



## SIO-6051 Data Mining

# Final Project Adult Dataset

Presented to  
**MR. Hirotoshi Takeda**

By :

**Dominique Loyer  
Pierre-Yves BOSSAN  
Isabelle Rochette  
Mahmoud Mohanna  
Ratiaray Romeo Ravelonjanahary**

Due date: May 2nd, 2014

## Table of content:

Five phases of CRISPDM of the Adult Dataset .....	2
1-Executive summary .....	2
2- Business understanding .....	2
Project objectives .....	3
3- Data pre-processing and exploratory data analysis .....	4
3.1- Data cleaning .....	4
3.2- Elimination of outliers .....	6
4- Modeling section .....	7
Preparing and selecting the variables and applying the appropriate modeling techniques .....	7
4.1 – Training and validation .....	7
5- .....	11
5.1- classification trees .....	11
5.2 - k-nearest neighbor .....	17
5.3- neural networks .....	19
5.4- analyze the effect of varying parameters .....	21
6- Conclusions on the best model(s) and their absolute significance .....	23
Evaluating those models and determining whether they achieve the business objectives and answer the research questions .....	23
7- Reference .....	29
8-Appendices .....	30
Graphs and R commands .....	30



## Fives phases of CRISPDM of the Adult Dataset

### 1-Executive summary

#### 1.1- Business Understanding Phase

In this project, we analyze the Adult dataset provided from the Census bureau in USA. The objective is to identify the best algorithm to be employed in order to predict whether a certain adult has annual income more than 50,000 \$ based on several attributes such as: age, education, occupation, capital gain, and capital loss as well as other variables as illustrated in details within this document. Such prediction is required to decide if a certain adult could be granted a loan or be targeted by a specific marketing campaign.

#### 1.2- Data Understanding Phase

The provided adult dataset contains 16281 observations of 15 variables: 6 numeric variables and 9 factor level variables. Certain numeric variables contain outliers whereas certain factor variables contain missing data.

#### 1.3- Data Preparation Phase

Data preparation is done in order to clean and recover missing data and elimination of outliers. Missed data are recovered using the method of '*Replace Missing Values with Random Values*', while outliers are removed using '*Min-max*' technique.

#### 1.4- Modeling Phase

Three different models are applied on both training and validation datasets after data cleaning for the purpose of predicting the target variable and thereafter identifying the best algorithm:

1. K-nearest neighbour: for  $K=5$ .
2. Decision trees: three different trees based of different sets of variables.
3. Neural networks: applied on three training sets of 4070, 8140 and 12210 observations (25%, 50% and 75%) respectively for 4,8, and 10 hidden layers respectively.

#### 1.5- Evaluation Phase

The performance of each algorithm is initially evaluated for each model by getting the percentage between estimated values and actual values. False negative and false positive rates are also evaluated. With linear performance evaluation, we get the best performance 91.71% with neural network algorithm with 8 hidden layers and 4070 observations training set. The next best performance is obtained with a decision tree based on: Capital.gain, Capital.loss, Education.num, Marital.status variables. In such case, the recorded performance is 85.01%

## 2- Business understanding

### 2.1-Project objectives

The Adult dataset is from the Census bureau of United States (<https://www.census.gov/>) and the task is to predict whether a given adult makes more than 50 000\$ a year based on attributes such as : (See Figure 1 in the appendix for the actual distribution)

1. Age;
2. Workclass;
3. Final weight;
4. Education (High School, Bachelors, Masters, etc.);
5. Education.num (the level of education : e.g. High School 1st year, Masters 2 <sup>nd</sup> year, etc.);
6. Marital status;
7. Occupation;
8. Relationship;
9. Race;
10. Sex;
11. Capital gain;
12. Capital loss;
13. Hours of work per week;
14. Native country;
15. The income class that shows whether if the revenue is higher or lower than 50 000\$.

According to the authors of the study made in 1996, Ronny Kohavi and Barry Becker, the data were split into :

"train-test using MLC++ GenCVFiles (2/3, 1/3 random).| 48842 instances, mix of continuous and discrete (train=32561, test=16281) and 45222 if instances with unknown values are removed (train=30162, test=15060) "i

### 3- Data pre-processing and exploratory data analysis

Data preprocessing is taking place through two steps:

1. Data cleaning: for recovering missing values.
2. Elimination of outliers.

#### 3.1- Data cleaning

By examining the existing data of the adult dataset, we discover that there are missing data in three fields: workclass, occupation, and native.country. The missing data are marked as question marks '?'.

We employ the cleaning technique of Replacing Missing Values with Random Values. When applying this technique, we take in consideration the probability of occurrence for each variable in order to replace the missing values in order to keep the consistency of the dataset.

For each of the three variables containing missing data which are mentioned earlier, we go through the following steps:

1. Determination of the number of missed values for each of them
2. Transformation of the variable (the data field) into character instead of factor.
3. Selecting a random sample of dataset of other rows of the dataset which contain valid values. The probability of the presence of each value in the selected sample depends on its number of occurrence in the entire dataset.
4. Replacing the missed values, which are indicated by the question mark '?' by the random sample.
5. Returning the variable back into a factor variable instead of character.

Finally, we checked the resulting data by applying the command *unique* for each cleaned variable as well as viewing the summary of the entire dataset as shown in the figures below in order to insure the recovering of missing data before going through the next steps.

```

RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Project: (None)

Console ~ /
> unique(workclass)
[1] Private      Local-gov      Self-emp-not-inc Federal-gov      State-gov
[6] Self-emp-inc without-pay      Never-worked
8 Levels: Federal-gov Local-gov Never-worked Private Self-emp-inc ... without-pay
> unique(occupation)
[1] Machine-op-inspct Farming-fishing Protective-serv Exec-managerial
[5] other-service Prof-specialty Craft-repair Adm-clerical
[9] Tech-support Sales Handlers-cleaners Priv-house-serv
[13] Transport-moving Armed-Forces
14 Levels: Adm-clerical Armed-Forces Craft-repair Exec-managerial ... Transport-moving
> unique(native.country)
[1] United-States Peru Guatemala
[4] Mexico Dominican-Republic Ireland
[7] Germany Philippines Thailand
[10] Haiti India El-Salvador
[13] Puerto-Rico Vietnam South
[16] Columbia Japan Cambodia
[19] Poland Laos England
[22] Cuba Taiwan Italy
[25] Canada Portugal China
[28] Nicaragua Honduras Iran
[31] Scotland Jamaica Ecuador
[34] Yugoslavia Hungary Hong
[37] Greece Trinidad&Tobago Outlying-US(Guam-USVI-etc)
[40] France
40 Levels: Cambodia Canada China Columbia Cuba Dominican-Republic ... Yugoslavia
> |

```

```

RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Project: (None)

Console ~ /
> str(adult)
'data.frame': 16281 obs. of 15 variables:
 $ age      : num  0.1096 0.2877 0.1507 0.3699 0.0137 ...
 $ workclass : Factor w/ 8 levels "Federal-gov",...: 4 4 2 4 4 4 4 6 4 4 ...
 $ fnlwgt    : num  0.1444 0.0517 0.219 0.0994 0.0609 ...
 $ education : Factor w/ 16 levels "10th","11th",...: 2 12 8 16 16 1 12 15 16 6 ...
 $ education.num : num  0.4 0.533 0.733 0.6 0.6 ...
 $ marital.status : Factor w/ 7 levels "Divorced","Married-AF-spouse",...: 5 3 3 3 5 5 3 5 3 ...
 $ occupation : Factor w/ 14 levels "Adm-clerical",...: 7 5 11 7 1 8 11 10 8 3 ...
 $ relationship : Factor w/ 6 levels "Husband","Not-in-family",...: 4 1 1 1 4 2 5 1 5 1 ...
 $ race       : Factor w/ 5 levels "Amer-Indian-Eskimo",...: 3 5 5 3 5 5 3 5 5 5 ...
 $ sex        : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 2 2 2 1 2 ...
 $ capital.gain : num  0 0 0 0.0769 0 ...
 $ capital.loss : num  0 0 0 0 0 0 0 0 0 ...
 $ hours.per.week : num  0.398 0.5 0.398 0.398 0.296 ...
 $ native.country : Factor w/ 40 levels "Cambodia","Canada",...: 38 38 38 38 38 38 38 38 38 ...
 $ income.class : Factor w/ 2 levels "<=50K.", ">50K.": 1 1 2 2 1 1 1 2 1 1 ...
> |

```

## 3.2 - Elimination of outliers

Data transformation method was chosen in order to eliminate the outliers of the "adult" dataset. According to Daniel T. Larose, several techniques can be used depending on the objectives and the analysis. For this project, the min-max normalization was applied to certain variables on the dataset in order to achieve the normalization. The concerning variables are the numeric variables such as: age, fnlwgt, education.num, hours.per.week, capital.loss and capital.gain.

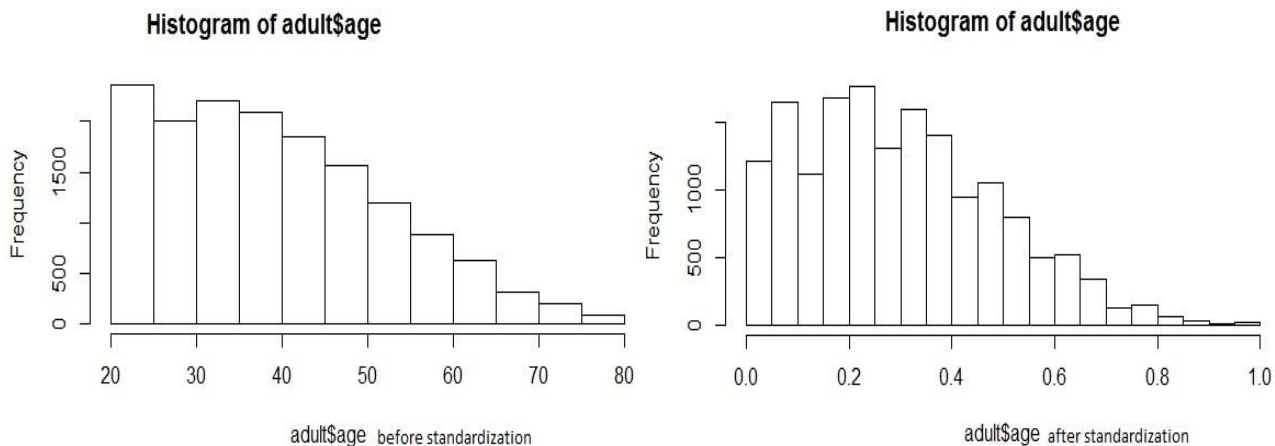
The process consisted of applying the min-max formula to selected variable and assigning the current result to the corresponding variable. In order to achieve the transformation, the algorithm below is applied:

```
adult$age=(adult$age-min(adult$age))/(max(adult$age)-min(adult$age))  
adult$fnlwgt=(adult$fnlwgt-min(adult$fnlwgt))/(max(adult$fnlwgt)-min(adult$fnlwgt))
```

```
adult$education.num=(adult$education.num-  
min(adult$education.num))/(max(adult$education.num)-min(adult$education.num))  
adult$hours.per.week=(adult$hours.per.week-  
min(adult$hours.per.week))/(max(adult$hours.per.week)-min(adult$hours.per.week))
```

```
adult$capital.loss=(adult$capital.loss-min(adult$capital.loss))/(max(adult$capital.loss)-  
min(adult$capital.loss))
```

```
adult$capital.gain=(adult$capital.gain-min(adult$capital.gain))/(max(adult$capital.gain)-  
min(adult$capital.gain))
```



Histogram of variable age before and after standardization

## 4- Modeling section

### Preparing and selecting the variables and applying the appropriate modeling techniques

#### 4.1 – Training and validation

##### 4.1.1 - Comparison of "mytreeadult" and "mytreeadult2"

```
all.equal(mytreeadult, mytreeadult2, use.edge.length = FALSE)
```

```
[1] "Attributes: < Component 2: Names: 5 string mismatches >"
[2] "Attributes: < Component 2: Length mismatch: comparison on first 5
components >"
[3] "Attributes: < Component 2: Component 1: Lengths (16, 8) differ
(string compare on first 8) >"
[4] "Attributes: < Component 2: Component 1: 8 string mismatches >"
[5] "Attributes: < Component 2: Component 2: Lengths (8, 16) differ
(string compare on first 8) >"
[6] "Attributes: < Component 2: Component 2: 8 string mismatches >"
[7] "Attributes: < Component 2: Component 3: Lengths (14, 7) differ
(string compare on first 7) >"
[8] "Attributes: < Component 2: Component 3: 7 string mismatches >"
[9] "Attributes: < Component 2: Component 4: Lengths (2, 14) differ
(string compare on first 2) >"
[10] "Attributes: < Component 2: Component 4: 2 string mismatches >"
[11] "Attributes: < Component 2: Component 5: Lengths (5, 6) differ (string
compare on first 5) >"
[12] "Attributes: < Component 2: Component 5: 5 string mismatches >"
[13] "Component 1: Attributes: < Component 2: Numeric: lengths (9, 13)
differ >"
[14] "Component 1: Component 1: Lengths: 9, 13"
[15] "Component 1: Component 1: Attributes: < Component 2: 4 string
mismatches >"
[16] "Component 1: Component 1: Lengths (9, 13) differ (string compare on
first 9)"
[17] "Component 1: Component 1: 7 string mismatches"
[18] "Component 1: Component 2: Numeric: lengths (9, 13) differ"
[19] "Component 1: Component 3: Numeric: lengths (9, 13) differ"
[20] "Component 1: Component 4: Numeric: lengths (9, 13) differ"
[21] "Component 1: Component 5: Numeric: lengths (9, 13) differ"
[22] "Component 1: Component 6: Numeric: lengths (9, 13) differ"
[23] "Component 1: Component 7: Numeric: lengths (9, 13) differ"
[24] "Component 1: Component 8: Numeric: lengths (9, 13) differ"
[25] "Component 1: Component 9: Attributes: < Component 1: Mean relative
difference: 0.4444444 >"
[26] "Component 1: Component 9: Numeric: lengths (54, 78) differ"
[27] "Component 2: Mean relative difference: 1.183825"
[28] "Component 3: target, current do not match when deparsed"
[29] "Component 4: formulas differ in contents"
[30] "Component 5: Attributes: < Component 1: Mean relative difference:
0.6666667 >"
[31] "Component 5: Numeric: lengths (15, 25) differ"
[32] "Component 8: Component 1: Mean relative difference: 19"
[33] "Component 8: Component 2: Mean absolute difference: 7"
[34] "Component 11: Attributes: < Component 1: Mean relative difference:
0.9565217 >"
[35] "Component 11: Attributes: < Component 2: Component 1: 23 string
mismatches >"
[36] "Component 11: Numeric: lengths (115, 225) differ"
```



```
[37] "Component 12: Attributes: < Component 1: Mean relative difference:
0.7142857 >"
[38] "Component 12: Numeric: lengths (304, 800) differ"
[39] "Component 13: Names: 5 string mismatches"
[40] "Component 13: Numeric: lengths (5, 12) differ"
[41] "Component 15: Names: 5 string mismatches"
[42] "Component 15: Lengths (6, 14) differ (comparison on first 6
components)"
```

#### 4.1.2 - Comparison of "mytreeadult" and "mytreeadult3"

```
> all.equal(mytreeadult, mytreeadult3, use.edge.length=FALSE)
```

```
[1] "Attributes: < Component 2: Names: 3 string mismatches >"
[2] "Attributes: < Component 2: Length mismatch: comparison on first 4
components >"
[3] "Attributes: < Component 2: Component 1: Lengths (16, 8) differ
(string compare on first 8) >"
[4] "Attributes: < Component 2: Component 1: 8 string mismatches >"
[5] "Attributes: < Component 2: Component 2: Lengths (8, 7) differ (string
compare on first 7) >"
[6] "Attributes: < Component 2: Component 2: 7 string mismatches >"
[7] "Attributes: < Component 2: Component 3: Lengths (14, 5) differ
(string compare on first 5) >"
[8] "Attributes: < Component 2: Component 3: 5 string mismatches >"
[9] "Component 1: Attributes: < Component 2: Numeric: lengths (9, 13)
differ >"
[10] "Component 1: Component 1: Lengths: 9, 13"
[11] "Component 1: Component 1: Attributes: < Component 2: 4 string
mismatches >"
[12] "Component 1: Component 1: Lengths (9, 13) differ (string compare on
first 9)"
[13] "Component 1: Component 1: 7 string mismatches"
[14] "Component 1: Component 2: Numeric: lengths (9, 13) differ"
[15] "Component 1: Component 3: Numeric: lengths (9, 13) differ"
[16] "Component 1: Component 4: Numeric: lengths (9, 13) differ"
[17] "Component 1: Component 5: Numeric: lengths (9, 13) differ"
[18] "Component 1: Component 6: Numeric: lengths (9, 13) differ"
[19] "Component 1: Component 7: Numeric: lengths (9, 13) differ"
[20] "Component 1: Component 8: Numeric: lengths (9, 13) differ"
[21] "Component 1: Component 9: Attributes: < Component 1: Mean relative
difference: 0.4444444 >"
[22] "Component 1: Component 9: Numeric: lengths (54, 78) differ"
[23] "Component 2: Mean relative difference: 1.19961"
[24] "Component 3: target, current do not match when deparsed"
[25] "Component 4: formulas differ in contents"
[26] "Component 5: Attributes: < Component 1: Mean relative difference:
0.6666667 >"
[27] "Component 5: Numeric: lengths (15, 25) differ"
[28] "Component 8: Component 1: Mean relative difference: 19"
[29] "Component 8: Component 2: Mean absolute difference: 7"
[30] "Component 11: Attributes: < Component 1: Mean relative difference:
0.826087 >"
[31] "Component 11: Attributes: < Component 2: Component 1: 21 string
mismatches >"
[32] "Component 11: Numeric: lengths (115, 210) differ"
[33] "Component 12: Attributes: < Component 1: Mean relative difference:
0.5428571 >"
[34] "Component 12: Numeric: lengths (304, 64) differ"
[35] "Component 13: Names: 4 string mismatches"
[36] "Component 13: Numeric: lengths (5, 9) differ"
[37] "Component 15: Names: 5 string mismatches"
[38] "Component 15: Lengths (6, 9) differ (comparison on first 6
components)"
```

### 4.1.3 - Comparison of "mytreeadult2" and "mytreeadult3"

```
> all.equal(mytreeadult2, mytreeadult3, dist.edge.length=FALSE)

[1] "Attributes: < Component 2: Names: 3 string mismatches >"
[2] "Attributes: < Component 2: Length mismatch: comparison on first 4
components >"
[3] "Attributes: < Component 2: Component 2: Lengths (16, 7) differ
(string compare on first 7) >"
[4] "Attributes: < Component 2: Component 2: 7 string mismatches >"
[5] "Attributes: < Component 2: Component 3: Lengths (7, 5) differ (string
compare on first 5) >"
[6] "Attributes: < Component 2: Component 3: 5 string mismatches >"
[7] "Attributes: < Component 2: Component 4: Lengths (14, 2) differ
(string compare on first 2) >"
[8] "Attributes: < Component 2: Component 4: 2 string mismatches >"
[9] "Component 1: Component 1: Attributes: < Component 2: 2 string
mismatches >"
[10] "Component 1: Component 1: 3 string mismatches"
[11] "Component 1: Component 2: Mean relative difference: 0.01844142"
[12] "Component 1: Component 3: Mean relative difference: 0.01844142"
[13] "Component 1: Component 4: Mean relative difference: 0.009607994"
[14] "Component 1: Component 6: Mean relative difference: 0.0310192"
[15] "Component 1: Component 8: Mean relative difference: 0.3"
[16] "Component 1: Component 9: Mean relative difference: 0.0184811"
[17] "Component 2: Mean relative difference: 2.324051"
[18] "Component 3: target, current do not match when deparsed"
[19] "Component 4: formulas differ in contents"
[20] "Component 5: Mean relative difference: 0.002749724"
[21] "Component 11: Attributes: < Component 1: Mean relative difference:
0.06666667 >"
[22] "Component 11: Attributes: < Component 2: Component 1: 41 string
mismatches >"
[23] "Component 11: Numeric: lengths (225, 210) differ"
[24] "Component 12: Attributes: < Component 1: Mean relative difference:
0.7333333 >"
[25] "Component 12: Numeric: lengths (800, 64) differ"
[26] "Component 13: Names: 9 string mismatches"
[27] "Component 13: Numeric: lengths (12, 9) differ"
[28] "Component 15: Names: 7 string mismatches"
[29] "Component 15: Lengths (14, 9) differ (comparison on first 9
components)"
```

```
> printcp(mytreeadult)
```

```
Classification tree:
rpart(formula = income.class ~ education + workclass + occupation +
sex + age + race, data = adult, method = "class", control =
rpart.control(minsplit = 1))
```

```
Variables actually used in tree construction:
[1] age      education occupation sex
```

```
Root node error: 3846/16281 = 0.23623
```

```
n= 16281
```

	CP	nsplit	rel	error	xerror	xstd
1	0.047322	0	1.00000	1.00000	0.014092	
2	0.022361	3	0.83411	0.83801	0.013220	
3	0.010000	4	0.81175	0.81955	0.013109	

```
> printcp(mytreeadult2)
```

```
Classification tree:
```

```
rpart(formula = income.class ~ ., data = adult, method = "class")
```

Variables actually used in tree construction:

```
[1] capital.gain capital.loss education relationship
```

Root node error: 3846/16281 = 0.23623

n= 16281

	CP	nsplit	rel	error	xerror	xstd
1	0.125715	0	1.00000	1.00000	0.014092	
2	0.054602	2	0.74857	0.74857	0.012658	
3	0.037181	3	0.69397	0.69397	0.012282	
4	0.010660	4	0.65679	0.65679	0.012011	
5	0.010000	6	0.63547	0.64587	0.011929	

```
> printcp(mytreeadult3)
```

Classification tree:

```
rpart(formula = income.class ~ age + workclass + education.num +  
marital.status + race + sex + capital.gain + capital.loss +  
hours.per.week, data = adult, method = "class")
```

Variables actually used in tree construction:

```
[1] capital.gain capital.loss education.num marital.status
```

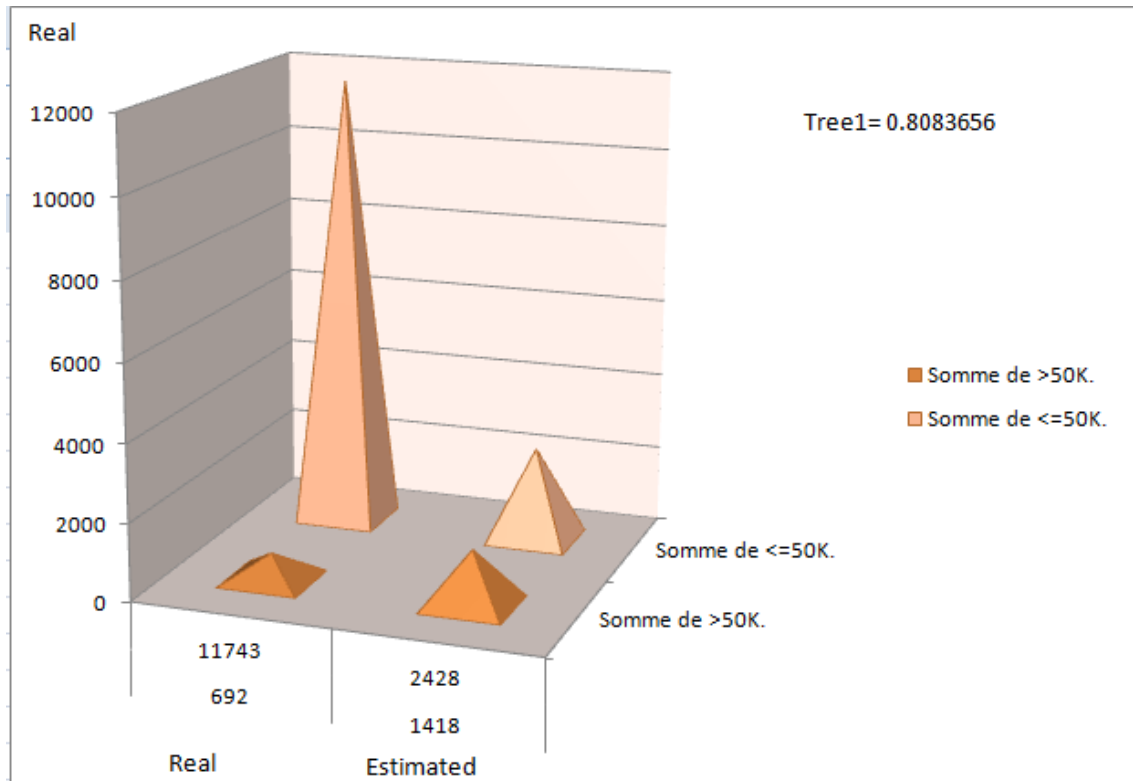
Root node error: 3846/16281 = 0.23623

n= 16281

	CP	nsplit	rel	error	xerror	xstd
1	0.126365	0	1.00000	1.00000	0.014092	
2	0.054862	2	0.74727	0.74805	0.012654	
3	0.036661	3	0.69241	0.69319	0.012277	
4	0.010660	4	0.65575	0.65627	0.012008	
5	0.010000	6	0.63443	0.63833	0.011872	

## 5- Algorithms

### 5.1- classification trees



```
>mytreeadult=rpart(income.class~education+workclass+occupation+sex+age+race
, data=adult, method="class", control=rpart.control(minsplit=1))
```

```
> mytreeadult
```

```
n= 16281
```

```
node), split, n, loss, yval, (yprob)
* denotes terminal node
```

```
1) root 16281 3846 <=50K. (0.7637737 0.2362263)
 2) education= 10th, 11th, 12th, 1st-4th, 5th-6th, 7th-8th, 9th, Assoc-
acdm, Assoc-voc, HS-grad, Preschool, Some-college 12238 1935 <=50K.
(0.8418859 0.1581141) *
 3) education= Bachelors, Doctorate, Masters, Prof-school 4043 1911
<=50K. (0.5273312 0.4726688)
 6) age< 0.1712329 795 105 <=50K. (0.8679245 0.1320755) *
 7) age>=0.1712329 3248 1442 >50K. (0.4439655 0.5560345)
 14) sex= Female 876 301 <=50K. (0.6563927 0.3436073) *
 15) sex= Male 2372 867 >50K. (0.3655143 0.6344857)
 30) occupation= Craft-repair, Farming-fishing, Handlers-cleaners,
Machine-op-inspct, Other-service, Transport-moving 262 87 <=50K.
(0.6679389 0.3320611) *
 31) occupation= Adm-clerical, Armed-Forces, Exec-managerial, Priv-
house-serv, Prof-specialty, Protective-serv, Sales, Tech-support 2110 692
>50K. (0.3279621 0.6720379) *
```

```
> plot(mytreeadult)
```

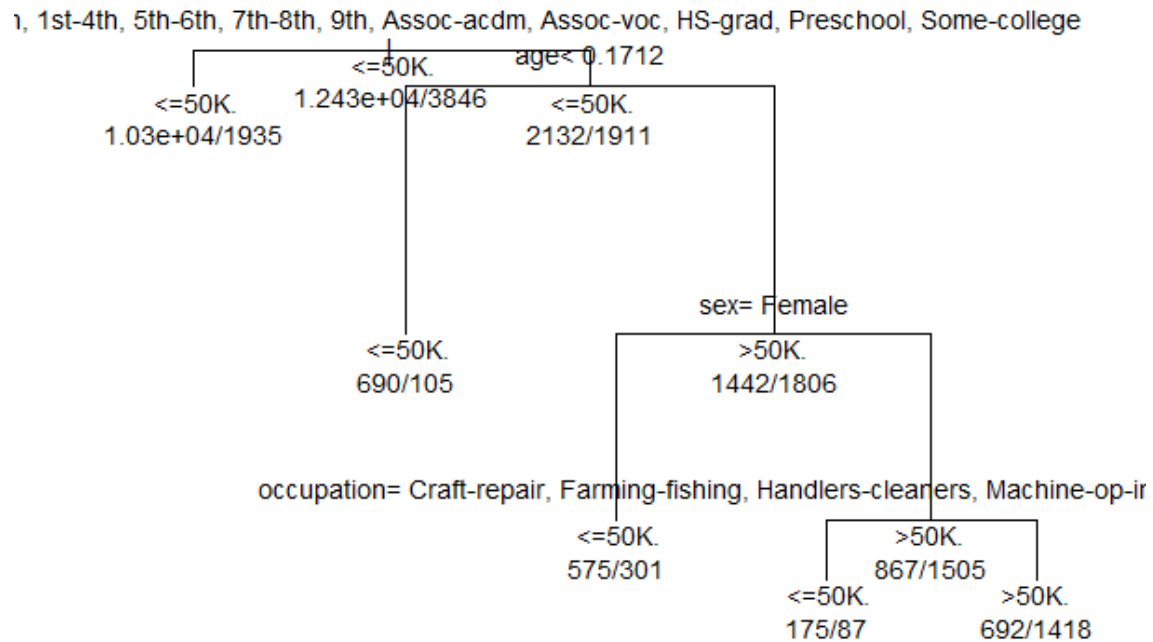
```
> text(mytreeadult, use.n=T, all=T, pretty=0, cex=0.9, xpd=TRUE)
```

```
> estincome.class=predict(mytreeadult, data=adult, type="class")
```

```
> t1=table(income.class, estincome.class)
```

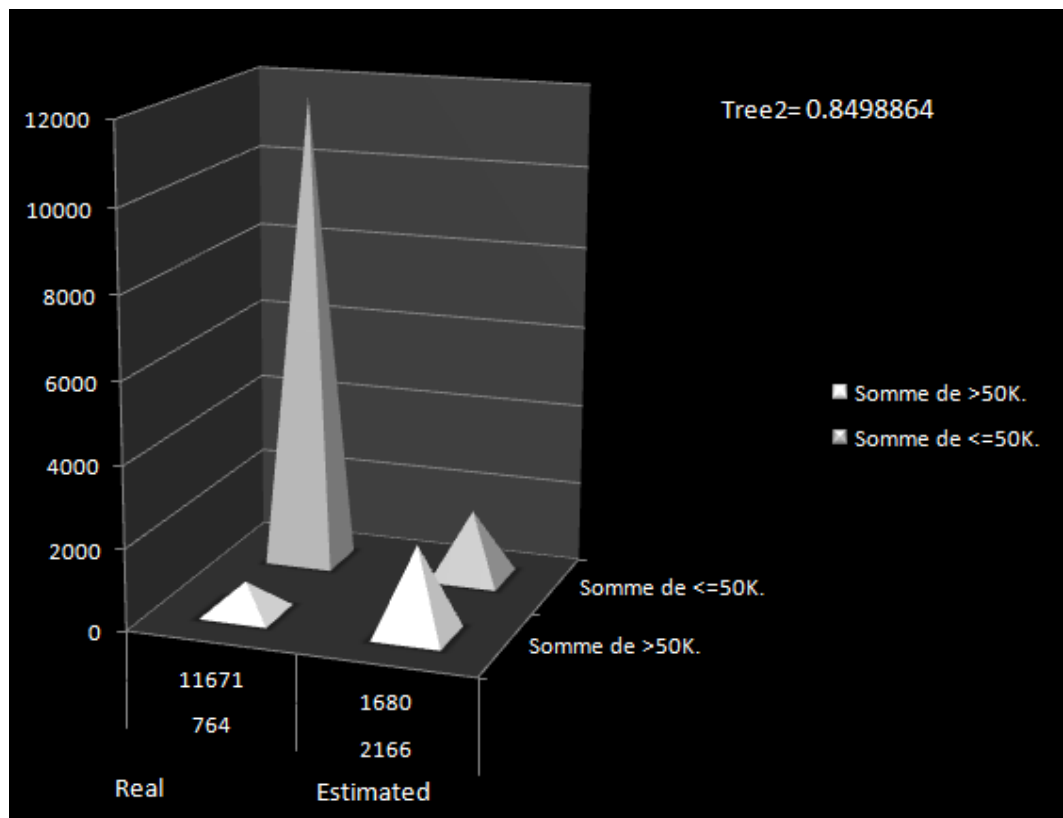
```
> t1
      estincome.class
income.class  <=50K.  >50K.
  <=50K.      11743    692
  >50K.       2428   1418
```

```
> (11743+1418)/16281
```



**Mytreeadult1**

```
[1] 0.8083656
```



```
> mytreeadult2=rpart(income.class~., data=adult, method="class")
```

```
> mytreeadult2
```

```
n= 16281
```

```
node), split, n, loss, yval, (yprob)
```

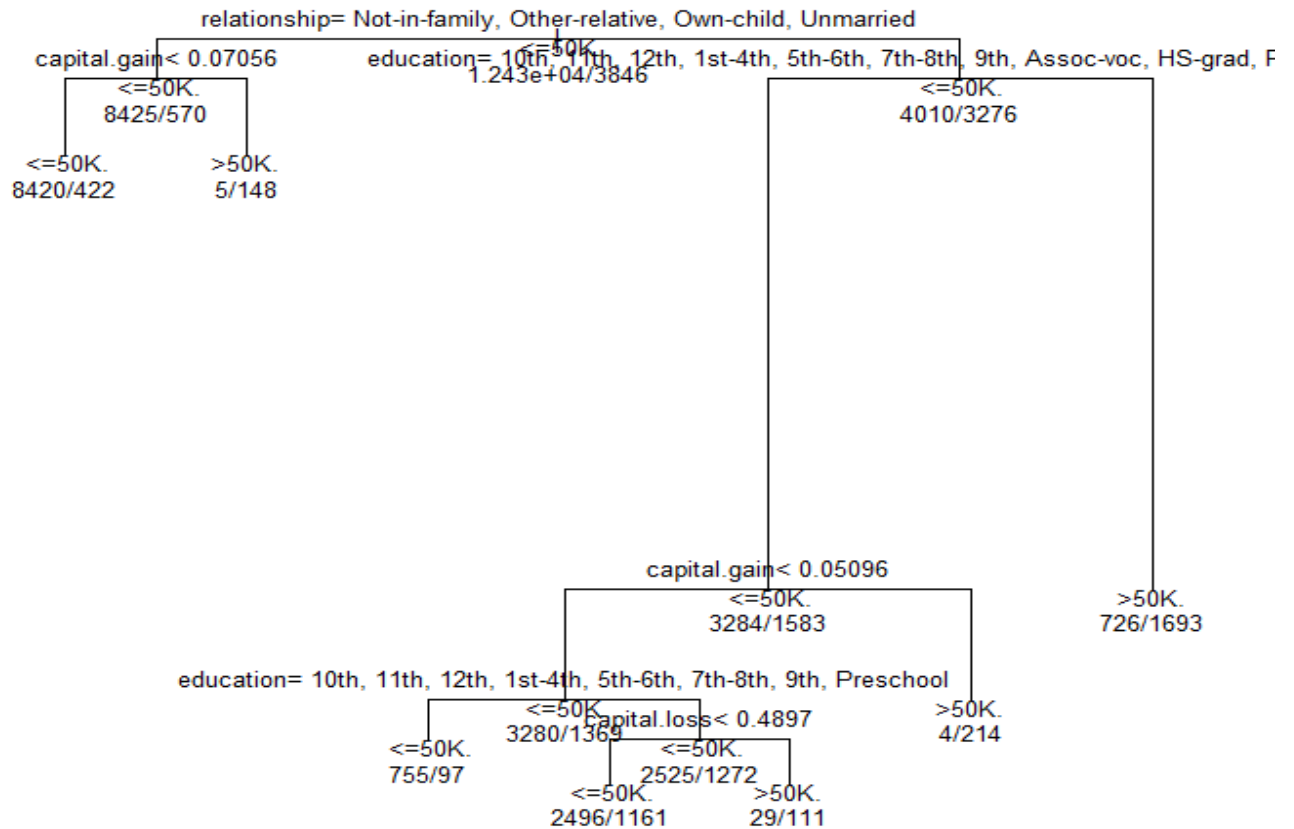
```
* denotes terminal node
```

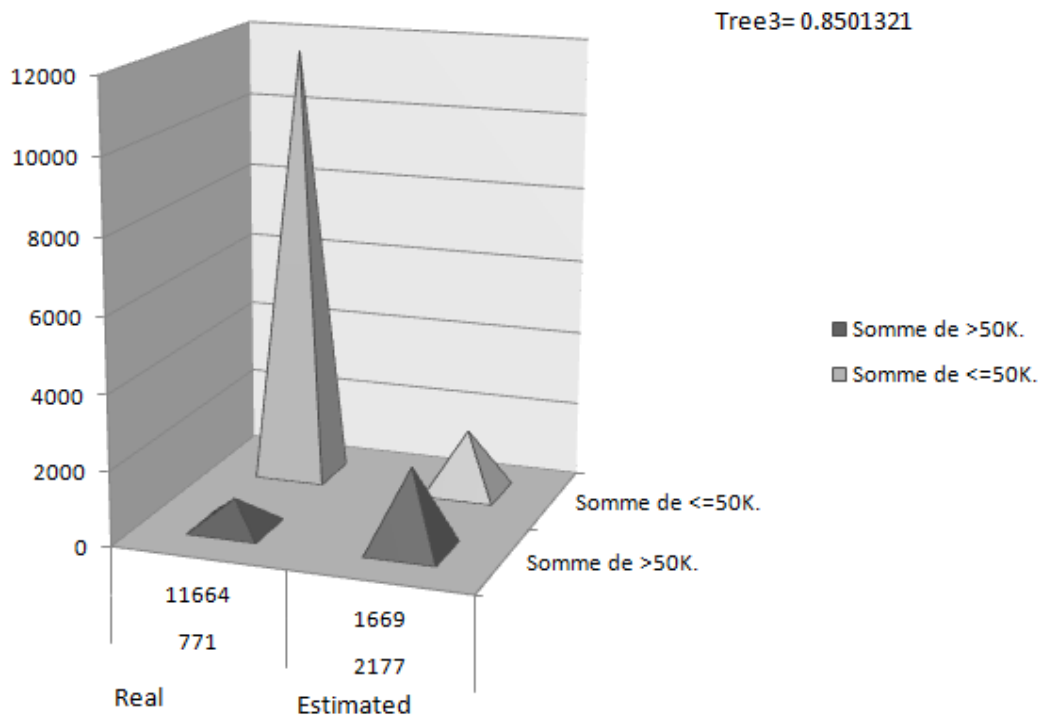
```
1) root 16281 3846 <=50K. (0.76377372 0.23622628)
  2) relationship= Not-in-family, Other-relative, Own-child, Unmarried
    8995 570 <=50K. (0.93663146 0.06336854)
      4) capital.gain< 0.07055571 8842 422 <=50K. (0.95227324 0.04772676)
        *
          5) capital.gain>=0.07055571 153 5 >50K. (0.03267974 0.96732026) *
            3) relationship= Husband, wife 7286 3276 <=50K. (0.55037057 0.44962943)
              6) education= 10th, 11th, 12th, 1st-4th, 5th-6th, 7th-8th, 9th, Assoc-
                voc, HS-grad, Preschool, Some-college 4867 1583 <=50K. (0.67474830
                  0.32525170)
                  12) capital.gain< 0.05095551 4649 1369 <=50K. (0.70552807
                    0.29447193)
                    24) education= 10th, 11th, 12th, 1st-4th, 5th-6th, 7th-8th, 9th,
                      Preschool 852 97 <=50K. (0.88615023 0.11384977) *
                      25) education= Assoc-voc, HS-grad, Some-college 3797 1272 <=50K.
                        (0.66499868 0.33500132)
                        50) capital.loss< 0.4896552 3657 1161 <=50K. (0.68252666
                          0.31747334) *
                          51) capital.loss>=0.4896552 140 29 >50K. (0.20714286
                            0.79285714) *
                            13) capital.gain>=0.05095551 218 4 >50K. (0.01834862 0.98165138)
                              *
                                7) education= Assoc-acdm, Bachelors, Doctorate, Masters, Prof-school
                                  2419 726 >50K. (0.30012402 0.69987598) *
```

```
> plot(mytreeadult2)
> text(mytreeadult2, use.n=T, all=T, pretty=0, cex=0.8, xpd=TRUE)
> estincome.class=predict(mytreeadult2, data=adult, type="class")
> t1=table(income.class, estincome.class)
```

```
> t1
      estincome.class
income.class <=50K. >50K.
  <=50K.      11671    764
  >50K.       1680   2166
```

```
> (11671+2166)/16281
[1] 0.8498864
```





```
>mytreeadult3=rpart(income.class~age+workclass+education.num+marital.status
+race+sex+capital.gain+capital.loss+hours.per.week, data=adult,
method="class")
```

## Mytreeadult2

```
> mytreeadult3
```

```
n= 16281
```

```
node), split, n, loss, yval, (yprob)
* denotes terminal node
```

```
1) root 16281 3846 <=50K. (0.76377372 0.23622628)
  2) marital.status= Divorced, Married-spouse-absent, Never-married,
Separated, widowed 8864 550 <=50K. (0.93795126 0.06204874)
    4) capital.gain< 0.07055571 8713 404 <=50K. (0.95363250 0.04636750)
      *
        5) capital.gain>=0.07055571 151 5 >50K. (0.03311258 0.96688742) *
      3) marital.status= Married-AF-spouse, Married-civ-spouse 7417 3296
<=50K. (0.55561548 0.44438452)
        6) education.num< 0.7 4983 1593 <=50K. (0.68031306 0.31968694)
          12) capital.gain< 0.05095551 4764 1378 <=50K. (0.71074727
0.28925273)
            24) education.num< 0.5 876 97 <=50K. (0.88926941 0.11073059) *
            25) education.num>=0.5 3888 1281 <=50K. (0.67052469 0.32947531)
              50) capital.loss< 0.4896552 3744 1168 <=50K. (0.68803419
0.31196581) *
              51) capital.loss>=0.4896552 144 31 >50K. (0.21527778
0.78472222) *
                13) capital.gain>=0.05095551 219 4 >50K. (0.01826484 0.98173516)
                  *
                    7) education.num>=0.7 2434 731 >50K. (0.30032868 0.69967132) *
```

```
> estincome.class3=predict(mytreeadult3, data=adult, type="class")
```



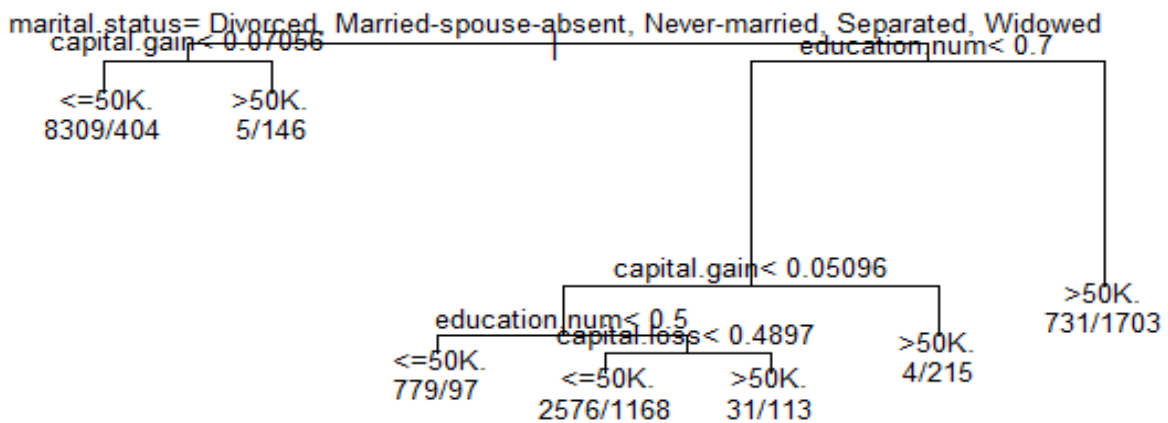
```
> t3=table(income.class, estincome.class3)

> t3
      estincome.class3
income.class  <=50K.  >50K.
  <=50K.      11664   771
  >50K.       1669  2177

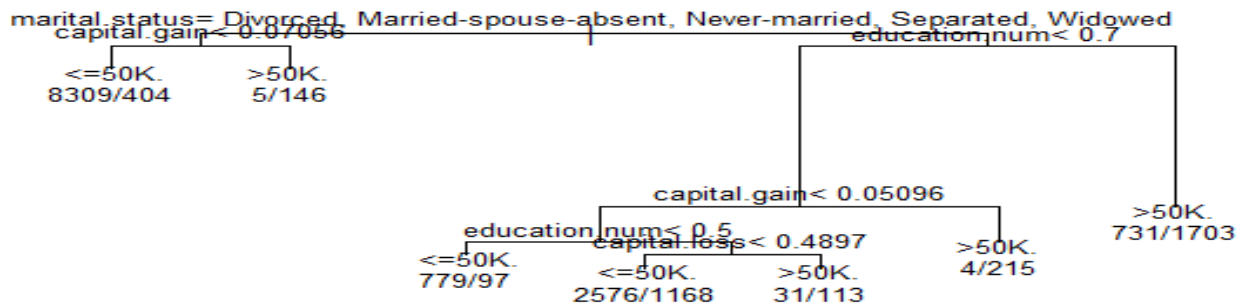
> (11664+2177)/16281
[1] 0.8501321

> plot(mytreeadult3)

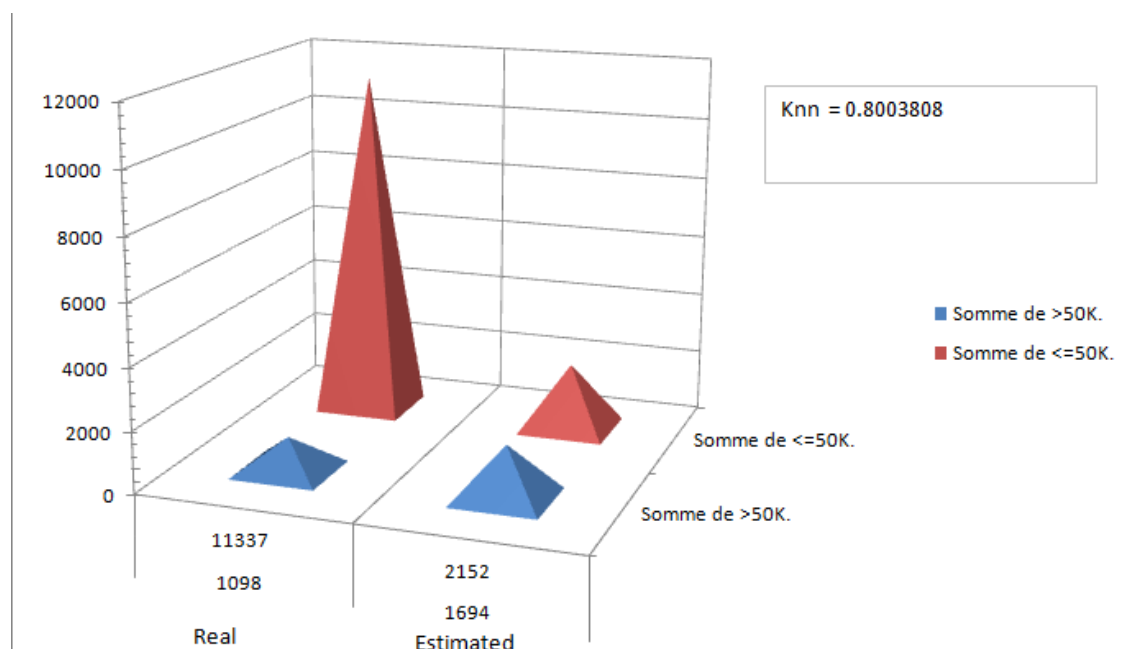
> text(mytreeadult3, use.n=T, all=F, pretty=0, cex=0.8, xpd=TRUE)
```



### Mvtreeadult3



## 5.2 - k-nearest neighbor



### KNN on variable "income.class"

Training set at 60 %, 9700 variables out of 16281

```
str(adult)

adult$workclass=NULL
adult$education=NULL
adult$marital.status=NULL
adult$occupation=NULL
adult$relationship=NULL
adult$race=NULL
adult$sex=NULL
adult$native.country=NULL

adult$income.class=as.factor(income.class)
adult$income.class=as.factor(adult$income.class)
str(adult)
estincome.class=knn.cv(adult[,-7], adult[,7], 5)
estincome.class

> t1=table(income.class, estincome.class)

> str(adult)

'data.frame': 16281 obs. of 7 variables:
 $ age      : num  0.1096 0.2877 0.1507 0.3699 0.0137 ...
```

```
$ fnlwgt      : num  0.1444 0.0517 0.219 0.0994 0.0609 ...
$ education.num : num  0.4 0.533 0.733 0.6 0.6 ...
$ capital.gain  : num  0 0 0 0.0769 0 ...
$ capital.loss  : num  0 0 0 0 0 0 0 0 0 ...
$ hours.per.week: num  0.398 0.5 0.398 0.398 0.296 ...
$ income.class  : Factor w/ 2 levels "<=50K.," >50K.": 1 1 2 2 1 1 1 2 1
1 ...
```

```
> t1
```

```
      estincome.class
income.class <=50K. >50K.
      <=50K.  11337  1098
      >50K.   2152  1694
```

```
> (11337+1694)/16281
```

```
[1] 0.8003808
```

```
> indtrain=sample(1:16281, 9700)
```

```
> ls.str()
```

```
adult : 'data.frame': 16281 obs. of 7 variables:
 $ age      : num  0.1096 0.2877 0.1507 0.3699 0.0137 ...
 $ fnlwgt    : num  0.1444 0.0517 0.219 0.0994 0.0609 ...
 $ education.num : num  0.4 0.533 0.733 0.6 0.6 ...
 $ capital.gain : num  0 0 0 0.0769 0 ...
 $ capital.loss : num  0 0 0 0 0 0 0 0 0 ...
 $ hours.per.week: num  0.398 0.5 0.398 0.398 0.296 ...
 $ income.class : Factor w/ 2 levels "<=50K.," >50K.": 1 1 2 2 1 1 1 2 1
1 ...
estincome.class : Factor w/ 2 levels "<=50K.," >50K.": 1 1 1 2 1 1 1 2 1
1 ...
income.class : Factor w/ 2 levels "<=50K.," >50K.": 1 1 2 2 1 1 1 2 1 1
...
indtrain : int [1:9700] 6203 15486 6667 13874 10072 9913 3408 3915 15131
4107 ...
native.country : Factor w/ 40 levels "Cambodia","Canada",...: 38 38 38 38
38 38 38 38 38 ...
occupation : Factor w/ 14 levels "Adm-clerical",...: 7 5 11 7 4 8 7 10 8 3
...
t1 : 'table' int [1:2, 1:2] 11337 2152 1098 1694
test : 'data.frame': 6581 obs. of 7 variables:
 $ age      : num  0.1507 0.3699 0.0137 0.2329 0.0959 ...
 $ fnlwgt    : num  0.219 0.0994 0.0609 0.1254 0.2412 ...
 $ education.num : num  0.733 0.6 0.6 0.333 0.6 ...
 $ capital.gain : num  0 0.0769 0 0 0 ...
 $ capital.loss : num  0 0 0 0 0 0 0 0 0 ...
 $ hours.per.week: num  0.398 0.398 0.296 0.296 0.398 ...
 $ income.class : Factor w/ 2 levels "<=50K.," >50K.": 2 2 1 1 1 2 1 2 2
1 ...
train : 'data.frame': 9700 obs. of 7 variables:
 $ age      : num  0.1507 0.7945 0.0822 0.274 0.2192 ...
 $ fnlwgt    : num  0.1532 0.0588 0.0722 0.2525 0.1292 ...
 $ education.num : num  0.533 0.333 0.533 0.133 0.867 ...
 $ capital.gain : num  0 0 0 0 0 0 0 0 0 ...
 $ capital.loss : num  0 0 0 0 0 0 0 0 0 ...
 $ hours.per.week: num  0.398 0.143 0.48 0.449 0.378 ...
 $ income.class : Factor w/ 2 levels "<=50K.," >50K.": 1 1 1 1 1 2 1 1 1
1 ...
workclass : Factor w/ 8 levels "Federal-gov",...: 4 4 2 4 4 4 6 4 4 ...
```

```
> realincome.class=adult[indtrain, 7]
```

```
> realtest=adult[-indtrain, 7]
```

```
> estincome.class=knn(train, train, realincome.class, 5)
```

```
> train=adult[indtrain, 1:6]
```

```

> test=adult[-indtrain, 1:6]
> estincome.class=knn(train, train, realincome.class, 5)
> t2=table(estincome.class, realincome.class)
> t2
      realincome.class
estincome.class <=50K. >50K.
      <=50K.    6975  1010
      >50K.     400  1315

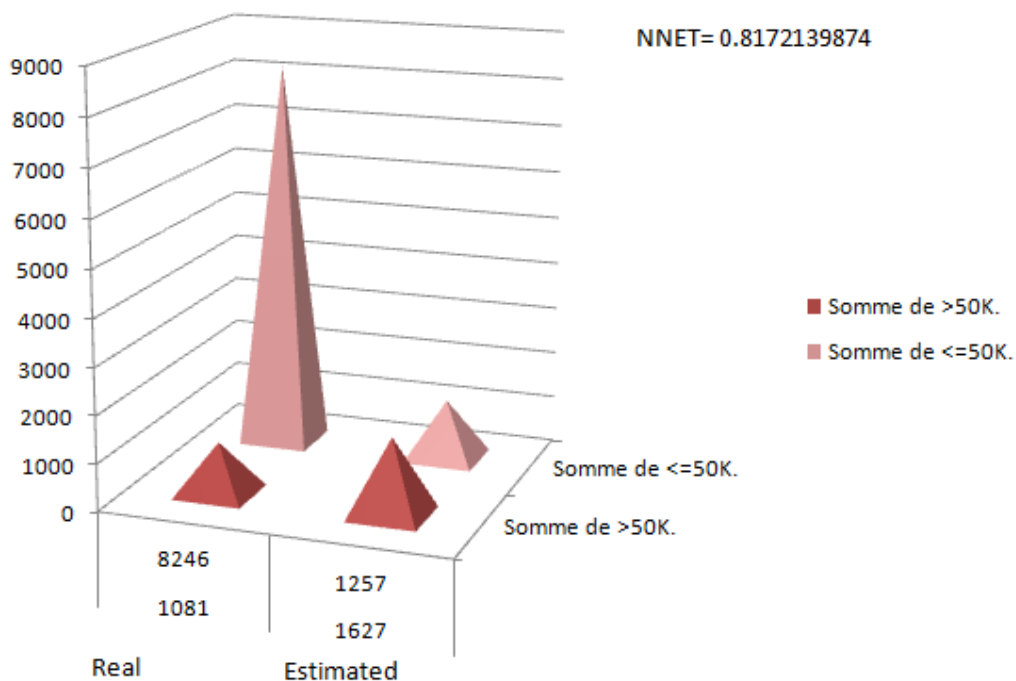
> (6975+1315)/9700
[1] 0.8546392

> esttest=knn(train, test, realincome.class, 5)
> t3=table(esttest, realtest)
> t3
      realtest
esttest <=50K. >50K.
      <=50K.  4604  875
      >50K.   456  646

> (4604+646)/9700
[1] 0.5412371

```

### 5.3- neural networks



We apply the neural network on variable "income.class" of the adult dataset.

After inputting the dataset in R, we prepare a training set of 9700 observations (60%). The rest will constitute the validation set.

```
> set.seed(99999999)
> index <- sample(1:nrow(adult), 9700)
> adult.test = adult[index,]
> adult.valid = adult[-index,]
```

Then we set up our neural network with 10 for the size of the hidden layer:

```
> adult.net = nnet(income.class~.,data=adult.test,size=10)
# weights: 981
initial value 10111.087684
iter 10 value 3505.577504
iter 20 value 3112.728652
iter 30 value 2909.482290
iter 40 value 2805.980720
iter 50 value 2750.823741
iter 60 value 2686.060553
iter 70 value 2633.928168
iter 80 value 2592.264548
iter 90 value 2517.967758
iter 100 value 2456.290412
final value 2456.290412
stopped after 100 iterations
```

Then we try our model on the validation set and use a cross validation table to assess the performance:

```
adult.valid$est.income = predict(adult.net,adult.valid,type="class")
> T=table(adult.valid$income.class,adult.valid$est.income)
> T
```

	<=50K.	>50K.
<=50K.	4650	395
>50K.	656	880

```
> (4650+880)/(16281-9700)
[1] 0.8402978
```

We get a pretty good 84.03% correct classification.

Now we do the same again with a smaller hidden layer (5):

```
> adult.net = nnet(income.class~.,data=adult.test,size=5)
# weights: 491
initial value 5853.263152
iter 10 value 3455.243801
iter 20 value 3154.626569
iter 30 value 3005.342318
iter 40 value 2935.386984
iter 50 value 2886.353716
iter 60 value 2824.031681
iter 70 value 2774.114975
iter 80 value 2745.270426
iter 90 value 2728.753792
iter 100 value 2713.742806
final value 2713.742806
stopped after 100 iterations
> adult.valid$est.income =
predict(adult.net,adult.valid,type="class")
> T=table(adult.valid$income.class,adult.valid$est.income)
> T
```

```

      <=50K. >50K.
<=50K.    4633   412
>50K.      635   901
> (4633+901)/(16281-9700)
[1] 0.8409056

```

The result is slightly better with an 84.09% correct classification.

#### 5.4- analyze the effect of varying parameters

The performance of each algorithm is initially evaluated for each model by getting the percentage between estimated values and actual values. False negative and false positive rates are also evaluated. With linear performance evaluation, we get the best performance 91.71% with neural network algorithm with 8 hidden layers and 4070 observations training set. The next best performance is obtained with a decision tree based on: Capital.gain, Capital.loss, Education.num, Marital.status variables. In such case, the recorded performance is 85.01%

#### Algorithme KNN sur la variable "income.class"

**Training set is à 60 %, 9700 variables on 16281**

**Almost same results with 25%**

```

> t2=table(estincome.class, realincome.class)
> t2
      realincome.class
estincome.class <=50K. >50K.
      <=50K.    2910   443
      >50K.     191   526
> (2910+526)/4070
[1] 0.844226

```

```
str(adult)
```

```

adult$workclass=NULL
adult$education=NULL
adult$marital.status=NULL
adult$occupation=NULL
adult$relationship=NULL
adult$race=NULL
adult$sex=NULL
adult$native.country=NULL

```

```
adult$income.class=as.factor(income.class)
```

```
adult$income.class=as.factor(adult$income.class)
```

```
str(adult)
```

```
estincome.class=knn.cv(adult[,-7], adult[,7], 5)
```

```
estincome.class
```

```
> t1=table(income.class, estincome.class)
```

```
> str(adult)
```

```
'data.frame': 16281 obs. of 7 variables:
 $ age      : num  0.1096 0.2877 0.1507 0.3699 0.0137 ...
 $ fnlwgt   : num  0.1444 0.0517 0.219 0.0994 0.0609 ...
 $ education.num : num  0.4 0.533 0.733 0.6 0.6 ...
 $ capital.gain : num  0 0 0 0.0769 0 ...
 $ capital.loss : num  0 0 0 0 0 0 0 0 0 ...
 $ hours.per.week: num  0.398 0.5 0.398 0.398 0.296 ...
 $ income.class : Factor w/ 2 levels "<=50K.," ">50K.": 1 1 2 2 1 1 1 2 1
1 ...
```

```
> t1
```

```
      estincome.class
income.class <=50K. >50K.
      <=50K. 11337 1098
      >50K. 2152 1694
```

```
> (11337+1694)/16281
```

```
[1] 0.8003808
```

```
> indtrain=sample(1:16281, 9700)
```

```
> ls.str()
```

```
adult: 'data.frame': 16281 obs. of 7 variables:
 $ age      : num  0.1096 0.2877 0.1507 0.3699 0.0137 ...
 $ fnlwgt   : num  0.1444 0.0517 0.219 0.0994 0.0609 ...
 $ education.num : num  0.4 0.533 0.733 0.6 0.6 ...
 $ capital.gain : num  0 0 0 0.0769 0 ...
 $ capital.loss : num  0 0 0 0 0 0 0 0 0 ...
 $ hours.per.week: num  0.398 0.5 0.398 0.398 0.296 ...
 $ income.class : Factor w/ 2 levels "<=50K.," ">50K.": 1 1 2 2 1 1 1 2 1
1 ...
estincome.class : Factor w/ 2 levels "<=50K.," ">50K.": 1 1 1 2 1 1 1 2 1
1 ...
income.class : Factor w/ 2 levels "<=50K.," ">50K.": 1 1 2 2 1 1 1 2 1 1
...
indtrain : int [1:9700] 6203 15486 6667 13874 10072 9913 3408 3915 15131
4107 ...
native.country : Factor w/ 40 levels " Cambodia"," Canada",...: 38 38 38 38
38 38 38 38 38 ...
occupation : Factor w/ 14 levels " Adm-clerical",...: 7 5 11 7 4 8 7 10 8 3
...
t1 : 'table' int [1:2, 1:2] 11337 2152 1098 1694
test : 'data.frame': 6581 obs. of 7 variables:
 $ age      : num  0.1507 0.3699 0.0137 0.2329 0.0959 ...
 $ fnlwgt   : num  0.219 0.0994 0.0609 0.1254 0.2412 ...
 $ education.num : num  0.733 0.6 0.6 0.333 0.6 ...
 $ capital.gain : num  0 0.0769 0 0 0 ...
 $ capital.loss : num  0 0 0 0 0 0 0 0 0 ...
 $ hours.per.week: num  0.398 0.398 0.296 0.296 0.398 ...
 $ income.class : Factor w/ 2 levels "<=50K.," ">50K.": 2 2 1 1 1 2 1 2 2
1 ...
train : 'data.frame': 9700 obs. of 7 variables:
 $ age      : num  0.1507 0.7945 0.0822 0.274 0.2192 ...
 $ fnlwgt   : num  0.1532 0.0588 0.0722 0.2525 0.1292 ...
 $ education.num : num  0.533 0.333 0.533 0.133 0.867 ...
 $ capital.gain : num  0 0 0 0 0 0 0 0 0 ...
 $ capital.loss : num  0 0 0 0 0 0 0 0 0 ...
 $ hours.per.week: num  0.398 0.143 0.48 0.449 0.378 ...
 $ income.class : Factor w/ 2 levels "<=50K.," ">50K.": 1 1 1 1 1 2 1 1 1
1 ...
workclass : Factor w/ 8 levels " Federal-gov",...: 4 4 2 4 4 4 4 6 4 4 ...
```

```
> realincome.class=adult[indtrain, 7]
```

```

> realtest=adult[-indtrain, 7]
> estincome.class=knn(train, train, realincome.class, 5)
> train=adult[indtrain, 1:6]
> test=adult[-indtrain, 1:6]
> estincome.class=knn(train, train, realincome.class, 5)
> t2=table(estincome.class, realincome.class)
> t2
      realincome.class
estincome.class  <=50K. >50K.
      <=50K.    6975  1010
      >50K.     400  1315
> (6975+1315)/9700
[1] 0.8546392
> esttest=knn(train, test, realincome.class, 5)
> t3=table(esttest, realtest)
> t3
      realtest
esttest  <=50K. >50K.
      <=50K.    4604   875
      >50K.     456   646
> (4604+646)/9700
[1] 0.5412371

```

## 6- Conclusions on the best model(s) and their absolute significance

Evaluating those models and determining whether they achieve the business objectives and answer the research questions



Impacts of variables on income class estimation			
Decision tree	Mytreeadult	Mytreeadult2	Mytreeadult3
Variables used in the tree construction	Age	Capital.gain	Capital.gain
	Education	Capital.loss	Capital.loss
	Occupation	Education	Education.num
	Sex	Relationship	Marital.status
Efficiency rate	80.83 %	84.98%	85.01%

We apply the neural network on variable “income.class” of the adult dataset.

After inputting the dataset in R, we prepare 3 training sets of 4070, 8140 and 12210 observations (25%, 50% and 75%) respectively. The rest will constitute the validation sets.

```
> set.seed(1234)
> index <- sample(1:nrow(adult), 12210)
> adult.test = adult[index,]
> adult.valid = adult[-index,]
```

Then we set up our neural network with 4, 8 and then 10 for the size of the hidden layer:

```
> adult.net = nnet(income.class~.,data=adult.test,size=10)
# weights: 981
initial value 3135.995213
iter 10 value 1465.151519
iter 20 value 1291.414488
iter 30 value 1171.231809
iter 40 value 1079.722209
iter 50 value 1001.254668
iter 60 value 937.061026
iter 70 value 882.046138
iter 80 value 843.915509
iter 90 value 823.209449
iter 100 value 807.510902
final value 807.510902
stopped after 100 iterations
```

Then we try our model on the validation set and use a cross validation table to assess the performance:

```
> adult.valid$est.income =
predict(adult.net,adult.valid,type="class")
>
Cross.Table.validation=table(adult.valid$income.class,adult.valid$es
t.income)
> Cross.Table.validation
```

	<=50K.	>50K.
<=50K.	2844	236
>50K.	383	608

```
> (2844+608)/(16281-12210)
[1] 0.8479489
```

We get a pretty good 84.79% correct classification for this model.

Now we do the same again with the 8 other experiences and compute the false negative and false positive rates. Results are shown in the tables below:

Overall Score	Training Set Size		
Hidden Layer Size	4070	8140	12210
4	83.12%	84.38%	85.21%
8	91.17%	83.50%	84.30%
10	80.85%	84.56%	84.79%

False Negative Rate	Training Set Size		
Hidden Layer Size	4070	8140	12210
4	11.70%	12.14%	12.20%
8	1.62%	12.23%	12.25%
10	13.23%	11.45%	11.87%

False Positive Rate	Training Set Size		
Hidden Layer Size	4070	8140	12210
4	65.10%	70.65%	64.15%
8	67.65%	71.00%	60.08%
10	60.08%	69.93%	72.04%

We see that overall, our nine models have similar performance (around 84% success rate) except for one at 91% and another at 80%. The models have a rather low false negative rate around 12% except for an outstanding 1.62% with model (8, 4070). Differences are more on the side of false positive rates where 2 models perform at 60% and the others around 70%.

To make a wise choice we would need more business insights. Model (8, 4070) performs very well but if false positive errors are costly, models (10, 4070) and (8, 12210) would be preferred.

R commands history:

```
> set.seed(1234)
```

```

> index <- sample(1:nrow(adult), 4070)
> adult.test = adult[index,]
> adult.valid = adult[-index,]
> adult.net = nnet(income.class~.,data=adult.test,size=4)
# weights: 393
initial value 2459.108513
iter 10 value 1442.041597
iter 20 value 1330.688498
iter 30 value 1275.115072
iter 40 value 1243.071689
iter 50 value 1224.722375
iter 60 value 1197.039751
iter 70 value 1155.090450
iter 80 value 1115.225276
iter 90 value 1089.683483
iter 100 value 1074.793746
final value 1074.793746
stopped after 100 iterations
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
>
Cross.Table.Validation=table(adult.valid$income.class,adult.valid$est.income)
> Cross.Table.Validation

```

	<=50K.	>50K.
<=50K.	8376	951
>50K.	1110	1774

```

> set.seed(12345)
> index <- sample(1:nrow(adult), 8140)
> adult.test = adult[index,]
> adult.valid = adult[-index,]
> adult.net = nnet(income.class~.,data=adult.test,size=4)
# weights: 393
initial value 6119.813088
iter 10 value 2898.599148
iter 20 value 2626.897882
iter 30 value 2480.079774
iter 40 value 2413.075999
iter 50 value 2364.028423
iter 60 value 2336.402275
iter 70 value 2307.687463
iter 80 value 2294.885504
iter 90 value 2282.884777
iter 100 value 2276.811490
final value 2276.811490
stopped after 100 iterations
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
>
Cross.Table.Validation=table(adult.valid$income.class,adult.valid$est.income)
> Cross.Table.Validation

```

	<=50K.	>50K.
<=50K.	5704	484
>50K.	788	1165

```

> set.seed(123456)
> index <- sample(1:nrow(adult), 12210)
> adult.test = adult[index,]
> adult.valid = adult[-index,]
> adult.net = nnet(income.class~.,data=adult.test,size=4)
# weights: 393
initial value 10471.002627
iter 10 value 4474.494058
iter 20 value 4076.006072
iter 30 value 3877.872209
iter 40 value 3734.606720
iter 50 value 3679.767884
iter 60 value 3623.159544

```

```

iter 70 value 3592.314267
iter 80 value 3571.323756
iter 90 value 3559.412370
iter 100 value 3545.174976
final value 3545.174976
stopped after 100 iterations
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
>
Cross.Table.Validation=table(adult.valid$income.class,adult.valid$est.income)
> Cross.Table.Validation

      <=50K. >50K.
<=50K.    2878    202
>50K.      400    591
> set.seed(1234)
> index <- sample(1:nrow(adult), 4070)
> adult.test = adult[index,]
> adult.valid = adult[-index,]
> adult.net = nnet(income.class~.,data=adult.test,size=8)
# weights: 785
initial value 2287.018750
iter 10 value 1627.926927
iter 20 value 1283.700990
iter 30 value 1157.988898
iter 40 value 1070.517132
iter 50 value 1009.628010
iter 60 value 965.731174
iter 70 value 935.132515
iter 80 value 911.736968
iter 90 value 890.600135
iter 100 value 876.674629
final value 876.674629
stopped after 100 iterations
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
>
Cross.Table.Validation=table(adult.valid$income.class,adult.valid$est.income)
> Cross.Table.Validation

      <=50K. >50K.
<=50K.    8501    826
>50K.     1406   1478
> set.seed(12345)
> index <- sample(1:nrow(adult), 8140)
> adult.test = adult[index,]
> adult.valid = adult[-index,]
> adult.net = nnet(income.class~.,data=adult.test,size=8)
# weights: 785
initial value 5785.640588
iter 10 value 3042.512614
iter 20 value 2685.571641
iter 30 value 2496.644827
iter 40 value 2401.234562
iter 50 value 2335.151536
iter 60 value 2280.621117
iter 70 value 2211.052535
iter 80 value 2167.950338
iter 90 value 2138.814242
iter 100 value 2099.496987
final value 2099.496987
stopped after 100 iterations
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
>
Cross.Table.Validation=table(adult.valid$income.class,adult.valid$est.income)
> Cross.Table.Validation

```

	<=50K.	>50K.
<=50K.	5629	559
>50K.	784	1169

```

> set.seed(123456)
> index <- sample(1:nrow(adult), 12210)
> adult.test = adult[index,]
> adult.valid = adult[-index,]
> adult.net = nnet(income.class~.,data=adult.test,size=8)
# weights: 785
initial value 10858.504549
iter 10 value 4289.889326
iter 20 value 3932.774901
iter 30 value 3703.149854
iter 40 value 3616.784994
iter 50 value 3542.756767
iter 60 value 3449.070057
iter 70 value 3397.384907
iter 80 value 3351.014643
iter 90 value 3323.158446
iter 100 value 3302.889626
final value 3302.889626
stopped after 100 iterations
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
>
Cross.Table.Validation=table(adult.valid$income.class,adult.valid$est.income)
> Cross.Table.Validation

```

	<=50K.	>50K.
<=50K.	2837	243
>50K.	396	595

```

> set.seed(1234)
> index <- sample(1:nrow(adult), 4070)
> adult.test = adult[index,]
> adult.valid = adult[-index,]
> adult.net = nnet(income.class~.,data=adult.test,size=10)
# weights: 981
initial value 2397.471223
iter 10 value 1468.756862
iter 20 value 1299.158308
iter 30 value 1172.898342
iter 40 value 1075.467612
iter 50 value 1008.177348
iter 60 value 951.346606
iter 70 value 878.801162
iter 80 value 820.238921
iter 90 value 780.151173
iter 100 value 744.367379
final value 744.367379
stopped after 100 iterations
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
>
Cross.Table.Validation=table(adult.valid$income.class,adult.valid$est.income)
> Cross.Table.Validation

```

	<=50K.	>50K.
<=50K.	8246	1081
>50K.	1257	1627

```

> set.seed(12345)
> index <- sample(1:nrow(adult), 8140)
> adult.test = adult[index,]
> adult.valid = adult[-index,]
> adult.net = nnet(income.class~.,data=adult.test,size=10)
# weights: 981
initial value 10935.744667
iter 10 value 3408.853241

```

```

iter 20 value 2786.442454
iter 30 value 2641.101986
iter 40 value 2566.489040
iter 50 value 2525.891694
iter 60 value 2479.240222
iter 70 value 2437.018291
iter 80 value 2398.966117
iter 90 value 2373.295341
iter 100 value 2337.661751
final value 2337.661751
stopped after 100 iterations
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
>
Cross.Table.Validation=table(adult.valid$income.class,adult.valid$est.income)
> Cross.Table.Validation

      <=50K. >50K.
<=50K.   5663   525
>50K.     732  1221
> set.seed(123456)
> index <- sample(1:nrow(adult), 12210)
> adult.test = adult[index,]
> adult.valid = adult[-index,]
> adult.net = nnet(income.class~.,data=adult.test,size=10)
# weights: 981
initial value 9431.329023
iter 10 value 4370.103027
iter 20 value 3974.326523
iter 30 value 3733.639628
iter 40 value 3628.923494
iter 50 value 3526.704198
iter 60 value 3418.288646
iter 70 value 3320.457069
iter 80 value 3253.267738
iter 90 value 3204.914289
iter 100 value 3161.658111
final value 3161.658111
stopped after 100 iterations
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
> adult.valid$est.income = predict(adult.net,adult.valid,type="class")
>
Cross.Table.Validation=table(adult.valid$income.class,adult.valid$est.income)
> Cross.Table.Validation

      <=50K. >50K.
<=50K.   2844   236
>50K.     383   608

```

## 7- Reference

Daniel T. Larose, *Discovering knowledge in data an introduction to data mining*, Wiley online , 2005, 241 pages

Daniel T. Larose, *Data mining Methods and Models*, Wiley-Interscience Ed., 2006, 340 pages

Robert I. Kabacoff, *R in action: Data analysis and graphics with R*, Manning Ed. 2011. 474 pages

Stéphane Tufféry, *Data Mining and statistics for decision-making*, Wiley Ed., 2011, 717 pages

Ron Lohavi, *Scaling Up The Accuracy of Naïve-Bayes Classifiers: a Decision-Tree hybrid*, 2011, 6 pages

Everything on the course site SIO-6051

<http://archive.ics.uci.edu/ml/datasets/Adult>

<http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.names>

## 8-Appendices

### Graphs and R commands

```
> summary(Provided.dataset)
   age          workclass      fnlwtg      education  education.num
Min.   :17.00   Private      :11210   Min.    : 13492   HS-grad    :5283   Min.    : 1.00
1st Qu.:28.00   Self-emp-not-inc: 1321   1st Qu.: 116736   Some-college:3587   1st Qu.: 9.00
Median :37.00   Local-gov       : 1043   Median : 177831   Bachelors  :2670   Median :10.00
Mean   :38.77   ?               : 963   Mean   : 189436   Masters    : 934   Mean   :10.07
3rd Qu.:48.00   State-gov       : 683   3rd Qu.: 238384   Assoc-voc  : 679   3rd Qu.:12.00
Max.   :90.00   Self-emp-inc    : 579   Max.   :1490400   11th       : 637   Max.   :16.00
              (other)      : 482              (other)      :2491

   marital.status      occupation      relationship      race
Divorced      :2190   Prof-specialty:2032   Husband       :6523   Amer-Indian-Eskimo: 159
Married-AF-spouse : 14   Exec-managerial:2020   Not-in-family :4278   Asian-Pac-Islander: 480
Married-civ-spouse :7403   Craft-repair  :2013   Other-relative: 525   Black              : 1561
Married-spouse-absent: 210   Sales         :1854   Own-child     :2513   Other              : 135
Never-married     :5434   Adm-clerical  :1841   Unmarried     :1679   white             :13946
Separated         : 505   Other-service :1628   wife          : 763
Widowed          : 525   (other)       :4893

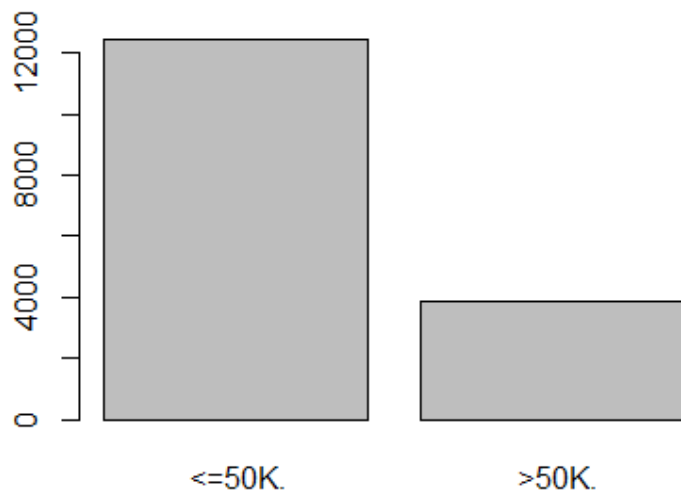
   sex      capital.gain      capital.loss      hours.per.week      native.country      income.class
Female: 5421   Min.    : 0   Min.    : 0.0   Min.    : 1.00   United-States:14662   <=50K.:12435
Male :10860   1st Qu.: 0   1st Qu.: 0.0   1st Qu.:40.00   Mexico        : 308   >50K. : 3846
              Median : 0   Median : 0.0   Median :40.00   ?             : 274
              Mean   :1082   Mean   : 87.9   Mean   :40.39   Philippines   : 97
              3rd Qu.: 0   3rd Qu.: 0.0   3rd Qu.:45.00   Puerto-Rico   : 70
              Max.   :99999   Max.   :3770.0   Max.   :99.00   Germany       : 69
              (other)      : 801
```

```

> ls.str()
Provided.dataset : 'data.frame': 16281 obs. of  15 variables:
 $ age           : int  25 38 28 44 18 34 29 63 24 55 ...
 $ workclass     : Factor w/ 9 levels " ?"," Federal-gov",...: 5 5 3 5 1 5 1 7 5 5 ...
 $ fnlwtg       : int 226802 89814 336951 160323 103497 198693 227026 104626 369667 104996 ...
 $ education     : Factor w/ 16 levels " 10th"," 11th",...: 2 12 8 16 16 1 12 15 16 6 ...
 $ education.num : int  7 9 12 10 10 6 9 15 10 4 ...
 $ marital.status: Factor w/ 7 levels " Divorced"," Married-AF-spouse",...: 5 3 3 3 5 5 5 3 5 3 ...
 $ occupation    : Factor w/ 15 levels " ?"," Adm-clerical",...: 8 6 12 8 1 9 1 11 9 4 ...
 $ relationship  : Factor w/ 6 levels " Husband"," Not-in-family",...: 4 1 1 1 4 2 5 1 5 1 ...
 $ race          : Factor w/ 5 levels " Amer-Indian-Eskimo",...: 3 5 5 3 5 5 3 5 5 5 ...
 $ sex           : Factor w/ 2 levels " Female"," Male": 2 2 2 2 1 2 2 2 1 2 ...
 $ capital.gain  : int  0 0 0 7688 0 0 0 3103 0 0 ...
 $ capital.loss  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ hours.per.week: int  40 50 40 40 30 30 40 32 40 10 ...
 $ native.country: Factor w/ 41 levels " ?"," Cambodia",...: 39 39 39 39 39 39 39 39 39 39 ...
 $ income.class  : Factor w/ 2 levels " <=50K.", ">50K.": 1 1 2 2 1 1 1 2 1 1 ...

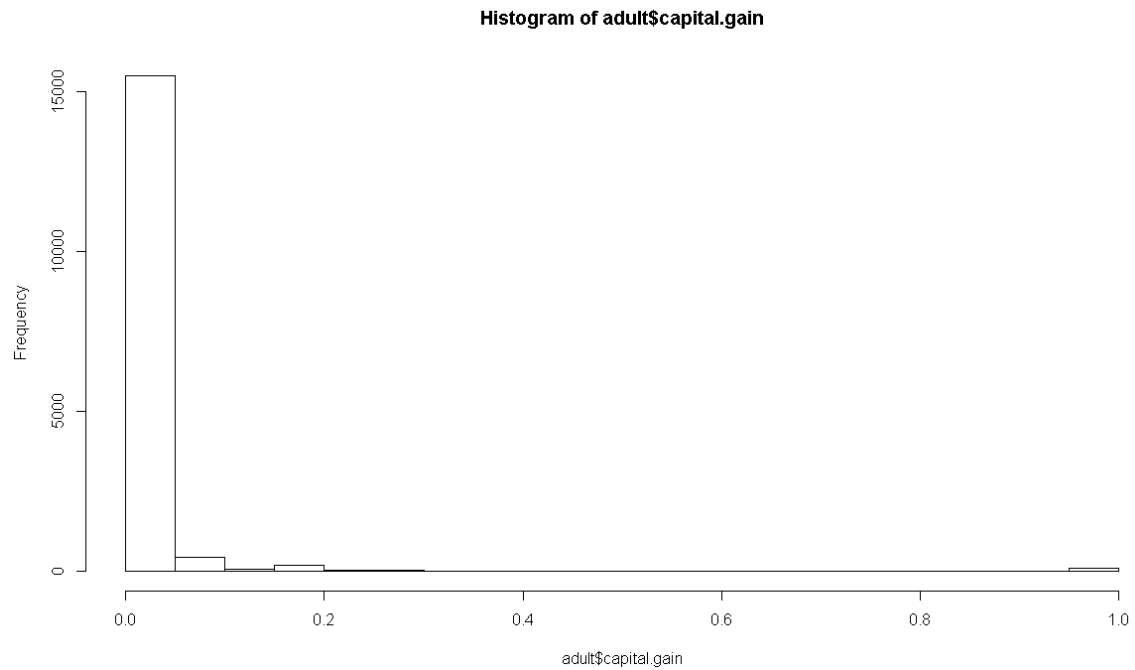
```

**Figure 1: Distribution of the Dataset : Salary higher than 50K\$ VS. lower lower or egal to 50k\$**



The most correlated variable among the dataset:





### History of R commands

```
Churn.dataset.from.Larose <- read.csv("~/Rdata/Churn dataset from Larose.csv")

View(Churn.dataset.from.Larose)

setwd("~/Users/Dom/Documents/Rdata")

View(Churn.dataset.from.Larose)

load("~/Rdata/cleaned.RData")

library(class)

attach(cleaned)

load("C:/Users/Pidory/Desktop/Projet_mahmoud_with_Min Max (1).RData")

load("C:/Users/Pidory/Desktop/Projet_mahmoud_with_Min Max (1).RData")

str(adult)

summary(adult)

adult$occupation [3,]
```

```

adult$occupation [2,]

adult$occupation

hist(adult$capitalgain)

str(adult)

hist(adult$capital.gain)

str(adult)

adult$workclass=NULL

adult$education=NULL

adult$marital.status=NULL

adult$occupation=NULL

adult$relationship=NULL

adult$race=NULL

adult$sex=NULL

adult$native.country=NULL

adult$income.class=as.factor(income.class)

adult$income.class=as.factor(adult$income.class)

str(adult)

estincome.class=knn.cv(adult[, -7], adult[, 7], 5)

estincome.class

> t1=table(income.class, estincome.class)

> str(adult)

str(adult)

adult$workclass=NULL

adult$education=NULL

adult$marital.status=NULL

adult$occupation=NULL

adult$relationship=NULL

adult$race=NULL

adult$sex=NULL

adult$native.country=NULL

adult$income.class=as.factor(income.class)

```

```

adult$income.class=as.factor(adult$income.class)

str(adult)

estincome.class=knn.cv(adult[, -7], adult[, 7], 5)

estincome.class

str(adult)

adult$workclass=NULL

adult$education=NULL

adult$marital.status=NULL

adult$occupation=NULL

adult$relationship=NULL

adult$race=NULL

adult$sex=NULL

adult$native.country=NULL

adult$income.class=as.factor(income.class)

adult$income.class=as.factor(adult$income.class)

str(adult)

estincome.class=knn.cv(adult[, -7], adult[, 7], 5)

knn?

help knn

library(class)

adult$income.class=as.factor(income.class)

hist(adult)

estincome.class=knn.cv(adult[, -7], adult[, 7], 5)

estincome.class

t1=table(income.class, estincome.class)

t1=table(income.class, estincome.class)

str(adult)

estincome.class=knn.cv(adult[, -7], adult[, 7], 5)

t1=table(income.class, estincome.class)

adult = read.csv("adult.csv", header = TRUE)

attach(adult)

str(adult)

```

```

summary(adult)

table(workclass)

table(occupation)

table(native.country)

adult$workclass = as.character(adult$workclass)

adult$workclass[adult$workclass==" ?"] = as.character(sample(adult$workclass[which(adult$workclass != " ?")], 963, replace = FALSE))

adult$workclass = as.factor(adult$workclass)

unique(adult$workclass)

adult$occupation = as.character(adult$occupation)

adult$occupation[adult$occupation==" ?"] = as.character(sample(occupation[which(adult$occupation != " ?")], 966, replace = FALSE))

adult$occupation = as.factor(adult$occupation)

unique(adult$occupation)

adult$native.country = as.character(adult$native.country)

adult$native.country[adult$native.country==" ?"] = as.character(sample(adult$native.country[which(adult$native.country != " ?")], 274,
replace = FALSE))

adult$native.country = as.factor(adult$native.country)

unique(adult$native.country)

str(adult)

summary(adult)

```

---

<sup>i</sup> <http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.names>