Can adult income be predicted: An analysis on various socioeconomic factors

Abstract—I have used the adult income database to predict whether an adult makes over $50k. The database contains multiple socioeconomic factors on which I hope to make an accurate prediction with.

Index Terms—Random Forest Classification, Naive-Bayes classification, Accuracy, Hyperparameters.

1 Introduction

ADULT income in the United States is a very controversial topic. The United States has some of the largest rates of income inequality in the world. Predicting adult income is a very important task for policy makers, businesses, and society. Analysing what factors contribute to the adult income of a person can help policy makers determine social inequality and mobility and design targeted interventions to support specific demographic groups.

The aim of my project is to provide a mechanism for the prediction of adult income to get some insight into what factors contribute. I will attempt to evaluate these factors with a machine learning model and see whether adult income can be predicted and what socioeconomic factors play into adult income.

2 Data analysis

The dataset for which I trained my models contains 14 features and 1 target variable. My features are divided into 7 categorical features, 6 integer features and 1 binary feature. Each feature being a socioeconomic factor, for example, occupation. These are the inputs to my machine learning model and I will be attempting to predict whether each instance earns over $50k. I will be using the machine learning workflow seen in Figure 1 to achieve this task.

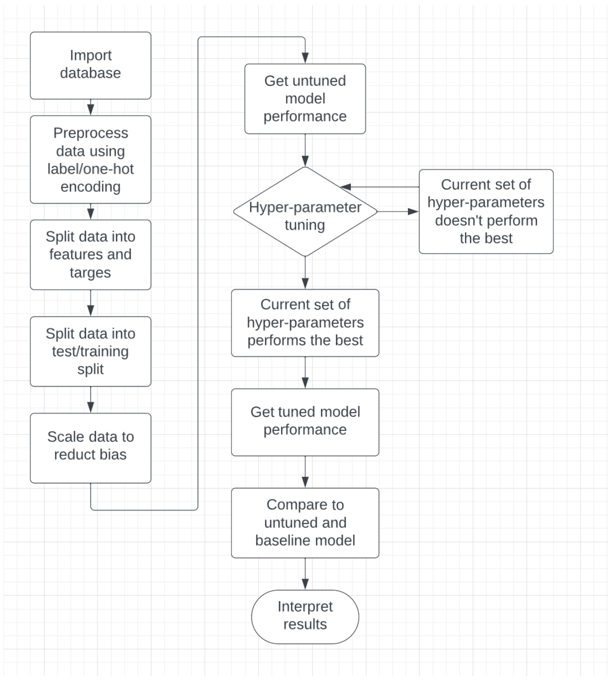
Import database
Preprocess data using label/one-hot encoding
Split data into features and targes
Split data into test/training split
Scale data to reduct bias
Get untuned model performance
Hyper-parameter tuning
Current set of hyper-parameters doesn't perform the best
Current set of hyper-parameters performs the best
Get tuned model performance
Compare to untuned and baseline model
Interpret results

Fig. 1. Machine learning workflow

To analyse my data, I first checked for any imbalance in features and found 6 major imbalances: Native-country, sex, capital-loss, capital-gain, income, and race. See Figure 2

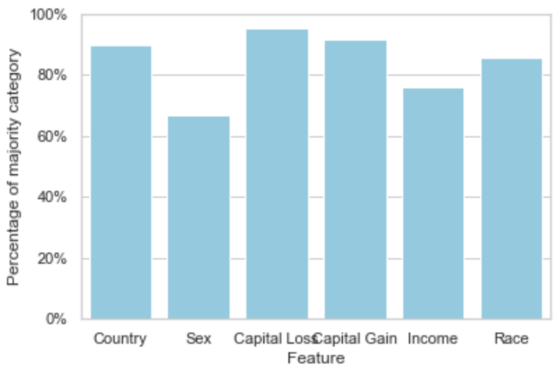

Fig. 2. Percentage of majority classes in imbalanced features

The dataset contains disproportionately more of the following instances: people from the United States, males, individuals of the white ethnicity, and people that make less than $50k. Capital gain and loss also contain mostly zeros. These imbalances will be important to consider later when choosing my models.

2.1 Data cleaning

I first checked for duplicates and dropped any with pd.dropduplicates(). Secondly, I checked for null values. Most of the missing/unknown data in my dataset is marked with a "?". I replaced every instance of "?" with np.nan and then removed every null row with pd.isnull()

The features 'marital-status' and 'education' both contain too many categories for features to fit into. For example, education contains preschool to grade 12. These categories can be condensed into 'below high school'. The same method can be applied to the multiple categories in 'marital-status'. This simplifies the categories in both features into categories that better represent the relationships with other features.

2.2 Data Pre-processing

The first data pre-processing step I implemented is label-encoding. Since classification models often cannot work with text data, I must convert every categorical feature to an integer representation based on the number of unique categories there are in the feature.

The second data pre-processing step I have implemented is one-hot encoding, which involves mapping my integer encoded categorical features to a two-dimensional array where each integer value has been mapped to a column in that array. With this, the number of features gets increased, leading to more features for my algorithms to learn from.

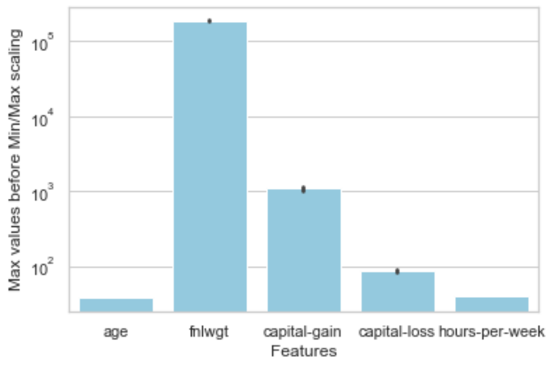

Fig. 3. Range of values before standardisation (Logarithmically scaled)

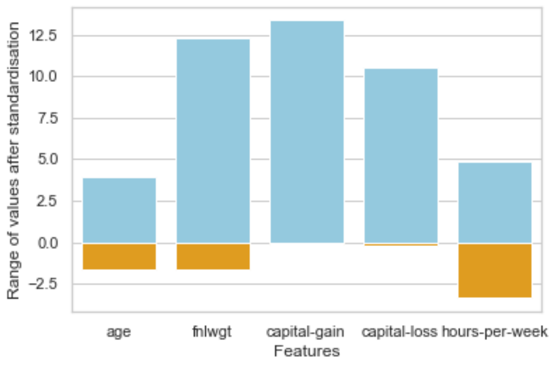

Fig. 4. Range of values after standardisation

The final data pre-processing step I have implemented is the standardisation of my data. In figure 3 we can see there is a large spread between maximum values. Most classification algorithms are affected by features that have significantly higher maximum values than other features. These features introduce bias in the models. This could lead to problems with convergence. Features with larger ranges in their values will dominate the optimisation process, slowing down the overall learning process.

To solve this issue, I use standardisation, which scales every feature using its Z-score. This prevents any feature from dominating the learning process and allows them all to contribute equally. We define the transformation to be,

$$Z = \frac{X - \mu}{\sigma}. \tag{1}$$

3 Model Selection

Since I am attempting to solve a binary classification problem, I chose to compare the performance of Naïve-Bayes and Random-Forest classification in determining whether a person earns more-than or less-than $50k.

I chose the Gaussian Naïve-Bayes algorithm for my baseline algorithm. It is the simplest and most efficient classification algorithm; it extends well to high-dimensional datasets which is particularly important after applying one-hot encoding. Given its minimal number of hyper-parameters, Gaussian Naïve-Bayes allows for computationally inexpensive hyper-parameter tuning, facilitating efficient iteration to enhance model performance.

I chose Random-Forest classification as my proposed algorithm. My dataset has many imbalanced features. One of the major reasons I chose RFC is its ability to handle imbalanced data better than other classification algorithms. My dataset is also very sparse especially after one-hot encoding; capital-loss and capital-gain also contain many missing values. The other major reason I chose RFC is its ability to handle sparse data and missing values. Finally, RFC solves the major cons of Naïve-Bayes classification in that it can capture non-linear and non-independent relationships between classes.

4 Training and Evaluation

The first thing I defined for my models was the training/test split. I decided on a standard 80/20 split. Considering that the size of my dataset has (n) instances, an 80/20 split gives my models enough to train on and ensures that the test set is large enough so that it is not over-fitted by chance.

I optimised both models on hyper-parameters determined using both a grid search and a randomised search. Naïve-Bayes was optimised using a grid search as it only has one major hyper-parameter and RFC was optimised using a random search due to its long training time. I also employed 5-fold cross validation when tuning my hyper-parameters to greatly reduce the chance of over-fitting, ensuring the most generalisation.

To measure model performance, I employed a confusion matrix and classification reports. This allowed to me to see false/true positives and negatives, measure accuracy, precision, recall, and F1 scores for each prediction.

4.1 Naïve-Bayes classification

The only major hyper-parameter Naïve-Bayes has is smoothing. Smoothing is the largest factor in the performance of Naïve-Bayes as it addresses the issue of zero-probabilities in unseen data. I chose 16 smoothing values ranging from $1e^{-12}$ to 1000, plotting the accuracy for each smoothing value.

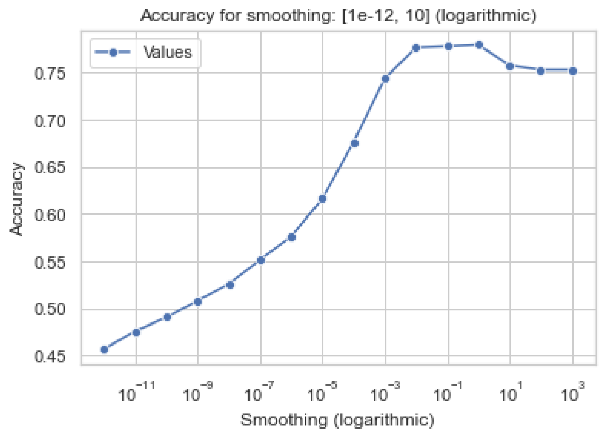

Fig. 5. Accuracy against smoothing value

	Smoothing	Accuracy
Baseline	1×10^{-9}	0.54
Tuned	1	0.77

TABLE 1
Effect of hyper-parameters on accuracy

We can see that the accuracy peaks at 1 which is relatively high, this makes sense considering the sparseness of my dataset. It allows Naïve-Bayes to deal with zero-probabilities easier.

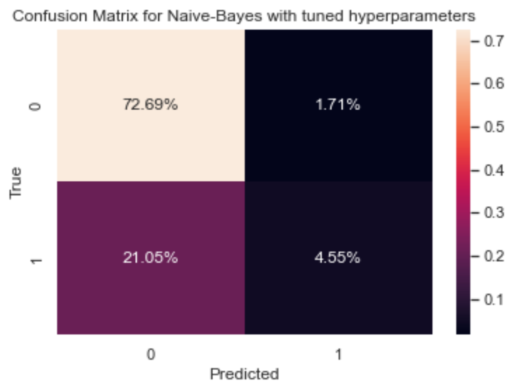

Fig. 6. Confusion matrix for Naive-Bayes post hyper-parameter tuning

	Precision	Recall	F1-score
0	0.78	0.98	0.86
1	0.73	0.18	0.29

TABLE 2
Precision, recall and F1 score for 0 and 1

We can also see that Naïve-Bayes has many false negatives. A lot of features contain a majority class, leading to Naïve-Bayes predicting the majority class more frequently, whether or not its correct.

4.2 Random-forest classification

The hyper-parameters I decided to tune were the number of estimators, the maximum tree depth, the minimum sample split, the minimum sample leaves and the method of calculating the maximum number of features.

	Estimators	Max-depth	Min-split	Min-leaves	Method	Accuracy
Baseline	100	None	2	1	auto	0.83
Tuned	100	None	10	2	auto	0.86

TABLE 3
Effects of hyper-parameters on accuracy

We can see from this table that the model was already well tuned initially. There is little variation in optimal hyper-parameters from the baseline hyper-parameters. The only change is in the minimum sample split/leaves; indicating that the baseline was slightly over-fit.

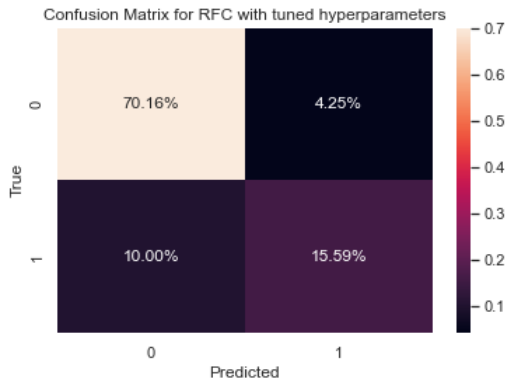

Fig. 7. Confusion matrix for RFC post hyper-parameter tuning

	Precision	Recall	F1-score
0	0.88	0.94	0.91
1	0.79	0.61	0.69

TABLE 4
Precision, recall and F1 score for 0 and 1

5 Model comparison

I tested each model using the same training and test data for both the baseline and tuned models to ensure the same testing conditions. After applying the same experimentation procedure to both models and comparing accuracy, precision, recall and F1-scores, I can confidently say that Random-forest classification is a better algorithm for predicting adult income when compared to Naïve-Bayes.

Naïve-Bayes is fundamentally not a good algorithm for predicting adult income with my dataset, given that my dataset is highly skewed to people that earn less than 50k and sparse. Naïve-Bayes was not able to predict the cases in which people earned over 50k. This can be seen in its recall for predicting over \$50k which is 0.18. Only 18% of the time, it predicted over \$50k correctly. If given data that was skewed in the other direction, Naïve-Bayes would have a very low accuracy.

In comparison, Random-forest classification is a vastly superior model. It outperforms Naïve-Bayes in every metric. However, it still is not clear whether it is a good model for predicting adult income. It still only has a 61% chance of predicting over $50k correctly.

5.1 Feature importance

One of the benefits gained from using RFC is that we can see which features RFC prioritises. This can give us a unique insight into what features play an important role in determining adult income.

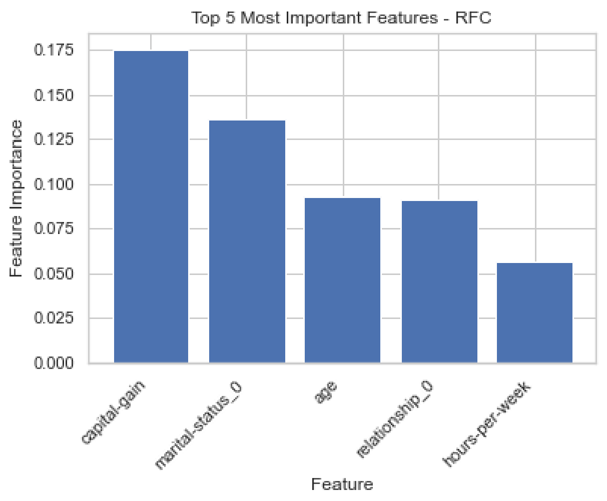

Fig. 8. Feature importance given by RFC

According to this graph, RFC places captial-gain, marital_status_0 (married), age, relationship_0 (husband), and hours per week as the most important factors for determining adult income. These features make sense as to why they are important. Making inferences from Figure 8 we can draw some conclusions about the factors that effect adult income. Capital gain is a clear indicator of wealth as it is the profit earned on the sale of an asset. A higher pay could also be a consequence of marital status, due to the stability afforded by marriage. Also, the age of an individual, and the hours they work, directly correlates with the amount of time they have had to climb through the working hierarchy. Marriage indicates and is a clear indicator of wealth.

The issue with this is that because of the class imbalance, we cannot get an accurate picture of more important socioeconomic factors such as gender or race.

6 Conclusion and self evaluation

I set out to predict adult income based on a multitude of socioeconomic factors. It is still not clear whether my machine learning models can make an accurate prediction on whether an adult makes over $50k. This is because of the large imbalances in my dataset. The dataset tends to over represent specific demographics. Despite the size of the dataset, there is not enough information about other classes for my models to learn about, and this shows in the accuracy of over $50k predictions.

This provides insight into problem but does not solve it. There simply is not enough information about all demographics to assist in the analysis of income inequality. If I could repeat this experiment, I would make my own dataset from national census data to train machine learning models instead. National census data is large enough to where I could get a sufficient sample of each demographic to train models. From that I could make a more accurate prediction.

The other issue with this experiment is the age of the data. The dataset is taken from the 1994 census. The lessons learned from this data cannot be extrapolated to today's society. A dollar in 1994 is worth $2.03 today [1], a wage of $50k is not something only achievable by the top 20% of people and this alone makes this dataset unreliable for predicting adult income in 2024.

The biggest lesson I learned from the coursework was to ensure proper data pre-processing procedures were taking place and no data leakage was occurring. This stops my models from learning patterns that leak from the test data. One example of this is standardising data before splitting it. Not doing this causes massive over-fitting to the test set and can ruin the performance of your models. I also learned to properly cross-validate my models to ensure better generalisation and less over-fitting.

The biggest lesson I learned from lectures was the wide variety of machine learning techniques and algorithms that I could apply to data to make predictions. Specifically the lectures on classification algorithms helped me determine the best algorithms for this dataset and how I could apply them to make predictions on the target variable.

The biggest difficulty I had with this module was with my mathematical understanding of machine learning. Specifically gradient descent and regression. I had to carefully re-read and consolidate my notes plenty of times before finally understanding. Other than that, this module was a nice introduction to machine learning and machine learning workflows.

I think that my coursework differentiates itself by clearly demonstrating the issues with an imbalanced dataset and what it can do to the accuracy of predictions even when using models designed to deal with imbalances.

References

[1] US inflation data 1960-2024 https://www.macrotrends.net/countries/USA/united-states/inflation-rate-cpi