On income classification prediction using different learning methods

Germán R. Pardo González
2023038054

March 12, 2024

Executive Summary

The Adult dataset was used to determine whether a given individual's income is higher or lower than 50k$. The data was analysed to reveal different insights and drop features that are redundant or not relevant for the above mentioned task. Some preprocessing was performed to maximise the model's accuracy. Three different models were tested: Logistic Regression, Random Forests and Neural Networks, being Random Forests the most promising one, yet computational expensive, achieving an overall accuracy of 86.29%.

Contents

1 Introduction and literature review

The dataset that we are using is known as the "Census Income" dataset. It was extracted by Barry Becker from the 1994 Census database and the main task is to predict whether a specific individual makes more than 50k$ a year [1]. It is composed by several variables that intuitively seem to explain the income of a person. Among these are age, hours per week, sex, race, education, among others. This is a very well studied dataset and hence there is a lot of documentation and past work to solve the problem. Also, this dataset (and along with others) is used as a benchmark to try and compare new algorithms in terms of relevant metrics such as accuracy or fairness. The most relevant goal, taking in to account the context, is to increase the accuracy. we want to have as less missclassifications as possible.

There are several interesting approaches or new algorithms that cite this dataset. For instance, in [4] they propose a Classification Based on Association (CBA) algorithm with two variant models and compare their model with Naïve Bayes, Support Vector Machine, Random Forests and Decision Trees using 18 datasets from the UCI repository. In [5] they propose an algorithm that incorporates the fairness constraint into the clustering problem and use the Adult data set as a benchmark to compare. Specifically, they use its first 25000 records with their 6 numerical features normalised to 0 mean and and variance of 1. Also, they measure the distance between clusters using the l_1 norm. They compare the cost of their algorithm to the cost of an unfair algorithm and they show how the latter is usually lower. Finally, in [3] they propose a *fair regression* under two notions: statistical parity and bounded group loss. They also provide theoretical guarantees on the optimality and fairness on the solutions. As usual, they use the Adult dataset as benchmark to compare their algorithms and show that their algorithms decreases disparity while maintaining accuracy and quality on the solutions. It is interesting how several authors are trying to implement fair algorithms in terms of sensitive features such as gender or race.
This document is organised as follows. Section 2 explains all the preprocessing and insights found in the dataset, section 3 shows the learning methods used, their hyper-parameters tuning and their final evaluation to choose one. Finally, section 4 is the conclusion and future work.

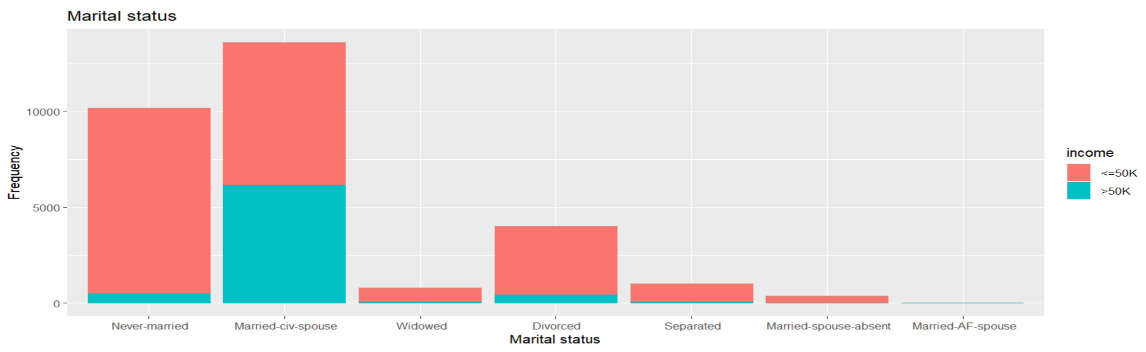
2 Data modelling and insights

2.1 Feature "fnlwgt"

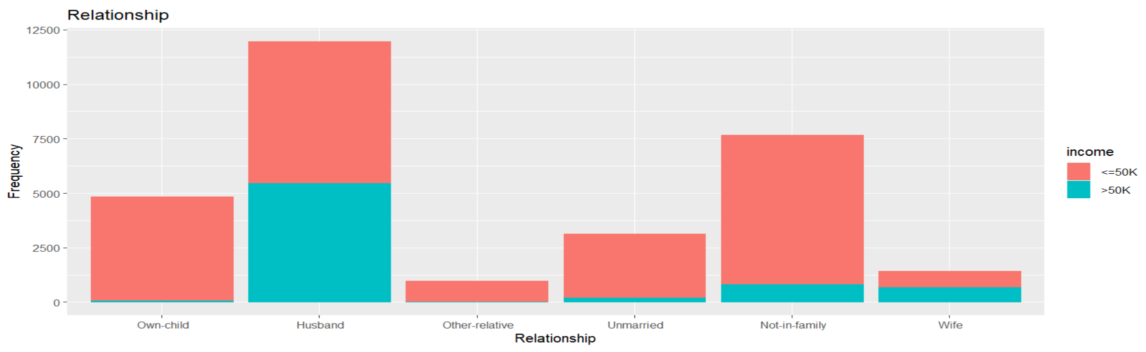
The first important thing to notice in this dataset is the feature named "fnlwgt". This feature is of special importance because it let us have a smaller dimension by accurately representing the frequency of each instance. This variable represents the number of times a particular instance is repeated in the original data. For this reason and to correctly treat this variable, I tried 2 different methods to take it into account. As a first approach, a bootstrap procedure to have a dataset that correctly represents the distribution of each instance. I first calculate the probability that a particular instance is selected (just the value of "fnlwgt" for that instance divided by the sum of "fnlwgt"). After this calculation, I sample B times with replacement from the original dataset with the probability calculated before. As a second approach, I used "fnlwgt" as weight in the learning methods. After experimenting with the two options I decided to just use "fnwlgt" as weight given that the algorithms performed better and the column respective for "fnlwgt" was dropped.

2.2 Insights and EDA

Without taking into account the ”fnlwgt” variable, the dataset consists of 13 features plus the target variable. The features include age, nationality, sex, occupation, among others. The target class is binary and determines whether the income is greater of lower than 50k\$. There are some features that are strongly correlated because they mean exactly the same. For instance, ”education-num” and ”education” are the same just that one of them is a number representing it. For this reason, ”education-num” is dropped.
Now, consider the pair of features ”marital-status” and ”relationship”. These are not identical as in the previous case, but one can notice that ”marital-status” somehow captures and correlates to ”relationship”. The barplots are shown in figure 1 and it was noted that they are not identically, hence I do not drop any of these.



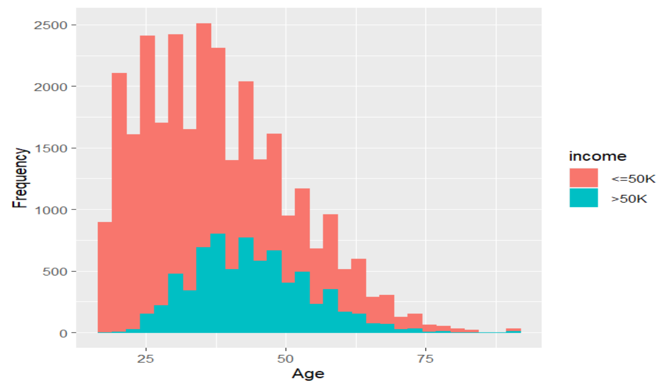
(a) Fig 1.1



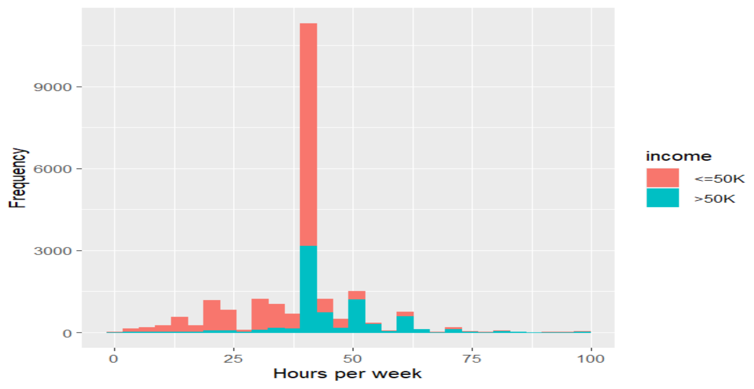
(b) Fig 1.2

Figure 1: Barplot of marital-status and relationship

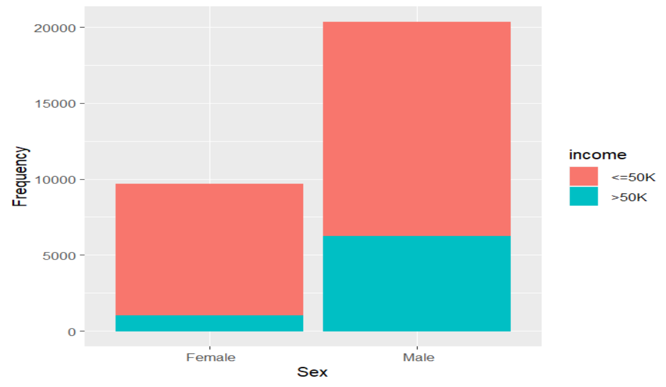
Now, it is useful to understand the data. Consider the following barplots and histograms of ”age”, ”hours-per-week”, ”sex” and ”race”. From figure 2 one can notice that most of the population of the dataset age’s is between around 24 and 40 years old. Also, around the double of females corresponds to males, most of the individuals belong to white ethnicity, followed by black and most of the individuals work between 30 to 50 hours per week (with a insignificant variation of male and female). It is important to take these insights into account and conclude accordingly. Most of the people in the data are white and therefore the income may not be precisely represented or explained by this feature.



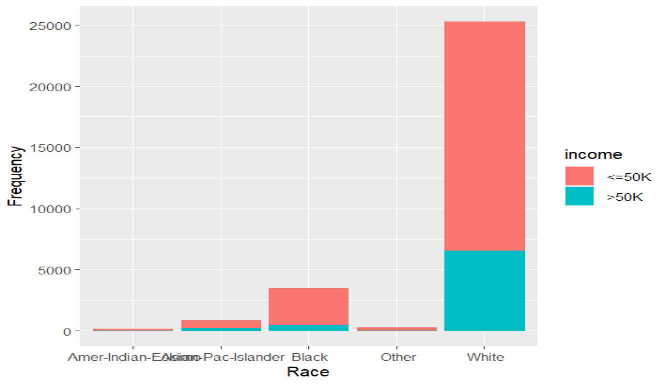
(a) Fig 2.1



(b) Fig 2.2



(c) Fig 2.3



(d) Fig 2.4

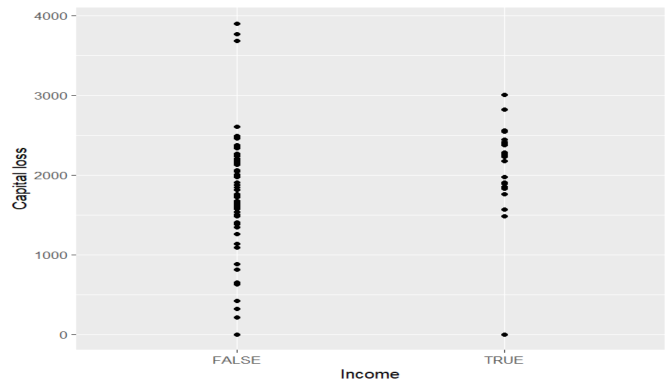
Figure 2: Some histograms and barplots of interesting features

2.3 Imbalanced target

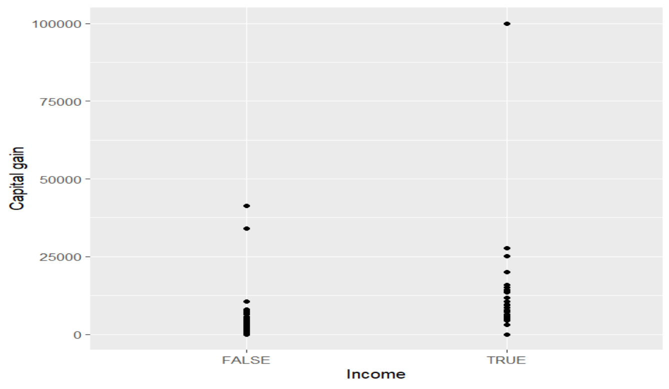
Now, regarding the target this is noticeable imbalanced, but in this case it is not very extreme (only around 25% and 75% between classes). If one want to consider to balance the classes, the models tested would probably lose some of the accuracy, so there is no action taken regarding this matter.

2.4 Outliers and scaling

By having cleaned the dataset I proceed to create train and test set, the train set will have 80% of the whole dataset and the test set will be composed by the rest.
There are two other features that are integer and it is interesting to compare them with the target. Consider figure 3. This figure shows that for "capital-loss" the distribution is wider for the "FALSE" target outcome compared to the "TRUE" outcome. Furthermore, in "capital-gain" there does not seem to be a big difference between the classes, but there is one clear outlier for the class "TRUE". This features along with "age" and "hours-per-week" are being standardised (only in the train set). Regarding the outlier found in capital gain, it was found that it is actually not an outlier and there are 143 (out of the 30000) instances with the value of 99999 for "capital-gain", so indeed it is not an outlier and it is relevant to take that instance into account.



(a) Fig 3.1



(b) Fig 3.2

Figure 3: Scatter plots of capital-gain and capital-loss features

2.5 Missing values

There are only three features that have missing values in the whole dataset represented by ' ?'. These are "workclass", "occupation" and "native-country", the number of missing values in the train set are 1274, 1280 and 506, respectively and the combined unique instances are only 1768 out of the 24000 in the test set. Therefore, there is no action taken for these instances.

3 Results

In this part I will discuss all the learning methods used and its results. It is important to emphasize in the following. To test the hyper-parameters I use k-fold cross validation using a train set, test set and a validation set. The latter is a subset of the training set that changes in every iteration of the cross validation. The test set is left untouched until the end.

3.1 Learning methods and hyper-parameters tuning

3.1.1 Simple, yet powerful: Logistic Regression

This method was used due to its simplicity to be implemented, yet it demonstrated to be very powerful in some cases. However, there are some drawbacks to consider. Furthermore, the data is treated as explained before with 20% test and 80% train.
This model creates one dummy variable for each class in a categorical variable, which is fine as far as there are enough repetitions of every class on a given feature. Nevertheless, if the training set does not consider every category of every categorical variable, this will give an error when this new category appears in the test set. For this reason and to avoid this issue the "native-country" feature was deleted just for this method (bare in mind that it does not mean that this issue could not happen with another categorical feature). Apart from this, the scaling was not applied, since it highly affected the performance in a negative way. Finally, this model does not have hyper-parameters to tune.
Having trained the model with the above specifications, an accuracy of 85.52% was achieved on the test set.

3.1.2 Random Forests

This is a less simple model, however it is usually very powerful. Its computation training time depends on the number of trees to be trained and as well as its performance. Therefore, this model was evaluated using 5 fold cross validation procedure for trees of size 10, 100, 500, 1000 and 2000. It is desired a model whose performance is high but also its training computation time is reasonable. Only the training set was used for the 5 fold cross validation and the validation set was a different subset of the training set at each iteration (and the training was performed with the complement of the validation set within the training set). Finally, the MSE was averaged and this was compared to determine the best hyper-parameter for the model. The results are summarised in table 1. Note,

Table 1: 5 fold cross validation results for Random Forests

Number of trees	Mean MSE	Mean accuracy
10	0.1460	0.8540
100	0.1411	0.8589
500	0.1398	0.8602
1000	**0.1371**	**0.8629**
2000	0.1389	0.8611

for instance, that the difference in accuracy between the model of 1000 trees and the one with 2000 is not very significant and the latter one is way more complicated to train. The hyper-parameter I choose to train the final model is number of trees equal to 1000. In the case were the computation time is very important, one can opt to choose the model of 500 trees since it is easier to train and its accuracy does not vary a lot. However, in this case the computation time is not bad.
After the 5 fold cross validation, the final model with 1000 trees was trained with the whole training set and an accuracy of 78.21% was obtained on the test set.

3.1.3 Neural Network

Now consider a simple neural network. In this case, the size of the neural network is the hyper-parameter (and maybe the number of maximum iterations but this was fixed for 200). I used again 5 fold cross validation to evaluate the model with different sizes from 1 to 6. The following results were obtained in table 2. Note in this case the mean MSE is very similar in almost every case.

Table 2: 5 fold cross validation results for Neural Network

Size of the NN	Mean MSE	Mean accuracy
1	0.1510	0.8490
2	0.1478	0.8522
3	0.1472	0.8528
4	**0.1434**	**0.8566**
5	0.1437	0.8563
6	0.1475	0.8525

Anyway, the highest is chosen which corresponds to a size of 4.
After training the NN with the whole training set with size 4 and a maximum of 200 iterations, an accuracy of 79.22% is obtained. Note that the training of these NNs is highly stochastic and the

performance can vary significantly between runs, having had even accuracies of over 20% in some iterations.

3.2 Evaluation of models

Now that the hyper-parameters for the models have been chosen, I run a 5 fold cross validation for the logistic regression and consider the previous mean MSE obtained from the tuning (of NN and Random Forests) to do the comparison. Table 3 shows the results.

Table 3: 5 fold cross validation results for model evaluation

Model	Mean MSE	Mean accuracy
Logistic Regression	0.1506	0.8494
Random Forests	**0.1371**	**0.8629**
Neural Network	0.1434	0.8566

The Random Forests present on average the highest accuracy, which is the most relevant metric in this context and case study. Therefore, this is chosen as the final model to use.
Some of the information about the learning methods was obtained from [2]

4 Conclusion and future work

In this work, the Adult dataset was used to predict whether a specific individual's income is greater or lower than 50k$ per year. There were some features dropped since they were shown to be redundant and not useful at all. Some preprocessing was done specific for some learning methods and 3 of them were tuned and compared. It was found that the most suitable model (from the ones tried) is the Random Forests with 1000 trees. This showed to have the best accuracy, which is a metric relevant given the context.
Finally, there is some future work that can be done. Since this is a problem of classification and the dataset accounts for some sensitive features such as race, gender, nationality, among others it would be very relevant and innovative to implement (or enhance) algorithms that take into account fairness.

References

[1] B. Becker and R. Kohavi, *Adult*, UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C5XW20, 1996.

[2] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. [Online]. Available: `https://faculty.marshall.usc.edu/gareth-james/ISL/`.

[3] A. Agarwal, M. Dudik, and Z. S. Wu, "Fair regression: Quantitative definitions and reduction-based algorithms," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Jun. 2019, pp. 120–129. [Online]. Available: `https://proceedings.mlr.press/v97/agarwal19d.html`.

[4] M. R. Ayyagari, “Integrating association rules with decision trees in object-relational databases,” *CoRR*, vol. abs/1904.09654, 2019. arXiv: `1904.09654`. [Online]. Available: `http://arxiv.org/abs/1904.09654`.

[5] M. Kleindessner, P. Awasthi, and J. Morgenstern, “Fair k-center clustering for data summarization,” in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Jun. 2019, pp. 3448–3457. [Online]. Available: `https://proceedings.mlr.press/v97/kleindessner19a.html`.