Write the title of your report here

John Smith[1]*, Jennie Smith[1]

Abstract
We found a dataset that could be used for classification tasks. In order to be able to use this dataset we had to do some feature engineering, handle missing values and do some other data cleaning such as label encoding. We chose two applicable models, the Decision Tree and the Random Forst models. The dataset was divided into training, validation and testing. We tuned hyperparameters to get the best possible validation results and to avoid overfitting. When we were satisfied with our models we found that both models performed about tha same with the Random Forest having about on percentage point better results but with much higher training times. We argue that the weighted accuracies of about 85% which at a glance might seem bad, actually are reasonable given the nature of our data sets and the choices we made.

Keywords
Optics — Interference — Diffraction

[1] *Department of Physics, Umeå University, Umeå, Sweden*
*__Corresponding author__: john@smith.com
*__Supervisor__: joe@doe.com

Contents

1. Introduction

Machine learning techniques have plenty of practical use cases. In this report we find a real world, dataset and train two machine learning models on it to try and get the best results possible.

2. Data analysis

2.1 Dataset

The dataset we decided to study is a labeled income prediction dataset. This dataset includes 14 features with information about the people in the study and a label with the income as either more than $50 000 per year or less than or equal to $50 000 per year. This means that we are looking at a binary classification problem. A lot of the features are discrete where only a set number of options available. This includes features such as marital status, education and working class. The dataset features around 32500 data points.

2.2 Data cleaning and feature engineering

There were a couple of things with our dataset that had to be modified in order for it to be usable in our ML application. We find that some of the features are redundant or not interesting in our project. We romove the redundant feature education since there is another already numerically encoded feature containing the same data. We also chose to remove the feature 'fnlwgt' since it is a already calculated number that is used by the Census Bureau to estimate population statistics. Since we want to estimate the population statistics based on the other features and not the already calculated weight we remove this feature. We have a mix of numerical and non-numerical features in our dataset. Since the machine learning models

cannot use non-numerical data we have to encode the non-numercial data into corresponding numbers. This is with the label encoder built into sci-kit learn and used on all non-numerical data.

2.3 Handling missing values

With our numerical version of the dataset we found with the info function in pandas that around 2500 values were NaN values. We reasoned that filling these values with something as the mean of the category does not make very much sense for our application. Since there are many discrete categories a mean value means nothing. Especially since we gave many categories arbitrary numbers the mean means nothing. We therefore decided to only use complete data points. This resulted in removing about 6% of the total amount of data points or about 2500 data points.

2.4 Training, validation and test sets

Before doing any sort of training or analysis on the data, se split it into training, test and validation data. We did this by first splitting a random 20% of the data into test data. This data is reserved for the final testing of the model and will not be touched until the model is finished. Then we did a further split of the rest of the data were 25% was designated as validation data. This data will be used for calibration of the model and hyperparameter tuning. The rest of the data which is 60% of the total data or around 18000 data points will be used to train the model.

3. Model selection

When selecting the model to use for this project we have to limit us to using models that are appropriate to the type of problem that we are trying to solve. The problem is a classification task so all models that are used for regression are immediately invalid. There are plenty of different types of classification models left to choose from. Many of them however, are good for data that has non-discrete features. This includes models such as logistic regression, KNN and other similar types of classification models. Also since we have so many features that are non-numerical and converted into arbitrary numbers these types of models would not be optimal. What is left is the Gaussian Naïve Baye's and the different tree based models. Naïve Baye's can be a bit troublesome for this dataset since we have found that some parameters are slightly correlated. However, this does not necessarliy make in an inappropriate method as it has been found to perform well despite this strict assumption. Therefore we are left with the tree based models such as the decision tree and random forests. We decided to implement two different types of models. We first do a decision tree and see how good we can get that model to work. We then do a random forest which may not be the absolute best model but since it is a continuation on the decision tree it might be interesting to see if it performs better. We then do analysis on both methods and see if these models are good enough and if there is any meaningful difference between the two.

3.1 Data cleaning and feature engineering

There were a couple of things with our dataset that had to be modified in order for it to be usable in our ML application. We find that some of the features are redundant or not interesting in our project. We romove the redundant feature education since there is another already numerically encoded feature containing the same data. We also chose to remove the feature 'fnlwgt' since it is a already calculated number that is used by the Census Bureau to estimate population statistics. Since we want to estimate the population statistics based on the other features and not the already calculated weight we remove this feature. We have a mix of numerical and non-numerical features in our dataset. Since the machine learning models cannot use non-numerical data we have to encode the non-numercial data into corresponding numbers. This is with the label encoder built into sci-kit learn and used on all non-numerical data.

3.2 Handling missing values

With our numerical version of the dataset we found with the info function in pandas that around 2500 values were NaN values. We reasoned that filling these values with something as the mean of the category does not make very much sense for our application. Since there are many discrete categories a mean value means nothing. Especially since we gave many categories arbitrary numbers the mean means nothing. We therefore decided to only use complete data points. This resulted in removing about 6% of the total amount of data points or about 2500 data points.

3.3 Training, validation and test sets

Before doing any sort of training or analysis on the data, se split it into training, test and validation data. We did this by first splitting a random 20% of the data into test data. This data is reserved for the final testing of the model and will not be touched until the model is finished. Then we did a further split of the rest of the data were 25% was designated as validation data. This data will be used for calibration of the model and hyperparameter tuning. The rest of the data which is 60% of the total data or around 18000 data points will be used to train the model.

4. Model selection

When selecting the model to use for this project we have to limit us to using models that are appropriate to the type of problem that we are trying to solve. The problem is a classification task so all models that are used for regression are immediately invalid. There are plenty of different types of classification models left to choose from. Many of them however, are good for data that has non-discrete features. This includes models such as logistic regression, KNN and other similar types of classification models. Also since we have

so many features that are non-numerical and converted into arbitrary numbers these types of models would not be optimal. At first glance, due to the many discrete features Naïve Baye's could be a possible contender. However, the dataset also includes some continious features which complicates things. The different versions of Naïve Baye's aren't really suitable to a mix of discrete and continuous features. Therefore we are left with the tree based models such as the decision tree and random forests. We decided to implement two different types of models. We first do a decision tree and see how good we can get that model to work. We then do a random forest which may not be the absolute best model but since it is a continuation on the decision tree it might be interesting to see if it performs better. We then do analysis on both methods and see if these models are good enough and if there is any meaningful difference between the two.

5. Model Training and Hyperparameter Tuning

5.1 Models and methods used

During the model training there are some important changes we can make to improve the accuracy of our model. One of the most fundemental procedures was hyperparameter tuning which was performed inside a custom class which performs model opitmization and comparison for different models. The class handles the full workflow of tuning the hyperparameters, training the models and recording evaluation metrics. More specifically the method used for hyperparameter tuning is Scikit Learn's GridSearchCV with accuracy as the scoring metric. This method tests different combinations of hyperparameters to establish the best one's. In addition it incorporates cross-validation to prevent overfitting and increase the reliability of the results. For the cross-validation, we used Scikit Learn's stratified k-fold. This type of cross validation is beneficial to use as it preserves the percentage of samples for the classes in each fold, making the model more robust. We used 10 folds for the cross validation, there is of course no "correct" number of folds to use as it's more of a trade off between performance and computational efficiency.

The hyperparameters included in the grid for the decision tree were the maximum depth and the minimum sample split. The maximum depth hyperparameter decides how deep the tree is allowed to go. If a tree is allowed to go very deep there is a high risk of overfitting, on the contrary, a shallow tree will instead risk underfitting. The minimum sample split states how many data points there has to be for a new split to be created. This is also a good measure against overfitting since if it is very low we risk training the noise of the data instead of the general trend and end up overfitting the data. It is also important that it is not to small since we then loose information and underfit instead. For Random Forest the hyperparameters in the grid were maximum depth, minimum sample split and number of estimators, which decides how many trees are used in the Random Forest algorithm.

When performing the hyperparameter tuning, we started out with a rough grid to get a decent estimate of the optimal configuration. From the resluts we then performed a finer grid around the optimal configuration. This way we where able to inspect both a wide range and a more precise range without severly increasing the computational load.

5.2 Caveats and restrictions

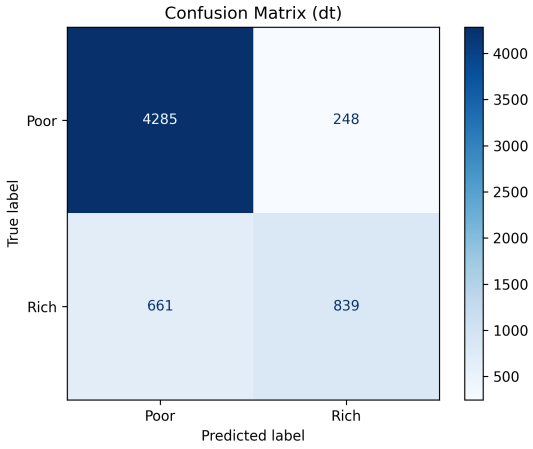
Although the validation results produced from the script are quite promising there are a couple of important notes to make, not only to better understand the final models but also to avoid pitfalls in potential future projects. Firstly, in our script we decided to not use any standardization as this is a sort of unique case where the models used do not require it. However, it's extremely important to understand that if we were to introduce another model, we would need to standardize the data to ensure that the features contribute equally. Secondly, there are more hyperparameters that one might want to consider as we only used a few of them. The problem with expanding the number of hyperparameters in the grid is that it will exponentially increase the computational load. Therefore we picked a few that we thought were most important. Continuing, the scoring metric used is not always the best choice. We used accuracy, meaning the model tries to correctly label as many datapoints as possible and does not care about keeping a similiar precision for both labels. Our goal of this project is somewhat arbitrary, we mainly want to train and compare models. However if such a model were to be used in a real world application, one might want to change the scoring to better adapt the model to the problem at hand.

6. Model Evaluations

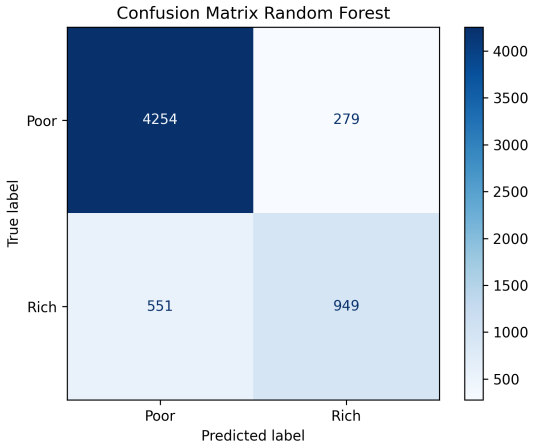
There are two interesting parts to look at after our analysis. One part is to analyze how well the actual models performed and compare the difference between the two models we have chosen to study. We fine tuned our models using the validation part of the data. After running it on the test data we can see how well it actually performs. A great way to get a quick overview of how well a model classifies is to look at the confusion matrix.

6.1 Analyzing the Confusion Matricies

As we can see in the confusion matricies there is not that big of a difference between the models. Both did an overall good job at identifying the two classes. There is a difference in how well the models did in identifying the two different classes. Overall they performed a lot better at classifying the poor people than the rich. We can see that for the both models are pretty good at classifying the poor class and worse at the rich class. The Random forest model is slightly better than the Decision Tree. This is a very interesting result and maybe not so weird as it first seems. There were a lot more poor people in our training data set than rich people. This would of course train our model to be better at classifying the poor. As well as looking at the classification matricies it is interesting to look



(a)



(b)

Figure 1. The confusion matricies of the Decision Tree model and the Random Forest model on the test data.

at the actual performance metrics that can be calculated from the matricies.

6.2 Analyzing Weighted Performance Metrics

We want to analyze to sets of metrics. First we have the validaton Metrics. These metrics can be seen in table(1). Then we have the actual test metrics which is the result from our model. These can be seen in table(2). Of note is that all of these metrics are calculated as weighted metrics which means that they account for the class imbalances seen in the confusion matrcies. Looking at the values we see that the

Table 1. The performance metrics of the models on the validation data.

Model	Accuracy	Precision	Recall	F1 Score	Total Time
RF	0.8589	0.8535	0.8589	0.8534	150.8154
DT	0.8483	0.8449	0.8483	0.8462	6.7357

Table 2. The performance metrics of the models on the test data.

Model	Precision	Recall	F1 Score
RF	0.86	0.86	0.86
DT	0.84	0.85	0.84

difference between our models is not that large. The Random forest model is on average about 1 percentage point better than the Decision Tree. We can also see that all metrics are at about 0.85. This means that our models are not very accurate and that the differences between them is not that large at all. Which model that is better depends a lot on what is the priority. While it is clear that the Random Forest has the better performance, even by just a little bit, it is also significanty slower on the validation data. So for this dataset was it really worth 30x the computational time to get a slightly better result? We are not really sure. The extra computational time is a definite negative but at the size of this dataset we are only talking about a couple of minutes which is not too bad. For another dataset the results may be different and it might be clearer which is really the prefered model.

Another thing to consider is the interpretability of the models. Here, there is quite a big difference that could possibly outweigh one model over the other. Starting with the Decision Tree, because the model's prediction process is quite simple, it is also highly interpretable. We can even plot the decision tree to see how the model handles every feature for a datapoint. This can be beneficial if we want to better understand the model. In contrast, Random Forest uses a more complicated method for prediction as it takes the averages over numerous decision trees with random subsets of features. This means that the model is more or less a black box. The importance of model interpretability is difficult to define as it will vary between different applications and there is even a subjective element to its importance.

6.3 Analyzing the Performance

At a first glance at both the confusion matricies and the performance metrics the models do not look to be that good. But what has to be considered is the data that we are analyzing. We are looking at what possible indicators there are for a person to earn more than a certain amount of money. This is real world data and in the real world there is a lot of unique ways of earning money. While there certainly are some indicators that will clearly tell that somebody is earning a lot of money, there are other factors that are not as telling. This means that some features are less important than others. This can be seen in our models int he feature importance graphs in figure(2(a)) and (2(b)). This also means that there will be plenty of outliers in the data. No matter how good the model is, it cannot possibly catch all of these outliers. If it did it would be overfitted. We simply cannot expect a model to have very good accuracy on this type of data set.

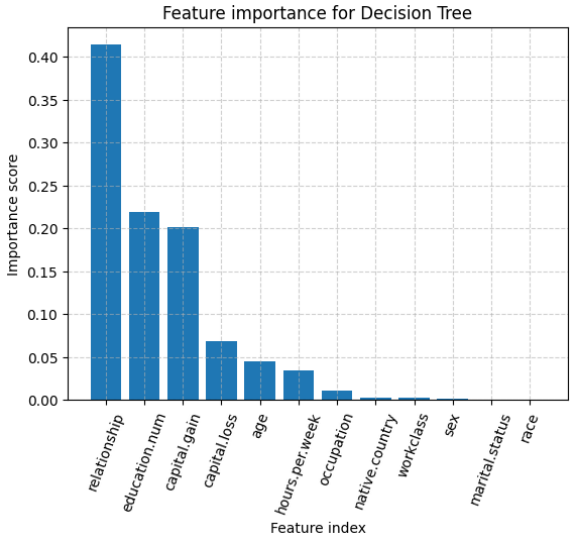
An important thing to touch on is the poor fit on rich people by our model. We see that only 60-70% where correctly identified which is quite bad. As we talked about above there may be many data reasons for this poor fit. Of note is that we have optimized this model to find the best accuracy on all data point. We therefore stride to classify as many total data points correctly as possible and not on getting the best average for the classes separetly. Since there are more poor people in our dataset it is very reasonable for the model to have optimised for that as well since it gives the best weighted accuracy.

6.4 Overfitting and Underfitting

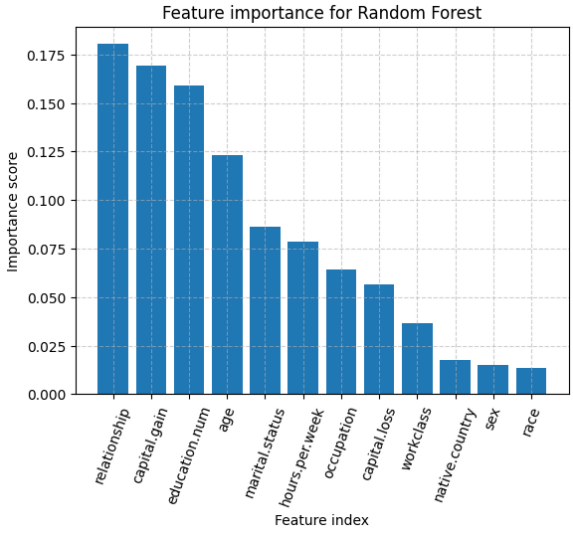
We spent some time tuning the hyperparameters to ensure that we did not overfit. If we compare the validation results with the test results we see that the performance metrics do not change much at all. This is what we want to see as this means that we have avoidede overfitting the model. This means that our model could be used on other similar datasets and hopefully give similar perfomances. We also do not want our model to be underfit. This is a bit harder to validate as we want the errors to be as small as possible for both training and testing and as we stated before I believe that this is a difficult dataaset to get a great fit to. Therefore we believe that we have found a model that has a decent enough balance between bias and variance.

6.5 Feature Importance

Taking a closer look at the feature importance graphs of the two models we notice an interesting difference. The Decision tree which is only one tree focuses has only a few main features where one is the most important. The rest are not used that much or almost not at all. The Random Forest uses a far wider range of features. They also rank the features a bit differently and the best feature for one model is not the best for the other. We considered removing the worst performing features to see if it would make a difference in the performanes. But since they have diffrent worst performing features we reasoned that to keep the comparison as fair as possible it would be more interesting to leave the features as is.



(a)



(b)

Figure 2. The feature importance graphs for the Decision Tree model and the Random Forest model.

7. Summary

We have succesfully trained two different but similar machine learning models on classifying the monetary status of people based on a bunch of different features. While some trade offs where made in regards to which features where kept and to what we optimized the model for. We still managed to get a respectable result especially regarding the difficult type of data that we had to work with.

References

[1] Steinhaus, H., Mathematical Snapshots, 3rd Edition. New York: Dover, pp. 93-94, (1999)

[2] Greivenkamp, J. E., Field Guide to Geometrical Optics, SPIE Press, Bellingham, WA, (2004)

[3] Pedrotti, F.L. and Pedrotti, L.S., Introduction to Optics, 3rd Edition, Addison-Wesley, (2006)

[4] UC Davis ChemWiki, Propagation of Error, Available at: https://chem.libretexts.org/Textbook_Maps/Analytical_Chemistry/Supplemental_Modules_(Analytical_Chemistry)/Quantifying_Nature/Significant_Digits/Propagation_of_Error, (Accessed: 10th March 2016).