Predicting Income - HarvardX Capstone Project

Kevin Jané

2022

Contents

Chapter 1

Preface - Introduction

The purpose of this capstone project is to create a predictive project to achieve the Professional Certificate of the Data Science[1] courses taught by Harvard University.

The data science job market is exponentially growing being in the top 3 of jobs most sought after[2], this can allow us to infer that the world is giving so much importance to open data than it was years ago, recognizing the potential of data analysis and prediction models for the global social-economic development.

I as an undergraduate economics student, being passionate about data, being able to manipulate data with R facilitates doing data analyses. Also, predictive models are essential to our, which become more time-efficient with R.

My lovely small country, Paraguay, in the center of South America, also called the "heart of South America". A country that has been growing these last years, but the needing of data, people who analyse the data and who investigate the behavior of the economy and everything that is going on in the country, created the my love for the data analysis.

In this project well going to use the 1994 Census Income Data Set[3], that is

[1] https://www.edx.org/es/professional-certificate/harvardx-data-science

[2] https://www.forbes.com/sites/forbeshumanresourcescouncil/2021/05/20/hr-leaders-share-14-in-demand-skills-employers-want-in-2021/?sh=44ba748d1e45

[3] http://www.census.gov/ftp/pub/DES/www/welcome.html

a dataset donated by Ronny Kohavi and Barry Becker and provided by the UCI Machine Learning Repository. One variable is related with income, our goal in this project is trying to predict the income based on data from the Census database.

Chapter 2

Exploratory Data Analysis

2.1 Data Preparation

In this section, we install and load every packages required for this project, as well as the 1994 Census database provided.

```
if(!require(randomForest)) install.packages("randomForest")
if(!require(reldist)) install.packages("reldist")
if(!require(readxl)) install.packages("readxl")
if(!require(dplyr)) install.packages("dplyr")
if(!require(tidyr)) install.packages("tidyr")
if(!require(dslabs)) install.packages("dslabs")
if(!require(stringr)) install.packages("stringr")
if(!require(forcats)) install.packages("forcats")
if(!require(ggplot2)) install.packages("ggplot2")
if(!require(caTools)) install.packages("caTools")
if(!require(rpart.plot)) install.packages("rpart.plot")
if(!require(ISLR)) install.packages("ISLR")
if(!require(e1071)) install.packages("e1071")
if(!require(OneR)) install.packages("OneR")
if(!require(tidyverse)) install.packages("tidyverse",
                                         repos = "http://cran.us.r-project.org")
if(!require(ggthemes)) install.packages("ggthemes",
                                        repos="http://cran.us.r-project.org")
```

```r
if(!require(caret)) install.packages("caret",
                                     repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table",
                                          repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart",
                                     repos="http://cran.us.r-project.org")
if(!require(MLmetrics)) install.packages("MLmetrics",
                                         repos="http://cran.us.r-project.org")
if(!require(haven)) install.packages("haven",
                                     repos="http://cran.us.r-project.org")
```

```r
library(randomForest)
library(reldist)
library(readxl)
library(dplyr)
library(tidyr)
library(dslabs)
library(stringr)
library(forcats)
library(ggplot2)
library(tidyverse)
library(OneR)
library(caret)
library(data.table)
library(ggthemes)
library(caTools)
library(rpart)
library(ISLR)
library(e1071)
library(MLmetrics)
library(haven)
library(hrbrthemes)
library(viridis)
library(rpart.plot)
```

```r
set.seed(1, sample.kind = "Rounding")
trainFileName = "adult.data"; testFileName = "adult.test"

if (!file.exists (trainFileName))
    download.file (
      url = "http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult
      destfile = trainFileName)

if (!file.exists (testFileName))
    download.file (
      url = "http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult
      destfile = testFileName)
colNames = c ("age", "workclass", "fnlwgt", "education",
              "educationnum", "maritalstatus", "occupation",
              "relationship", "race", "sex", "capitalgain",
              "capitalloss", "hoursperweek", "nativecountry",
              "incomelevel")

adult = read.table (trainFileName, header = FALSE, sep = ",",
                    strip.white = TRUE, col.names = colNames,
                     na.strings = "?", stringsAsFactors = TRUE)
```

```r
str(adult)
```

```
## 'data.frame':    32561 obs. of  15 variables:
##  $ age          : int  39 50 38 53 28 37 49 52 31 42 ...
##  $ workclass    : Factor w/ 8 levels "Federal-gov",..: 7 6 4 4 4 4 4 6 4 4 ...
##  $ fnlwgt       : int  77516 83311 215646 234721 338409 284582 160187 209642 45
##  $ education    : Factor w/ 16 levels "10th","11th",..: 10 10 12 2 10 13 7 12 1
##  $ educationnum : int  13 13 9 7 13 14 5 9 14 13 ...
##  $ maritalstatus: Factor w/ 7 levels "Divorced","Married-AF-
spouse",..: 5 3 1 3 3 3 4 3 5 3 ...
##  $ occupation   : Factor w/ 14 levels "Adm-clerical",..: 1 4 6 6 10 4 8 4 10 4
##  $ relationship : Factor w/ 6 levels "Husband","Not-in-
family",..: 2 1 2 1 6 6 2 1 2 1 ...
##  $ race         : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 5 3 3 5 3 5 5
##  $ sex          : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 1 1 2 1 2 ...
```

```
##  $ capitalgain  : int  2174 0 0 0 0 0 0 0 14084 5178 ...
##  $ capitalloss  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hoursperweek : int  40 13 40 40 40 40 16 45 50 40 ...
##  $ nativecountry: Factor w/ 41 levels "Cambodia","Canada",..: 39 39 39 39 5 39
##  $ incomelevel  : Factor w/ 2 levels "<=50K",">50K": 1 1 1 1 1 1 1 2 2 2 ...
```

The adult dataset contains **32561** rows and **15** variables, wich are:

- age <int>: continuous.
- workclass <Factor>: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
- fnlwgt <int>: continuous.
- education <Factor>: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
- educationnum <int>: continuous.
- maritalstatus <Factor>: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
- occupation <Factor>: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
- relationship <Factor>: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- race <Factor>: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
- sex <Factor>: Female, Male.
- capitalgain <int>: continuous.
- capitalloss <int>: continuous.

- hoursperweek <int>: continuous.
- nativecountry <Factor>: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad Tobago, Peru, Hong, Holand-Netherlands.
- incomelevel <Factor>: >50K, <=50K.

NA values

```
na_v <- sapply(adult, function(x) sum(is.na(x)))
na_v
```

```
##           age     workclass        fnlwgt     education  educationnum
##             0          1836             0             0             0
## maritalstatus    occupation  relationship          race           sex
##             0          1843             0             0             0
##   capitalgain   capitalloss  hoursperweek nativecountry   incomelevel
##             0             0             0           583             0
```

Percentage of NA values

```
pna_v <- sapply(adult, function(adult){sum(is.na(adult)==T)*100/length(adult)})
round(pna_v, digits = 3)
```

```
##           age     workclass        fnlwgt     education  educationnum
##         0.000         5.639         0.000         0.000         0.000
## maritalstatus    occupation  relationship          race           sex
##         0.000         5.660         0.000         0.000         0.000
##   capitalgain   capitalloss  hoursperweek nativecountry   incomelevel
##         0.000         0.000         0.000         1.790         0.000
```

We can see that the variables `workclass` (**5.639%**),`occupation`(**5.660%**) and `nativecountry`(**1.790%**) have NAs. Actually, this is not a good thing because these variables could be a very good predictors of income.

So, we want to remove all the NAs from the dataset

```
adult = adult[!is.na(adult$workclass) & !is.na(adult$occupation),]
adult = adult[!is.na(adult$nativecountry),]
adult$fnlwgt = NULL
```

2.2 Data Analysis

The variables/columns used in `adult` are:

- age <int>
- workclass <Factor>
- education <Factor>
- educationnum <int>
- maritalstatus <Factor>
- occupation <Factor>
- relationship <Factor>
- race <Factor>
- sex <Factor>
- capitalgain <int>
- capitalloss <int>
- hoursperweek <int>
- nativecountry <Factor>

We want to see the distribution of income between the variables, we can plot it and see their behavior.

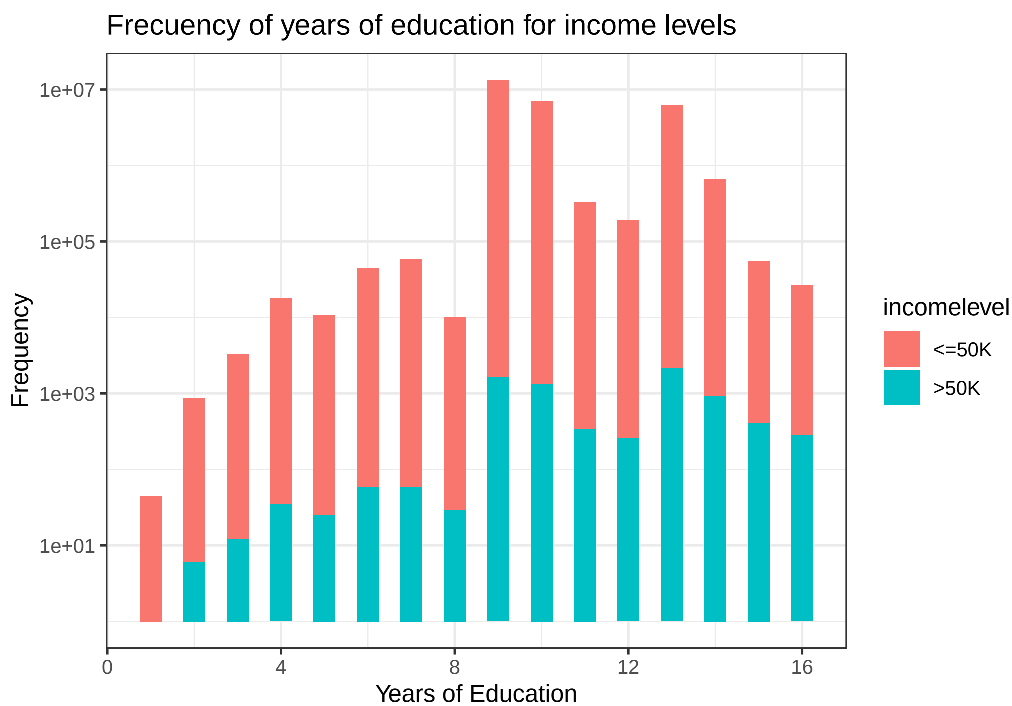
In the next plot we see the frequency of ages in the database, with the condition of the income.

```
adult %>%
  ggplot(aes(age)) +
  geom_histogram(aes(fill=incomelevel), binwidth = 0.5) +
  theme_bw() + xlab("Age") + ylab("Frecuency") +
  ggtitle("Frecuency of ages for income levels")
```

In the next plot, we graph the frequency of years of education with the condition of the level of income.

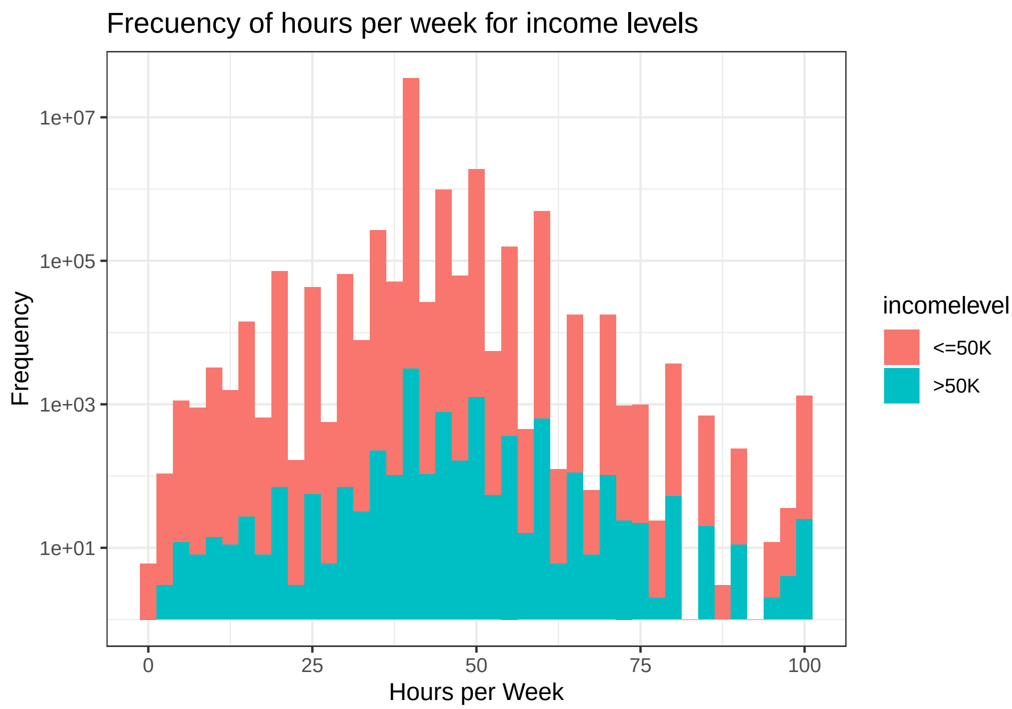
We actually see that since 9 years of studying dedication, that is a high school degree, the frequency of people who earns **<=50K** are the predominant. But also there is more frequency of people who earns **>50K** than 8 years of study or earlier.

```
adult %>%
  ggplot(aes(educationnum)) +
  geom_histogram(aes(fill=incomelevel), binwidth = 0.5) +
  scale_y_log10() + theme_bw() +
  xlab("Years of Education") + ylab("Frequency") +
  ggtitle("Frecuency of years of education for income levels")
```



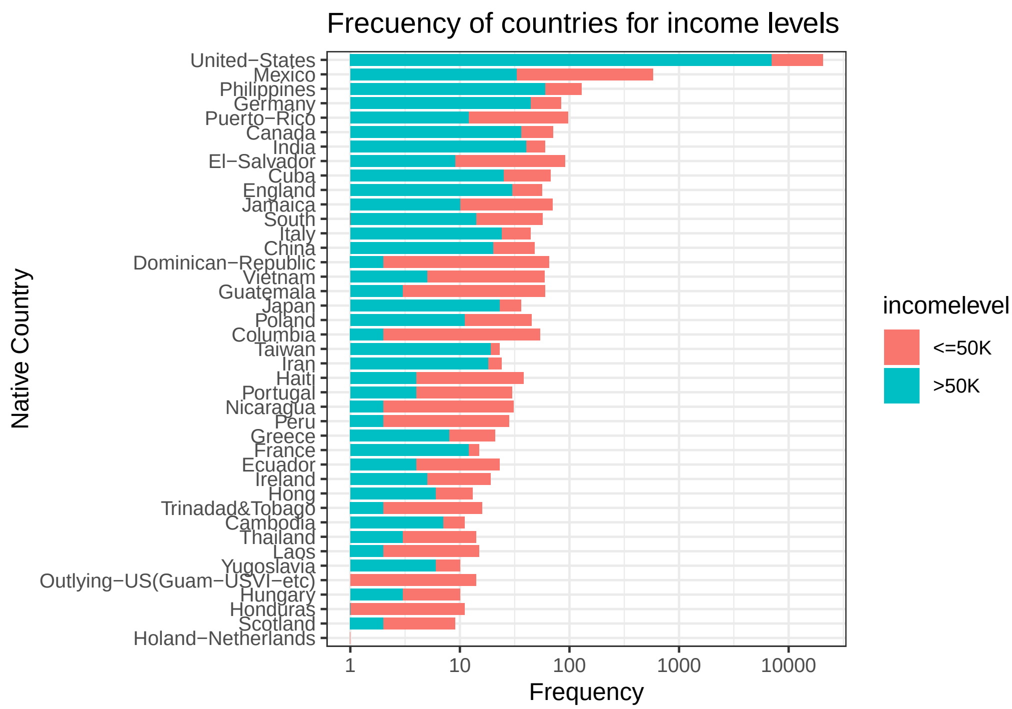
We can see that there is more frequency of people who works in a 40 hour job. And we can see that at every level, is more frequently to find people who earns less than 50k (**<=50K**)

```
adult %>%
  ggplot(aes(hoursperweek)) +
  geom_histogram(aes(fill=incomelevel), binwidth = 2.5) +
  scale_y_log10() + theme_bw() +
  xlab("Hours per Week") + ylab("Frequency") +
  ggtitle("Frecuency of hours per week for income levels")
```



If we want to see how much do people earn based on the country that they are from, we see that as the last plot, the behavior is very similar. At very level or country, we see that it is more common to see more people that earns less than 50k. (**<=50K**), but in the case of United States, we see that there is more people who earns more than 50k (**>50k**)

```
adult %>%
  ggplot(aes(x=reorder(nativecountry, nativecountry, function(x) length(x)))) +
  geom_bar(aes(fill=incomelevel), width = 0.8, position = "identity") +
  scale_y_log10() + theme_bw() +
  xlab("Native Country") + ylab("Frequency") +
  ggtitle("Frecuency of countries for income levels") +
  coord_flip()
```



2.3 Variables for modeling

After the initial data exploration, we want to select at least three variables for the income prediction.

So, for the predictions, we are going to use the next variables:

- age <int>
- education <Factor>
- occupation <Factor>
- race <Factor>
- sex <Factor>

2.4 Methology for modeling

We are going to use three models in this project, those are:

- SVM (Support Vector Machine)
- Decision Tree
- Random Forest

For evaluating those models, we are going to use four metrics, those are:

- Accuracy

$$\frac{\text{true positives} + \text{true negatives}}{\text{true positives} + \text{false negatives} + \text{true negatives} + \text{true negatives}}$$

- Sensitivity

$$\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

- Specificity

$$\frac{\text{true negatives}}{\text{true negatives} + \text{false positives}}$$

- F1 Score

$$2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

The summary of the results with all the metrics are going to be in the results section.

Chapter 3

Modeling

For this project, we have to divide `adult` database into `train_set` and `test_set`. `train_set` is used to create all the models and `test_set` is used to prove how nice those models works.

```
# Sample, train and test sets for the models
sample.adult <- sample.split(adult$incomelevel, SplitRatio = 0.80)
train_set = subset(adult, sample.adult == TRUE)
test_set = subset(adult, sample.adult == FALSE)
```

3.1 SVM (Support Vector Machine)

This is a supervised model known as Support Vector Machine.

This is a classification algorithm, with the objective of finding a hyperplane that separates data points of one class from those of another class. Basically, this works on the principle of a maximum marginal classifier.
Source: Math Works[1]

```
# Applying SVM Model
svm.adult = svm(incomelevel ~
                  age+education+occupation+race+sex,
```

[1] https://www.mathworks.com/discovery/support-vector-machine.html

```r
                data = train_set)
# Prediction of data and Confusion Matrix
test_set$pred.value = predict(svm.adult, newdata=test_set, type="response")
model1 <- table(test_set$income, test_set$pred.value)
confusionMatrix(model1)
```

```
## Confusion Matrix and Statistics
##
##
##         <=50K >50K
##   <=50K  4323  208
##   >50K   1029  473
##
##                Accuracy : 0.795
##                  95% CI : (0.7845, 0.8051)
##     No Information Rate : 0.8871
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3291
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8077
##             Specificity : 0.6946
##          Pos Pred Value : 0.9541
##          Neg Pred Value : 0.3149
##              Prevalence : 0.8871
##          Detection Rate : 0.7166
##    Detection Prevalence : 0.7510
##       Balanced Accuracy : 0.7512
##
##        'Positive' Class : <=50K
##
```

```r
F1_Score(test_set$income, test_set$pred.value)
```

```
## [1] 0.8748356
```

We add the results of this model to a data frame.

```
results<- data.frame(
  Model="SVM (Support Vector Machine)",
  Accuracy=
    Accuracy(test_set$income,
             test_set$pred.value),
  F1Score=
    F1_Score(test_set$income,
             test_set$pred.value),
  Sensitivity=
    sensitivity(test_set$income,
                test_set$pred.value),
  Specificity=
    specificity(test_set$income,
                test_set$pred.value))
results
```

```
##                          Model Accuracy   F1Score Sensitivity Specificity
## 1 SVM (Support Vector Machine) 0.794961 0.8748356   0.8077354   0.6945668
```

We can see that with this model we have a really good accuracy and a f1 score, but a little low specificity.

3.2 Decision Tree

This model that we are going to apply in this case is a one step decision tree. This model is harder to interpret but has an accuracy a little better than the linear regression. It goes thru the different variables to see which bracked it ends.
Source: Cran Project[2]

[2]https://cran.r-project.org/web/packages/OneR/index.html

```r
# Applying Decision Tree Model
detree <- rpart(incomelevel ~
                  age+education+occupation+race+sex,
                data = train_set)
# Prediction of data and Confusion Matrix
test_set$pred.value2 = predict(detree, newdata=test_set, type="class")
model2 <- table(test_set$income, test_set$pred.value2)
confusionMatrix(model2)
```

```
## Confusion Matrix and Statistics
##
##
##          <=50K >50K
##    <=50K  4311  220
##    >50K    974  528
##
##                Accuracy : 0.8021
##                  95% CI : (0.7918, 0.8121)
##     No Information Rate : 0.876
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3641
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8157
##             Specificity : 0.7059
##          Pos Pred Value : 0.9514
##          Neg Pred Value : 0.3515
##              Prevalence : 0.8760
##          Detection Rate : 0.7146
##    Detection Prevalence : 0.7510
##       Balanced Accuracy : 0.7608
##
##        'Positive' Class : <=50K
##
```

```r
F1_Score(test_set$income, test_set$pred.value2)
```

```
## [1] 0.8783619
```

```r
results<- bind_rows(
  results,
  data.frame(Model="Decision Tree",
             Accuracy=Accuracy(test_set$income,
                               test_set$pred.value2),
             F1Score=F1_Score(test_set$income,
                              test_set$pred.value2),
             Sensitivity=sensitivity(test_set$income,
                                     test_set$pred.value2),
             Specificity =specificity(test_set$income,
                                      test_set$pred.value2)))
results
```

```
##                           Model  Accuracy   F1Score Sensitivity Specificity
## 1 SVM (Support Vector Machine) 0.7949610 0.8748356   0.8077354   0.6945668
## 2                 Decision Tree 0.8020885 0.8783619   0.8157048   0.7058824
```

In this case, we see that our specificity improved, we can try another model to see how it behave.

3.3 Random Forest

This model consist of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest splits out a class prediction and the class with the most votes becomes our model's prediction. Source: Towards Data Science[3]

[3] https://towardsdatascience.com/understanding-random-forest-58381e0602d2

```r
set.seed(4543) # this is for reproducibility
# Applying Random Forest Model
rfmodel <- randomForest(incomelevel ~
                          age+education+occupation+race+sex,
                        data = train_set, importance = TRUE)
# Prediction of data and Confusion Matrix
test_set$pred.value3 = predict(rfmodel, newdata=test_set)
model3 <- table(test_set$income, test_set$pred.value3)
confusionMatrix(model3)
```

```
## Confusion Matrix and Statistics
##
##
##         <=50K >50K
##   <=50K  4187  344
##   >50K    817  685
##
##                Accuracy : 0.8076
##                  95% CI : (0.7974, 0.8174)
##     No Information Rate : 0.8294
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4249
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8367
##             Specificity : 0.6657
##          Pos Pred Value : 0.9241
##          Neg Pred Value : 0.4561
##              Prevalence : 0.8294
##          Detection Rate : 0.6940
##    Detection Prevalence : 0.7510
##       Balanced Accuracy : 0.7512
##
##        'Positive' Class : <=50K
##
```

```r
F1_Score(test_set$income, test_set$pred.value3)
```

```
## [1] 0.8782381
```

```r
results<- bind_rows(
  results,
  data.frame(Model="Random Forest",
             Accuracy=Accuracy(test_set$income,
                               test_set$pred.value3),
             F1Score=F1_Score(test_set$income,
                              test_set$pred.value3),
             Sensitivity=sensitivity(test_set$income,
                                     test_set$pred.value3),
             Specificity =specificity(test_set$income,
                                      test_set$pred.value3)))
results
```

```
##                           Model  Accuracy   F1Score Sensitivity Specificity
## 1 SVM (Support Vector Machine) 0.7949610 0.8748356   0.8077354   0.6945668
## 2                Decision Tree 0.8020885 0.8783619   0.8157048   0.7058824
## 3                Random Forest 0.8075584 0.8782381   0.8367306   0.6656948
```

For this final model, we see that the specificity is a little lower too, but a really nice accuracy and f1 Score.

Chapter 4

Results

This is a summary of the results of all the models that we did before.
All of these models were trained on `train_set` (**80%** of adult database) and validated with `test_set` (**20%** of adult database).

```
results
```

```
##                             Model  Accuracy   F1Score Sensitivity Specificity
## 1 SVM (Support Vector Machine) 0.7949610 0.8748356   0.8077354   0.6945668
## 2                Decision Tree 0.8020885 0.8783619   0.8157048   0.7058824
## 3                Random Forest 0.8075584 0.8782381   0.8367306   0.6656948
```

Decision Tree is the best model if we look at the `F1 Score` and the `specificity`.
But, **Random Forest** is the best if we look at the `accuracy` and the `sensitivity`.
In this case **SVM (Support Vector Machine)** had the lowest percentages in the indicators, being the worst among them.

Chapter 5

Conclusion

As a first step, we loaded de “Adult” or “Census+Income” database from the 1994. We split it into two parts, one for training (80%) and the other one for testing (20%).

After the exploration we proceed to model the algorithms.

Limitations

We actually used only three types of models, and this project can be used for a more rigorous machine learning project.

Future Work

As mentioned, this project can be used for a more rigorous machine learning project.
Other thing that can be done in the future is the database, this is from the 1994, this kind of investigations can be very useful for reports and education sources for machine learning and data analysis for timelines analysis and predictions.
As well, the Specificity and the Sensitivity can be prioritize one or another based on the type of policy we want to make. And, considering the consequences of making one type of error or another, we'll know which type of error is more severe or costly than making the other type of error. We can make clear and choose what type of error(type 1 or 2) based on which one have more significance and power for the test.