MISCADA Core IIA (3) Classification

Bingchao Wang

20 March 2020

Github repository: https://github.com/chuckwong13/MISCADA_CoreIIA_Classification.git

Part I: Executive summary

In this project, we are going to analyze "adult" data from UCI, which contain people's personal information like age, workclass, education, occupation, capital gain and so on. Our target is to use these features to train models to predict if a person earns more than 50000 dollars a year.

In real-world, these model can be very practical because companies like investment banks, insurance companies, luxury goods company focus more on high-net-worth clients, whose annual income are more likely over 50000 dollars. By using these models, we can tell if a person is a potential target client based on other features we have, which helps improve efficiency while telemarketing and digital marketing. Meanwhile, companies can reduce advertising costs by achieving this, and people with lower income can avoid harassed by irrelevant advertising.

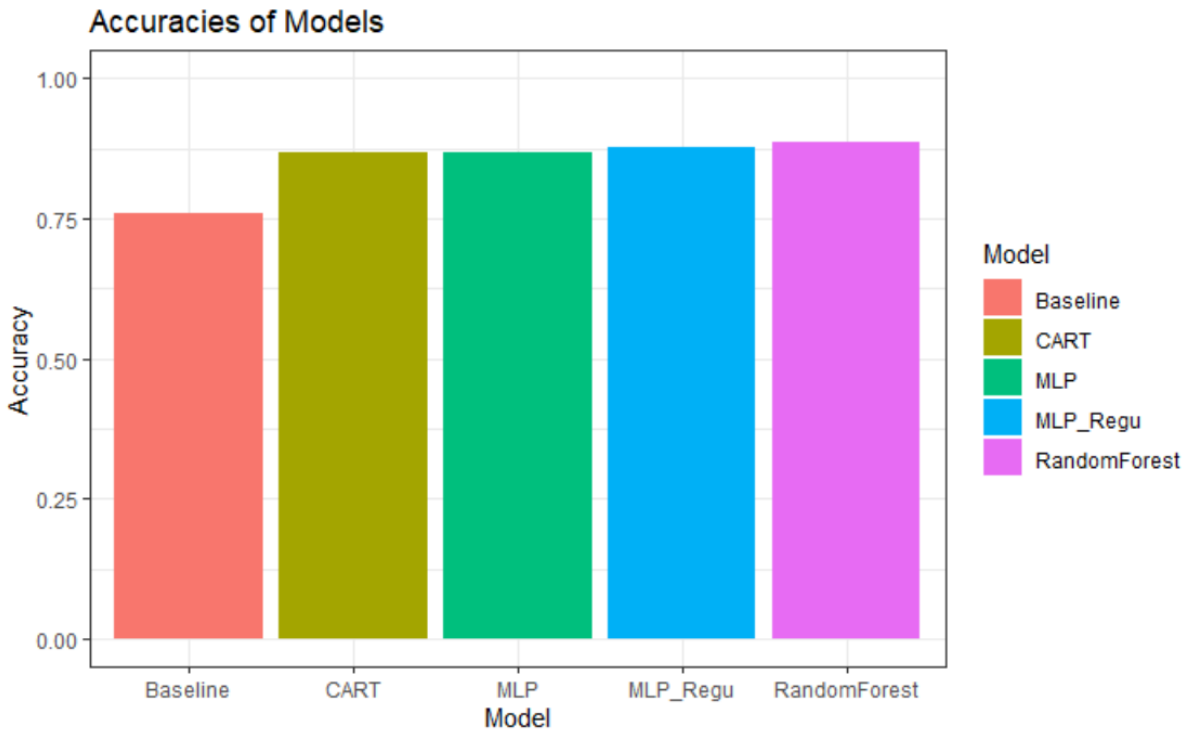


Figure 1: Accuracies of trained models.

We fit five predictive models - a baseline model, a CART (Classification And Regression Trees) model, a random forest model, an MLP (Multi-Layer Perceptron) model and an MLP model

with regularisation. Their accuracies (Number of successful predictions/ Number of predictions) are shown in Figure 2. According to the figure, we can observe that random forest performs the best among these models, it can achieve the accuracy of 88%. Compared with the baseline model (randomly choose a class), it made great progress.

However, accuracy is not the most important objective we care about. With the purpose of targeting high-income clients, companies are more interested in the true positive rate (the percentage of rich people who are correctly identified as rich). And the true positive rates of different models are shown below.

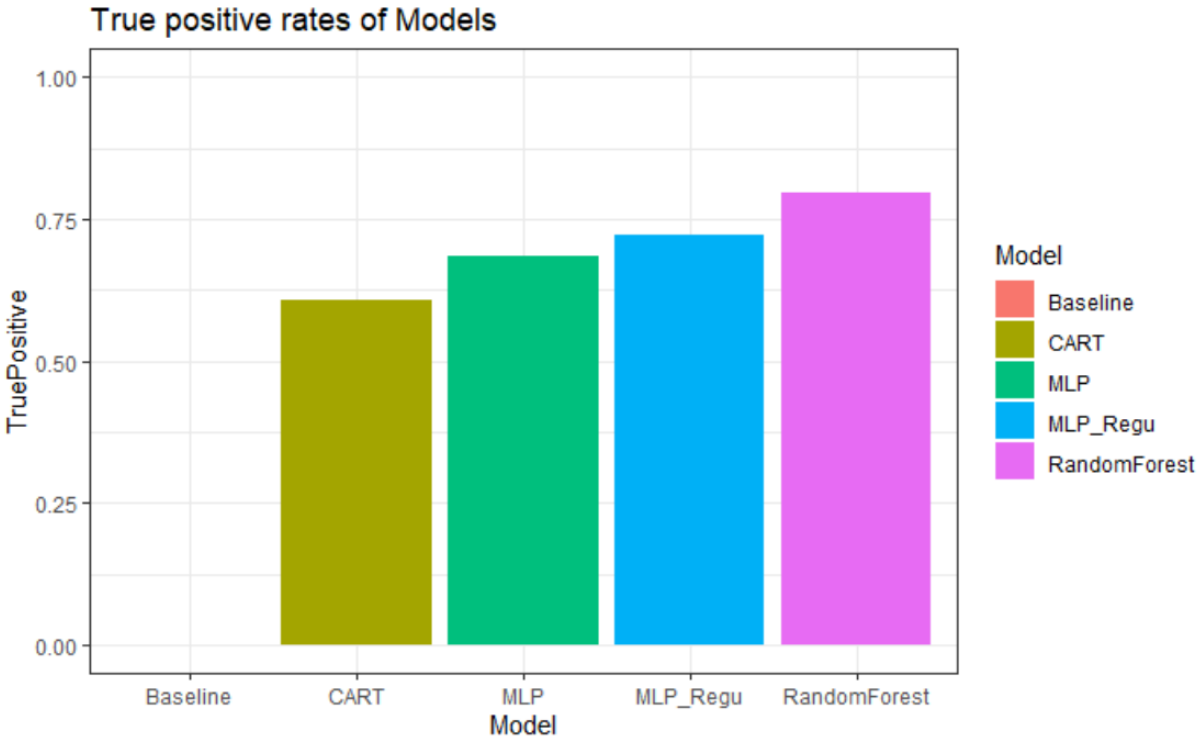


Figure 2: True positive rates of trained models.

It seems that the random forest model wins again, by achieving the true positive rate of 79%. Considering the performance both in accuracy and true positive rate, we can know that the random forest model is the best among these models, and it can predict nearly 80 percent if a person earns over 50000 dollars per year. Besides, the MLP-regularisation performs not bad in this case, with the accuracy of 88% and the true positive rate of 72%.

Part II: Technical summary

The modelling process can be divided into three parts – data preprocessing, predictive models training and performance comparison.

A. Data Preprocessing

1. Overview of Data

```
-- Variable type: factor ---------------------------------------------------------------
# A tibble: 9 x 6
  skim_variable  n_missing complete_rate ordered n_unique
* <chr>              <int>         <dbl> <lgl>      <int>
1 workclass           1836         0.944 FALSE          8
2 education              0             1 FALSE         16
3 marital_status         0             1 FALSE          7
4 occupation          1843         0.943 FALSE         14
5 relationship           0             1 FALSE          6
6 race                   0             1 FALSE          5
7 sex                    0             1 FALSE          2
8 native_country       583         0.982 FALSE         41
9 annual_income          0             1 FALSE          2
  top_counts
* <chr>
1 " Pr: 22696,  Se: 2541,  Lo: 2093,  St: 1298"
2 " HS: 10501,  So: 7291,  Ba: 5355,  Ma: 1723"
3 " Ma: 14976,  Ne: 10683,  Di: 4443,  Se: 1025"
4 " Pr: 4140,  Cr: 4099,  Ex: 4066,  Ad: 3770"
5 " Hu: 13193,  No: 8305,  Ow: 5068,  Un: 3446"
6 " Wh: 27816,  Bl: 3124,  As: 1039,  Am: 311"
7 " Ma: 21790,  Fe: 10771"
8 " Un: 29170,  Me: 643,  Ph: 198,  Ge: 137"
9 " <=: 24720,  >5: 7841"

-- Variable type: numeric --------------------------------------------------------------
# A tibble: 6 x 11
  skim_variable  n_missing complete_rate     mean        sd    p0    p25    p50    p75
* <chr>              <int>         <dbl>    <dbl>     <dbl> <dbl>  <dbl>  <dbl>  <dbl>
1 age                    0             1     38.6     13.6     17     28     37     48
2 fnlwgt                 0             1 189778.  105550.    12285 117827 178356 237051
3 education_num          0             1     10.1      2.57     1      9     10     12
4 capital_gain           0             1   1078.    7385.      0      0      0      0
5 capital_loss           0             1     87.3    403.      0      0      0      0
6 hours_per_week         0             1     40.4     12.3     1     40     40     45
```

From the summary of the dataset, we know that it has 9 categorical characteristics and 6 numeric characteristics. For factor variables, "workclass", "occupation" and "native country" have missing data; besides, "education", "occupation" and "native country" have more than 10 levels, which is unnecessary and may easily lead to overfitting.

2. Cleaning Data

Firstly, we begin by minimizing the factor variables' levels to make them easy to understand and meaningful. For the "native_country" variables, what we can do is to group several countries by different regions:

Original levels	**New Levels**
outlying_us	South, Outlying-US(Guam-USVI-etc)
Southeast_Asia	Vietnam, Laos, Cambodia, Thailand
Asia	China, India, Hong, Iran, Philippines, Taiwan
NorthAmerica	Canada, Cuba, Dominican-Republic, Guatemala, Haiti, Honduras, Jamaica, Mexico, Nicaragua, Puerto-Rico, El-Salvador, United-States
SouthAmerica	Ecuador, Peru, Columbia, Trinadad&Tobago
Europe	France, Germany, Greece, Holand-Netherlands, Italy, Hungary, Ireland, Poland, Portugal, Scotland, England, Yugoslavia, France

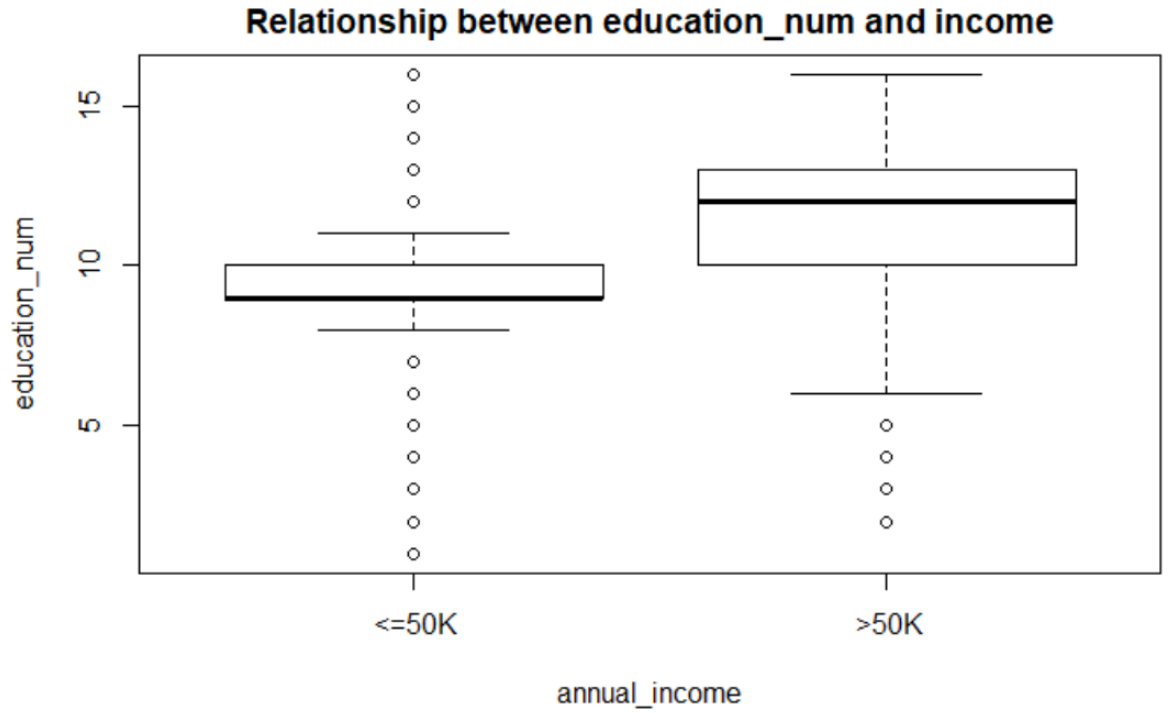
Similarly, we can apply this method to collapse levels of factor variables "education" and "occupation".

Secondly, in order to deal with missing data, we will randomly impute data from that variable by using package "missForest". Other than making all the missing data unknown, this method

helps models improve their accuracies by nearly 3%.

3. Analysis of Data Relationships

There are some features we should pay more attention to, such as workclass, education, working hours and occupation. In real life, we can assume that well-educated people are more likely to have high salary jobs, so let's have a look at relationships between education number and annual income.



It is obvious that the more educated, the easier people might have an over 50000 dollars annual income. The same situation happens in variables such as working hours, ages, occupation, etc. By this reason, I think CART is a proper model to fit these datasets.

B. Predictive Models Training

1. Baseline and CART

Because of the imbalance of our response variable – 7841 over 50k, 24720 below, we need to introduce a baseline model to see if our models work. It shows that the baseline model can gain 76% accuracy and 50% AUC. Then we fit a CART model by using the package "mlr3verse". With 10-fold cross-validation, this model can reach 86% accuracy, 87% AUC and 61% true positive rate. To improve this model, nest-validation was employed, from the cost penalty-error plot, we can set cp as 0.023 to avoid overfitting.

However, the model accuracy drops to 86% and AUC fall to 86%, but the true positive rate increase to 65%. The result is good, but let's try constructing a multitude of decision trees(random forest) and see if it can improve.

2. Random Forest

By using the package "randomForest", we trained a random forest model with 500 trees. Similarly, applying cross-validation, this model has 88% accuracy, 93% AUC and 79% true positive rate.

3. Multi-Layer Perceptron

Recalling that we have a huge dataset, containing 32561 observations of 15 variables, I think

it quite suitable to train a neural network. Instead of applying cross-validation, here we split the datasets into train/test/validate data, turn all the categorical variables into one-hot coding and scale all the numeric variables to mean 0, standard deviation 1. As for loss function and optimizer choices, we set binary cross-entropy and Adam respectively.

At first, I tried to train a neural work with 5 layers, whose units number are 256, 128, 128, 64, 1. To avoid overfitting, batch normalization and layer dropout(0.2) were added. After training, this model can get 87% accuracy, 93% AUC and 66% true positive, which is a good result except for the true positive rate compared with the random forest.

Aiming at getting a better true positive rate, we are going to raise the dropout rate to 0.4 and embed L2 regularizer(lambda = 0.0001) into each layer. Besides, "callbacks" function is utilized to save the best model we trained. The training curves of loss and accuracy are shown below.

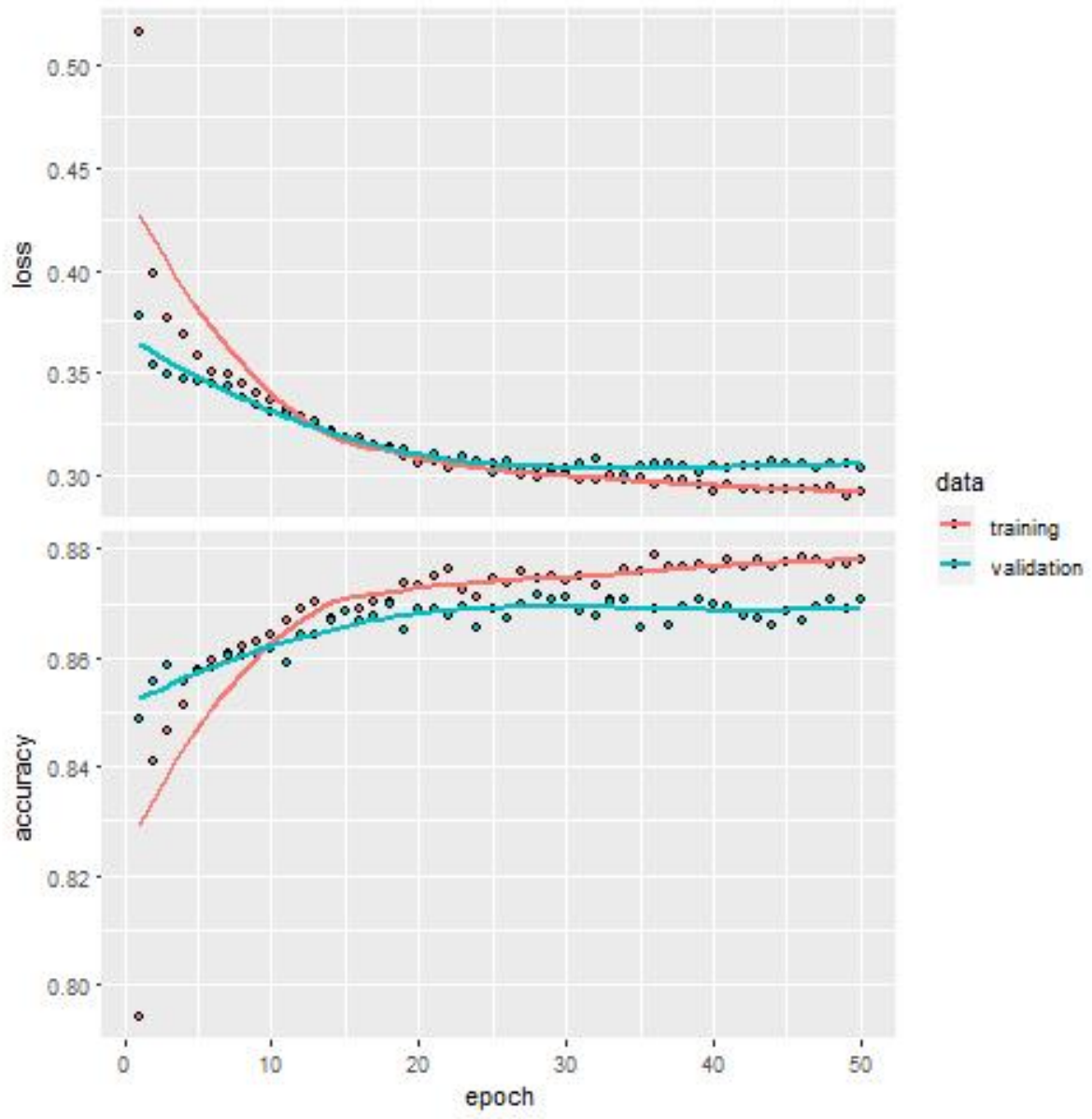


Figure 3: Training curves of MLP-regularisation.

This time we get a better result as wishes. Even though the increases in accuracy and AUC is small, 88% and 93% respectively, the true positive rate boost from 66% to 72%, which is huge progress.

C. Performance Comparison

From the executive summary, we can know that the random forest model and the MLP-regularisation model perform the best among all these models. But there is still an objective we should care about – AUC. Literally, AUC means the area under the receiver operating characteristics curve, in other words, it tells us how well the model can distinguish between people with high income and low income.

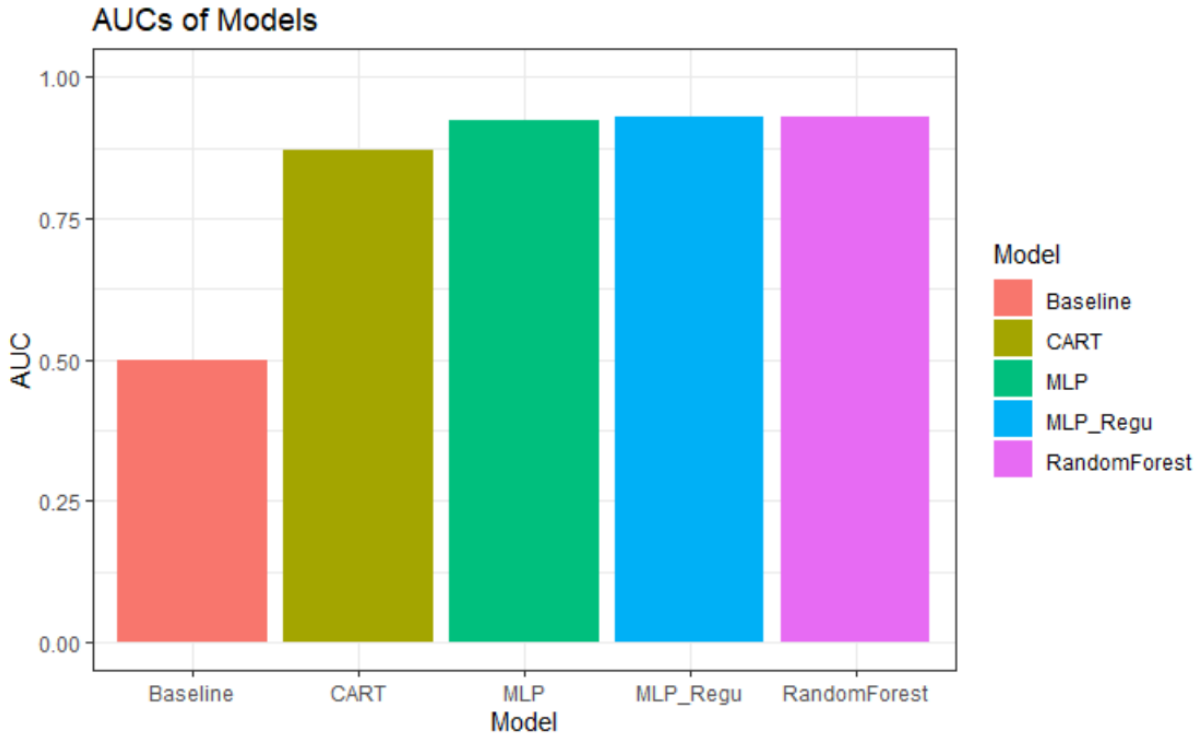


Figure 4: AUC of trained models.

The random forest, the MLP and the MLP-regularisation nearly achieve the same AUC in this case at around 93%, while the CART has only 87%. Considering the performance in true positive rate, the random forest is the best model despite MLP-regularisation performs not bad at both accuracy and AUC.