel_to_2020	True
Base_Area_Orilon	False
Marital_Status_Married	True
Marital_Status_Single	False
Marital_Status_Divorced	False
Lives_with_Children	True
Lives_with_Husband	True
Lives_with_Alone	True
Lives_with_Other Family	True
Role_Management	True
Role_Other services	True
Role_?	False
Role_Administratives	True
Role_Cleaners & Handlers	True
Role_Professor	True
Working Hours per week	True
Money Received	True
Ticket Price	True
Education_Level_Classified	True
Employment_Sector_Self-Employed (Company)	True
Employment_Sector_Private Sector - Services	True

Figure 3: Resulting DataFrame from RFE

Following the RFE, feature importance was assessed using a Ridge Classifier. The output of this process is a chart comparing all features' importances which is shown in Figure 4.

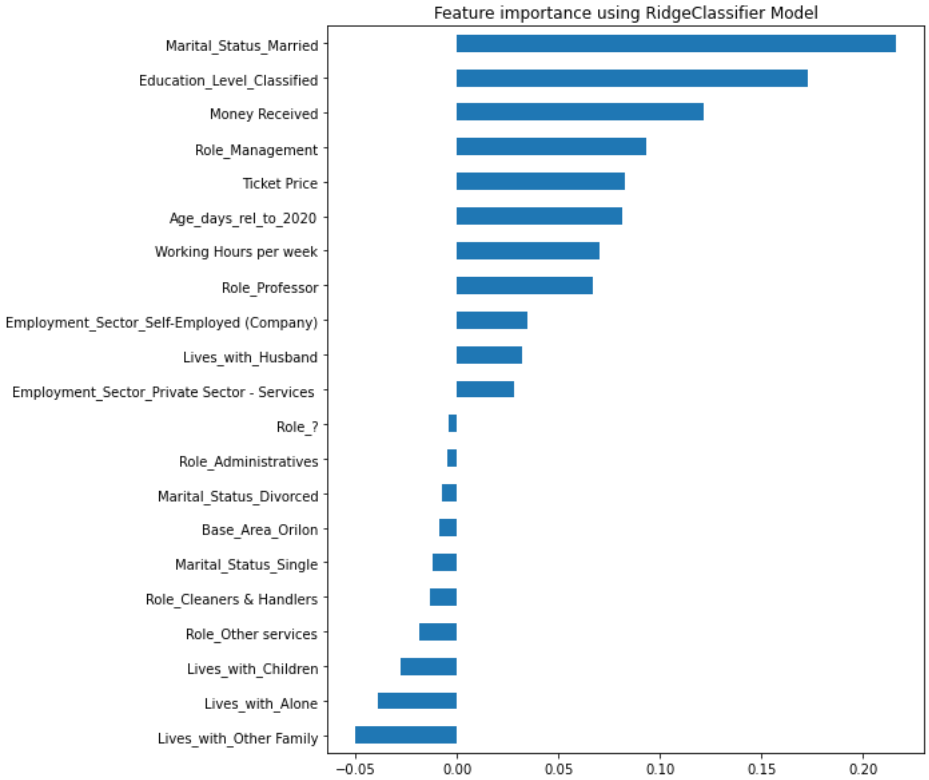

Figure 4: Resulting chart from Ridge Classifier

From Figure 4 it can be seen that all metric features are considered important as well as 'Marital_Status_Married', 'Education_Level_Classified', 'Role_Management', 'Working Hours per week', 'Role_Professor' and 'Lives_with_Other_Family'.

The next technique used, for feature selection, is an XGBoost Classifier. The computed feature importances from this step are shown in Figure 5.

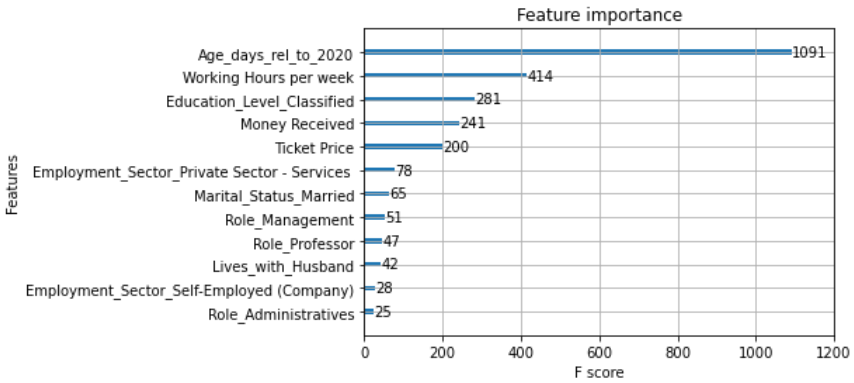

Figure 5: Resulting chart from XGBoost Classifier

According to this technique, what is most important to point out is the high importance of metric features as well as ‘Education_Level_classified’. Apart from those, no features seem to add much value.

After applying these three techniques, the goal is to come up with a new list of features that provides us with a balanced result. Considering this, only ‘Role_?’ is eliminated, since this feature is the only one that is not identified as important by any of the techniques. All the other features have relative importance and can bring versatility to our model. As a result of the steps described a list of the names of the remaining features is created, which is stored in the variable ‘features_to_keep_2’, which is shown inTable 4: ‘features_to_keep_2’.

Variable
Age_days_rel_to_2020
Marital_Status_Married
Marital_Status_Single
Marital_Status_Divorced
Lives_with_Children
Lives_with_Husband
Lives_with_Alone
Lives_with_Other Family
Role_Management
Role_Other services
Role_?
Role_Administratives
Role_Cleaners & Handlers
Role_Professor
Working Hours per week
Money Received
Ticket Price
Education_Level_Classified
Employment_Sector_Self-Employed (Company)
Employment_Sector_Private Sector - Services

Table 4: ‘features_to_keep_2’

IV.3.ii Results for verification of feature selection for MLP and AdaBoost classifier

For the step described in III.3.iii point 1, the first subset of features for which the models’ performance is evaluated is defined by the list ‘features_certainly_to_keep’ which consists of the features ‘Age_days_rel_to_2020’, ‘Working Hours per week’, ‘Money Received’, ‘Ticket Price’, ‘Education_Level_Classified’. These features are the five metric features but with the substitution of ‘Years of Education’ by 'Education_Level_Classified’. These features are regarded to be the most essential features of the dataset and are supposed to serve as a base case in this step of the analysis. The result for this feature set for the MLP and the AdaBoost classifiers are shown in Table 5.

	Training score	Validation score	Iterations
MLP classifier	0.83	0.828	71.9
AdaBoost classifier	0.846	0.844	-

Table 5: Results for base case ('features_certainly_to_keep')

The seconds subset of features is 'features_to_keep_2'. For this subset the results are shown in Table 6.

	Training score	Validation score	Iterations
MLP classifier	0.859	0.853	71.9
AdaBoost classifier	0.867	0.865	-

Table 6: Results for 'features_to_keep_2'

The third subset of features is 'features_to_keep_1'. For this subset the results are shown in Table 7.

	Training score	Validation score	Iterations
MLP classifier	0.862	0.855	71.9
AdaBoost classifier	0.867	0.864	-

Table 7: Results for 'features_to_keep_1'

The last subset is the whole set of pre-processed training data which is stored in the variable 'train_data_scaled'. For this, the results are shown in Table 8.

	Training score	Validation score	Iterations
MLP classifier	0.874	0.851	71.9
AdaBoost classifier	0.872	0.867	-

Table 8: Results for all data ('train_data_scaled')

Looking at these results it can be stated that for both models used, the features defined by 'features_to_keep_2' are a good choice for both models. The validation scores for 'features_to_keep_2' and 'features_to_keep_1' are approximately the same whilst the difference in the training and validation scores is lower for 'features_to_keep_2' than for 'features_to_keep_1'. Also, the scores of the base case are lower than for 'features_to_keep_2', which was expected since the base case has significantly less features. Using the whole data is not favourable, since it leads to overfitting for both models as can be seen from in Table 8.

For the step described in III.3.iii point 2, which is the analysis of the validation score when randomly dropping one feature each iteration, starting with the features from 'features_to_keep_2', yields the result shown in Figure 1 and Figure 7, suggesting that it is probably favourable not remove any more features from 'features_to_keep_2'.

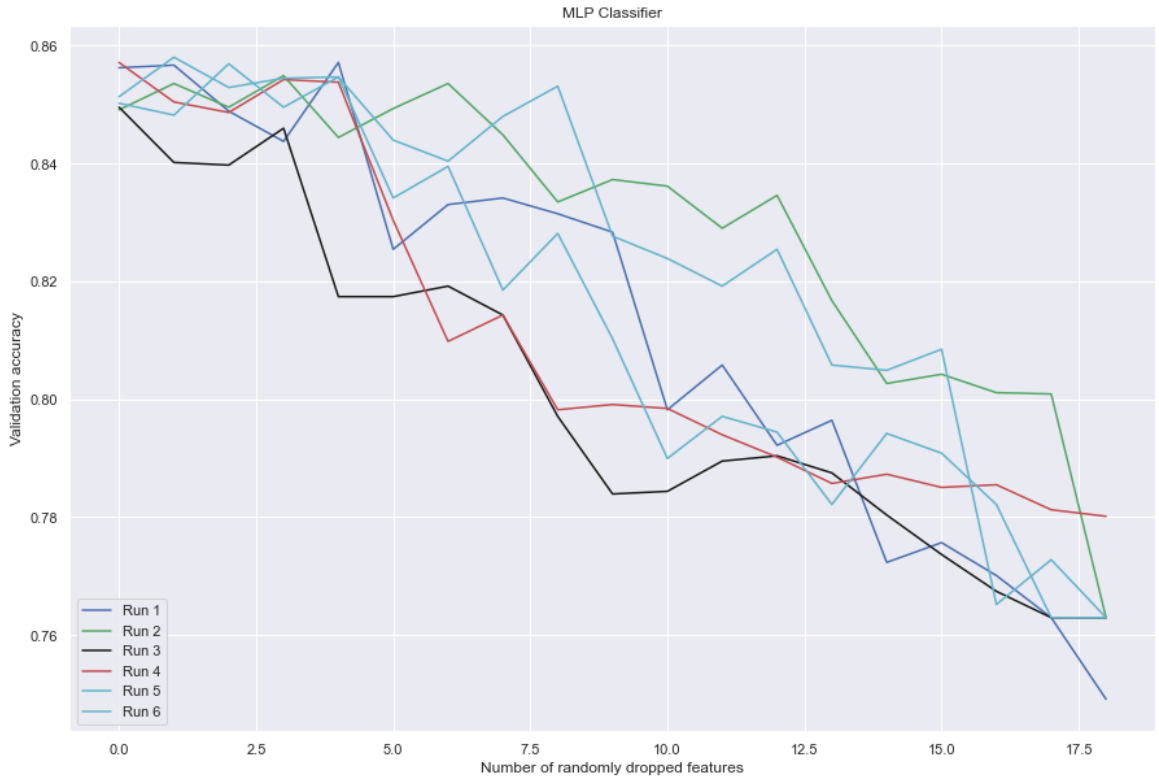

Figure 6: Visualisation of the decrease in validation accuracy for the MLP classifier

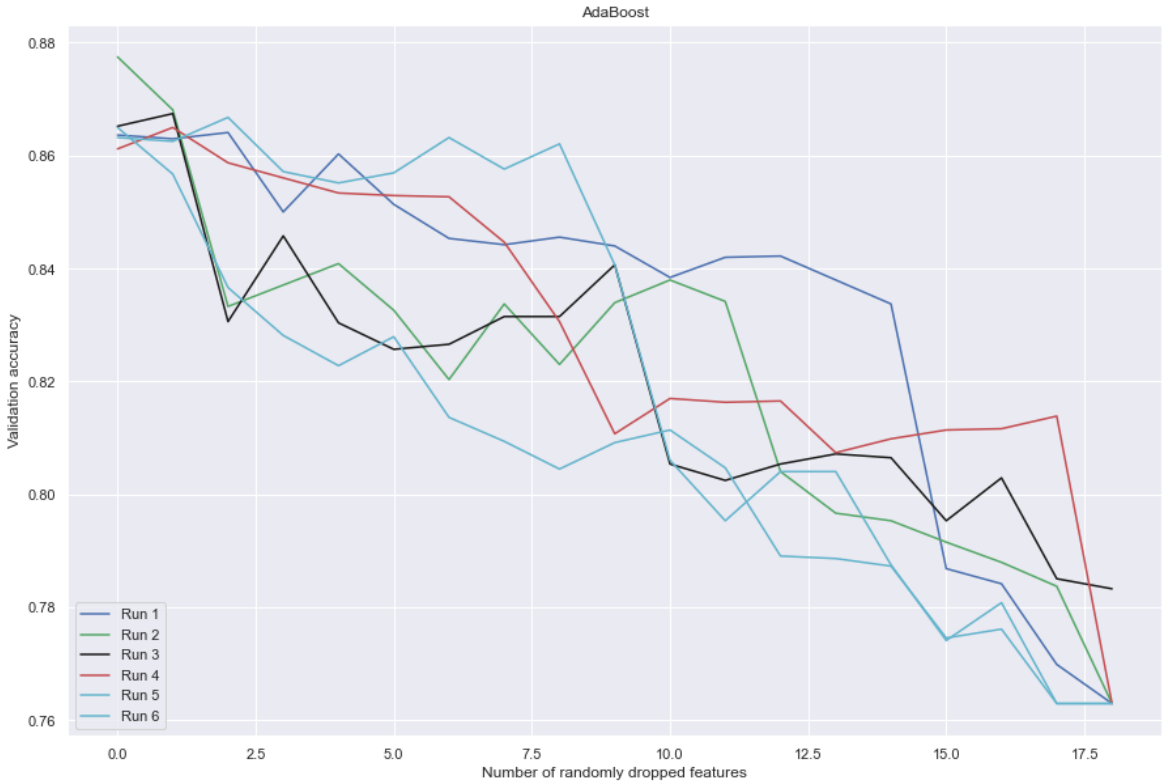

Figure 7: Visualisation of the decrease in validation accuracy for the AdaBoost classifier

IV.6 Results for the finding of parameters of the different models

IV.6.i Results for Grid Search for MLP classifier

The Grid Search conducted in order to find optimal or near optimal parameters for the MLP classifier resulted in the conclusion that the best parameters for this particular application are:

activation = 'tanh',
alpha = 4e-05,
hidden_layer_sizes = (10,),
learning_rate = 'constant',
learning_rate_init = 0.01,
random_state = 42, and
solver = 'adam'.

IV.6.ii Results of Grid Search for AdaBoost

The best parameters found for AdaBoost are:

n_estimators=962,
learning_rate=0.61,
algorithm='SAMME.R', and
random_state=42.

IV.6.ii Results for parameters for Gradient Boosting Classifier

For the Gradient Boosting Classifier, different parameters were manually tested. The first set of parameters is as follows:

{'n_estimators': [50, 75, 100, 200],
'learning_rate': [0.1, 0.2, 0.3],
'max_features': [None, 7],
'max_depth': [3, 4, 5],
'min_samples_leaf': [5, 6, 7],
'min_samples_split': [4, 7, 10],
'subsample': [1.0, 0.9, 0.8],
'max_features': [5, 9, None],
'random_state': [42]}

After testing these parameters and iteratively applying small changes, the result set of parameters for the manual Gradient Boosting Classifier is:

{'learning_rate': 0.1,
'max_depth': 4,
'max_features': None,
'min_samples_leaf': 2,
'min_samples_split': 10,
'n_estimators': 150,

'random_state': 42}

In addition to the manual testing, a Grid Search was conducted which included the following p arameters:

{'n_estimators': [50, 100, 150, 300, 400],
'learning_rate': [0.1, 0.3],
'max_features': [None, 7],
'max_depth': [2, 3, 5],
'min_samples_leaf': [1, 3, 5],
'min_samples_split': [4, 8, 10],
'random_state': [42]}

The optimal or near optimal parameters found in this search are as follows:

{'learning_rate': 0.1,
'max_depth': 3,
'max_features': 7,
'min_samples_leaf': 3,
'min_samples_split': 8,
'n_estimators': 300,
'random_state': 42}

When comparing the models' performances using the parameters from the manual search and the Grid Search, the results presented in Table 9 are obtained:

	Time	Train	Validation	Iterations
Manual	4.018+/-1.0	0.877+/-0.0	0.867+/-0.01	nan+/-nan
Grid	5.418+/-0.21	0.876+/-0.0	0.867+/-0.01	nan+/-nan

Table 9: Comparing the parameters obtained from a manual search and a Grid Search for configuring a Gradient Boosting Classifier

The scores of the models for the validation set are exactly the same, however, the model using the parameters from the manual search seems to require less computation time. Additionally, t he performances of the two models were compared when incorporated in the final ensemble a lgorithm and here, as well, the model obtained from the manual search gave better results. For those two reasons, the model obtained from the manual search is used in the final steps of the project and the other one discarded.

IV.6.iii Results for parameters for a Random Forest Classifier

With the Random Forest classifier, a similar approach to the one for the Gradient Boosting Cl assifier is followed. The set of parameters manually tested is as follows:

{'n_estimators': [50, 100, 150, 200, 250, 300, 350, 400, 450, 500],
'max_features': ['auto', 'sqrt'],
'max_depth': [5, 10, 15, 20, 25, 30, 35, 40, 45, 50, None],
'min_samples_split': [10, 20, 30],
'min_samples_leaf': [6, 10, 14],
'bootstrap': [True, False]}

The optimal or near optimal parameters found in this search are as follows:

{'n_estimators': 250,
'max_features': 'auto'
'max_depth': 30,
'min_samples_split': 10,
'min_samples_leaf': 6,
'bootstrap': False}

Using that configuration, the score obtained is:

	Time	Train	Validation	Iterations
Random Forest classifier	5.179+/-0.89	0.885+/-0.0	0.861+/-0.01	nan+/-nan

Table 10: Training and validation scores for the parameters found through the manual search for parameters for a Random Forest classifier

IV.6.iv Results for parameters for a Support Vector Classification model

After evaluating the random forest, we will now look into the Support Vector Classification (SVC) performance. A Grid Search is used in order to find optimal or near optimal parameter. The set of parameters used for the search are as follows:

{'C': [1, 5, 10],
'gamma': ['auto', 'scale', 0.1, 5],
'kernel': ['rbf', 'linear', 'sigmoid']}

As the result of the Grid Search, the best configuration within that set appears to be:

{'C': 5,
'gamma': scale,
'kernel': rbf}

The corresponding results of this model are shown in Table 11:

	Time	Train	Validation	Iterations
SVC classiffier	3.055+/-0.16	0.858+/-0.0	0.847+/-0.00	nan+/-nan

Table 11: Training and validation scores for the parameters found through the manual search for parameters for a SCV classifier

IV.6.iv Results of a high-level analysis of the models

Next, a high-level analysis of the models' performances is done. Using the best configuration s found in the previous steps, the models are compared in regards of their validation scores. T his is done by using stratified k-fold cross validation with k = 10. The goal of this step is to d ecide which models are the best ones to use for the ensemble classifier built in the next steps. The results of this analysis are shown in Table 12 and Figure 8.

	Avg. train score	Std. dev. train score	Avg. validation score	Std. dev. validation score	Avg. time	Std. dev. time
Multilayer perceptron	0.8595	0.001137	0.8528	0.006613	6.12	1.28
Gradient Boosting Classifier	0.8768	0.000766	0.8671	0.007770	3.79	0.39
Ada Boost Classifier	0.8666	0.001012	0.8648	0.008005	11.40	1.00
Random Forest Classifier	0.8856	0.000937	0.8610	0.008026	5.83	1.32
SVC	0.8594	0.001000	0.8482	0.006426	18.10	0.50
K Neighbors	0.8721	0.001069	0.8379	0.009219	1.02	0.05

Table 12: Comparing the scores of all the different models used in the project

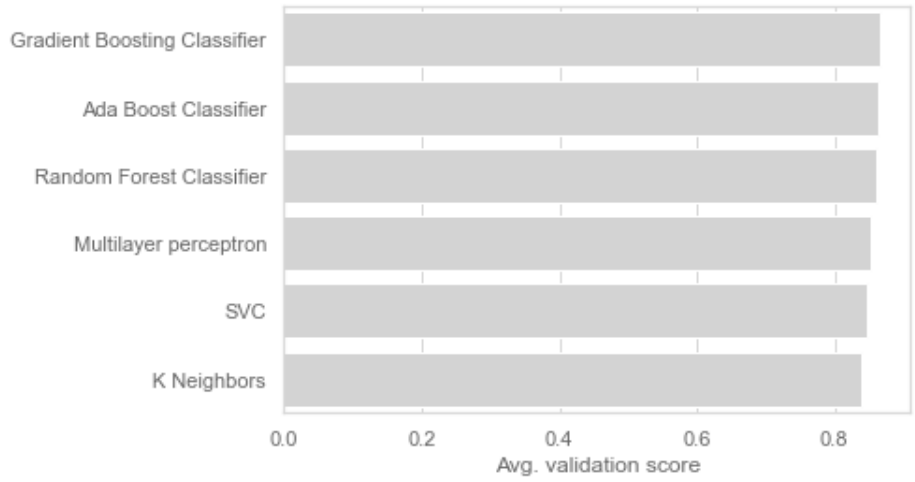

Figure 8: Assessing the scores of the models used in the project

As can be seen in the chart above, there are three models that stand out due to their relatively high validation score. Those are the Gradient Boosting classifier, the Ada Boost classifier and the Random Forest classifier.

In a next step, the F1-scores of different combinations of models are computed and compared . In order to compute the needed values, a for loop is created that iterates through a list contai ning the different combinations of classifiers. The result of this step is shown in Table 13:

Models	**F1 Score training set**	**F1 Score validation set**
GBC, ABC, RF	0.8784598	0.8725446
GBC, ABC	0.8771763	0.8707590
ABC, RF	0.8841518	0.8642857

Table 13: Comparing the results obtained after trying different combinations of classifiers to use in the stacking classifier

These results shown in Table 13 are all referring to the same validation set, which was acquire d using the of the 'train_test_split' function using stratification and a validation set size of 20 % of the whole dataset used. The values obtained are close to one another, although the first c ombination seems to obtain the best generalization.

The result of the steps before is that the final model is chosen to be a stacking classifier using a Gradient Boosting classifier, an AdaBoost classifier and a Random Forest classifier.

Random Oversampling

Random Oversampling, which is a technique described in chapter II, was tested in different settings and with different algorithms. The initial reason for testing it was that the dataset is

highly skewed in favor of the target class 0 ('Income lower than the average')[13]. However, since applying Random Oversampling did not yield better results than not applying it (and sometimes even worse results), the authors decided not to pursue the application of this technique for this project any further.

V. Conclusion

In the following, the most important findings of this report will be presented.

The first finding is that no outliers need to be removed from the training dataset.

Following this, an analysis of the discriminatory power of the categorical features is performed. The goal of this step is to evaluate which features affect the target variable 'Income'. The outcome of this analysis is the removal of two variables: 'Native Continent' and 'Base Area'.

After the performance of the encoding of the categorical values, 85 features are obtained.

Then, correlations between features are computed to evaluate not only relevance but also redundancy between them. The result of this step is a dataset with 20 columns.

The next step is made up by three features selection techniques: RFE, Ridge classifier and XGBoost classifier. The result of this step is a list of 19 features whose names are saved in the variable 'features_to_keep_2'. A further feature selection process taking into consideration the behaviors of an MLP and an AdaBoost classifier, confirms the result from the previous step.

After finding the best features to use, different models are evaluated. Within this model selection phase, different configurations for each model are tested, keeping always the one that yields better scores. The result of this process is displayed in Table 12 and Table 13. Evaluating the tables, it becomes clear that the best individual models are Gradient Boosting classifier (GBC), AdaBoost classifier (ABC) and Random Forest classifier (RF).

Next, the obtained models are combined in an ensemble classifier of the type 'StackingClassifiersCV'.

For this, the best combination of classifiers is identified as Gradient Boosting classifier (GBC), AdaBoost classifier (ABC) and Random Forest classifier (RF) which yields an F1-score on the validation set of 0.8725.

Lastly, the model is put into practice by using it on the test set provided. The score obtained on Kaggle, which is computed from 30% of the test dataset is 0.8627. This means that the model implemented in this project is probably slightly overfitted to the training dataset.

The model implemented in this project allows the user to correctly predict whether the income of a citizen is above or below the average the average in more than 85% of the cases. Considering its score of 86.27%, it can be stated that the model is close to as good as it can possibly be.

[13] 23.71 % of the individuals in the training set have a higher-than-average income. 76.29 % of the individuals in the training set have a lower-than-average income.

Further research in this matter could include testing the model with sets of features different to the ones tested in this project. This could possibly yield an increased performance of the model. Furthermore, a wider range of training data would most likely improve the model's performance.

VI. REFERENCES

[1] Fernández, Alberto & García, Salvador & Galar, Mikel & Prati, Ronaldo & Krawczyk, Bartosz & Herrera, Francisco. (2018). Learning from Imbalanced Data Sets. 10.1007/978-3-319-98074-4.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html, viewed on 25. Dec. 2020 at 13:07.

[3] Bishop, Christopher. (2006). Pattern Recognition and Machine Learning. 10.1117/1.2819119.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier.score, viewed on 25. Dec. 2020 at 16:14.

[5] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html, viewed on 25. Dec. 2020 at 17:39.

[6] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html, viewed on 26. Dec. 2020 at 15:17.

[7] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html, viewed on 26. Dec. 2020 at 18:06

[8] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, viewed on 26. Dec. 2020 at 18:07

[9] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html, viewed on 26. Dec. 2020 at 18:08

[10] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html, viewed on 26. Dec. 2020 at 18:10

[11] Boehmke, Bradley; Greenwell, Brandon (2019). "Gradient Boosting". Hands-On Machine Learning with R. Chapman & Hall. ISBN 978-1-138-49568-5.

[12] https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm, viewed on 26. Dec. 2020 at 18:18