Harvard Data Science Final Project: Predicting individual income

Maxandre Hebert

2023-08-25

Contents

Introduction

In the world of data, stories emerge from numbers. With the rise of technology, our ability to dive deep into datasets and glean meaningful insights is a skill that is increasingly in demand. My journey in this paper begins with a dataset that many might view as just numbers and categories - the 1994 Adult Census Income data, generously shared by the University of California, Irvine.

Here is the link :https://www.kaggle.com/datasets/uciml/adult-census-income.

Dataset description

This dataset, a product of meticulous collection by Ronny Kohavi and Barry Becker, paints a picture of an individual's economic standing based on diverse personal and professional parameters. It's a tableau of life in numbers. Overall the dataset contain

Goal of the project

The goal of this project is straightforward: to understand the tale these numbers tell and to see if, with the right tools, I can predict whether an individual's income surpasses $50,000 annually.

In this study, I have choosen 5 models that will help me predicting the outcome.

1- Logistic regression : Think of it as trying to predict the odds of something happening (like earning more than $50k). It's good for our project because it gives clear probabilities and is straightforward to understand.

2- Random Forest : It's like gathering opinions from a crowd of experts (trees) and trusting the majority. Perfect for our project because it handles lots of data well and can manage various types of variables.

3- XGboost : Similar to Random Forest but with a twist: it corrects its previous mistakes in each step. It's great for our project as it's known for delivering high performance and accuracy.

4- support vector machine : Imagine drawing the best boundary line that separates two groups (like income brackets) as widely as possible. It's a good fit because it captures the nuances in our data, especially when the boundary isn't straightforward.

5- K-Nearest Neighbors : Think of it as asking your closest neighbors for advice and following the most common suggestion. Useful for our project because it directly considers the surrounding data points to make a decision.

Steps that were performed

1-Download the data. 2-Clean the data NA and ? 3- Transform character data into factor for machine learning. 4- Data visualization of categorical and numerical variable. 5- Process five different machine learning model. 6- Presenting the result. 7- Conclusion of the result.

Methodology/Analysis

In this section we gonna first import the data then clean the data then do the exploratory work.

Importing the data

Now First let's import the data. To make the job easy I have it ready on my github on csv.

```
#Load necessary package
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Le chargement a nécessité le package : tidyverse
```

```
## Warning: le package 'tidyverse' a été compilé avec la version R 4.2.3
```

```
## Warning: le package 'ggplot2' a été compilé avec la version R 4.2.3
```

```
## Warning: le package 'tibble' a été compilé avec la version R 4.2.3
```

```
## Warning: le package 'tidyr' a été compilé avec la version R 4.2.2
```

```
## Warning: le package 'readr' a été compilé avec la version R 4.2.3
```

```
## Warning: le package 'purrr' a été compilé avec la version R 4.2.2
```

```
## Warning: le package 'dplyr' a été compilé avec la version R 4.2.3
```

```
## Warning: le package 'stringr' a été compilé avec la version R 4.2.2
```

```
## Warning: le package 'forcats' a été compilé avec la version R 4.2.2
```

```
## Warning: le package 'lubridate' a été compilé avec la version R 4.2.3
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

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

```
## Le chargement a nécessité le package : caret
```

```
## Warning: le package 'caret' a été compilé avec la version R 4.2.3
```

```
## Le chargement a nécessité le package : lattice
##
## Attachement du package : 'caret'
##
## L'objet suivant est masqué depuis 'package:purrr':
##
##     lift
```

```
if(!require(corrplot)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Le chargement a nécessité le package : corrplot

## Warning: le package 'corrplot' a été compilé avec la version R 4.2.3

## corrplot 0.92 loaded
```

```
if(!require(ggridges)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Le chargement a nécessité le package : ggridges

## Warning: le package 'ggridges' a été compilé avec la version R 4.2.3
```

```
if(!require(plotly)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Le chargement a nécessité le package : plotly

## Warning: le package 'plotly' a été compilé avec la version R 4.2.3

##
## Attachement du package : 'plotly'
##
## L'objet suivant est masqué depuis 'package:ggplot2':
##
##     last_plot
##
## L'objet suivant est masqué depuis 'package:stats':
##
##     filter
##
## L'objet suivant est masqué depuis 'package:graphics':
##
##     layout
```

```
if(!require(knitr)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Le chargement a nécessité le package : knitr

## Warning: le package 'knitr' a été compilé avec la version R 4.2.3
```

```r
if(!require(fmsb)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Le chargement a nécessité le package : fmsb
```

```
## Warning: le package 'fmsb' a été compilé avec la version R 4.2.3
```

```
## Registered S3 methods overwritten by 'fmsb':
##   method    from
##   print.roc pROC
##   plot.roc  pROC
```

```r
if(!require(vcd)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Le chargement a nécessité le package : vcd
```

```
## Warning: le package 'vcd' a été compilé avec la version R 4.2.3
```

```
## Le chargement a nécessité le package : grid
##
## Attachement du package : 'vcd'
##
## L'objet suivant est masqué depuis 'package:fmsb':
##
##     oddsratio
```

```r
if(!require(randomForest)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Le chargement a nécessité le package : randomForest
```

```
## Warning: le package 'randomForest' a été compilé avec la version R 4.2.3
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attachement du package : 'randomForest'
##
## L'objet suivant est masqué depuis 'package:dplyr':
##
##     combine
##
## L'objet suivant est masqué depuis 'package:ggplot2':
##
##     margin
```

```r
if(!require(xgboost)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Le chargement a nécessité le package : xgboost
```

```
## Warning: le package 'xgboost' a été compilé avec la version R 4.2.3
```

```
##
## Attachement du package : 'xgboost'
##
## L'objet suivant est masqué depuis 'package:plotly':
##
##     slice
##
## L'objet suivant est masqué depuis 'package:dplyr':
##
##     slice
```

```
if(!require(e1071)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Le chargement a nécessité le package : e1071
```

```
## Warning: le package 'e1071' a été compilé avec la version R 4.2.3
```

```
if(!require(class)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Le chargement a nécessité le package : class
```

```
library(tidyverse)
library(caret)
library(corrplot)
library(ggridges)
library(plotly)
library(knitr)
library(fmsb)
library(vcd)
library(randomForest)
library(xgboost)
library(e1071)
library(class)
```

```
#Download the csv file

original_data <- read_csv("https://raw.githubusercontent.com/01110001/Harvard-data-science-final/main/Al
```

```
## Rows: 32561 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (9): workclass, education, marital.status, occupation, relationship, rac...
## dbl (6): age, fnlwgt, education.num, capital.gain, capital.loss, hours.per.week
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
str(original_data)
```

```
## spc_tbl_ [32,561 x 15] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ age           : num [1:32561] 90 82 66 54 41 34 38 74 68 41 ...
##  $ workclass     : chr [1:32561] "?" "Private" "?" "Private" ...
##  $ fnlwgt        : num [1:32561] 77053 132870 186061 140359 264663 ...
##  $ education     : chr [1:32561] "HS-grad" "HS-grad" "Some-college" "7th-8th" ...
##  $ education.num : num [1:32561] 9 9 10 4 10 9 6 16 9 10 ...
##  $ marital.status: chr [1:32561] "Widowed" "Widowed" "Widowed" "Divorced" ...
##  $ occupation    : chr [1:32561] "?" "Exec-managerial" "?" "Machine-op-inspct" ...
##  $ relationship  : chr [1:32561] "Not-in-family" "Not-in-family" "Unmarried" "Unmarried" ...
##  $ race          : chr [1:32561] "White" "White" "Black" "White" ...
##  $ sex           : chr [1:32561] "Female" "Female" "Female" "Female" ...
##  $ capital.gain  : num [1:32561] 0 0 0 0 0 0 0 0 0 0 ...
##  $ capital.loss  : num [1:32561] 4356 4356 4356 3900 3900 ...
##  $ hours.per.week: num [1:32561] 40 18 40 40 40 45 40 20 40 60 ...
##  $ native.country: chr [1:32561] "United-States" "United-States" "United-States" "United-States" ...
##  $ income        : chr [1:32561] "<=50K" "<=50K" "<=50K" "<=50K" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   workclass = col_character(),
##   ..   fnlwgt = col_double(),
##   ..   education = col_character(),
##   ..   education.num = col_double(),
##   ..   marital.status = col_character(),
##   ..   occupation = col_character(),
##   ..   relationship = col_character(),
##   ..   race = col_character(),
##   ..   sex = col_character(),
##   ..   capital.gain = col_double(),
##   ..   capital.loss = col_double(),
##   ..   hours.per.week = col_double(),
##   ..   native.country = col_character(),
##   ..   income = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

We can see that we have that the dataframe contain 32561 rows and 15 variables (column).

The variable we want to predict is the income variable.

Data Cleaning

Now we will see if the data contain missing value.

```
sum(is.na(original_data))
```

```
## [1] 0
```

The dataframe contain no NA,but it contain some missing data representing as '?'.

Let's remove them.

```r
#removing the ? from the data set.
clean_dataset <- filter(original_data,
                        !workclass == "?",
                        !occupation == "?",
                        !native.country == "?")

str(clean_dataset)
```

```
## spc_tbl_ [30,162 x 15] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ age           : num [1:30162] 82 54 41 34 38 74 68 45 38 52 ...
##  $ workclass     : chr [1:30162] "Private" "Private" "Private" "Private" ...
##  $ fnlwgt        : num [1:30162] 132870 140359 264663 216864 150601 ...
##  $ education     : chr [1:30162] "HS-grad" "7th-8th" "Some-college" "HS-grad" ...
##  $ education.num : num [1:30162] 9 4 10 9 6 16 9 16 15 13 ...
##  $ marital.status: chr [1:30162] "Widowed" "Divorced" "Separated" "Divorced" ...
##  $ occupation    : chr [1:30162] "Exec-managerial" "Machine-op-inspct" "Prof-specialty" "Other-servi
##  $ relationship  : chr [1:30162] "Not-in-family" "Unmarried" "Own-child" "Unmarried" ...
##  $ race          : chr [1:30162] "White" "White" "White" "White" ...
##  $ sex           : chr [1:30162] "Female" "Female" "Female" "Female" ...
##  $ capital.gain  : num [1:30162] 0 0 0 0 0 0 0 0 0 0 ...
##  $ capital.loss  : num [1:30162] 4356 3900 3900 3770 3770 ...
##  $ hours.per.week: num [1:30162] 18 40 40 45 40 20 40 35 45 20 ...
##  $ native.country: chr [1:30162] "United-States" "United-States" "United-States" "United-States" ...
##  $ income        : chr [1:30162] "<=50K" "<=50K" "<=50K" "<=50K" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   workclass = col_character(),
##   ..   fnlwgt = col_double(),
##   ..   education = col_character(),
##   ..   education.num = col_double(),
##   ..   marital.status = col_character(),
##   ..   occupation = col_character(),
##   ..   relationship = col_character(),
##   ..   race = col_character(),
##   ..   sex = col_character(),
##   ..   capital.gain = col_double(),
##   ..   capital.loss = col_double(),
##   ..   hours.per.week = col_double(),
##   ..   native.country = col_character(),
##   ..   income = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Much cleaner now !

Data type

```r
  str(clean_dataset)
```

```
## spc_tbl_ [30,162 x 15] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
```

```
##  $ age           : num [1:30162] 82 54 41 34 38 74 68 45 38 52 ...
##  $ workclass     : chr [1:30162] "Private" "Private" "Private" "Private" ...
##  $ fnlwgt        : num [1:30162] 132870 140359 264663 216864 150601 ...
##  $ education     : chr [1:30162] "HS-grad" "7th-8th" "Some-college" "HS-grad" ...
##  $ education.num : num [1:30162] 9 4 10 9 6 16 9 16 15 13 ...
##  $ marital.status: chr [1:30162] "Widowed" "Divorced" "Separated" "Divorced" ...
##  $ occupation    : chr [1:30162] "Exec-managerial" "Machine-op-inspct" "Prof-specialty" "Other-servi
##  $ relationship  : chr [1:30162] "Not-in-family" "Unmarried" "Own-child" "Unmarried" ...
##  $ race          : chr [1:30162] "White" "White" "White" "White" ...
##  $ sex           : chr [1:30162] "Female" "Female" "Female" "Female" ...
##  $ capital.gain  : num [1:30162] 0 0 0 0 0 0 0 0 0 0 ...
##  $ capital.loss  : num [1:30162] 4356 3900 3900 3770 3770 ...
##  $ hours.per.week: num [1:30162] 18 40 40 45 40 20 40 35 45 20 ...
##  $ native.country: chr [1:30162] "United-States" "United-States" "United-States" "United-States" ...
##  $ income        : chr [1:30162] "<=50K" "<=50K" "<=50K" "<=50K" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   workclass = col_character(),
##   ..   fnlwgt = col_double(),
##   ..   education = col_character(),
##   ..   education.num = col_double(),
##   ..   marital.status = col_character(),
##   ..   occupation = col_character(),
##   ..   relationship = col_character(),
##   ..   race = col_character(),
##   ..   sex = col_character(),
##   ..   capital.gain = col_double(),
##   ..   capital.loss = col_double(),
##   ..   hours.per.week = col_double(),
##   ..   native.country = col_character(),
##   ..   income = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

We can see that our categorical variable are not considered as factor. Let's transform them into factor which is really important for machine learning.

For the purpose of this project we need to transform to character data into factor. This makes it easier for machine learning algorithms to recognize patterns, compare and differentiate between different categories, and make predictions.

```r
# Convert character columns to factors using a for loop
for(col in names(clean_dataset)) {
  if(is.character(clean_dataset[[col]])) {
    clean_dataset[[col]] <- factor(clean_dataset[[col]])
  }
}
str(clean_dataset)
```

```
## spc_tbl_ [30,162 x 15] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ age           : num [1:30162] 82 54 41 34 38 74 68 45 38 52 ...
##  $ workclass     : Factor w/ 7 levels "Federal-gov",..: 3 3 3 3 3 6 1 3 5 3 ...
##  $ fnlwgt        : num [1:30162] 132870 140359 264663 216864 150601 ...
```

```
##  $ education     : Factor w/ 16 levels "10th","11th",..: 12 6 16 12 1 11 12 11 15 10 ...
##  $ education.num : num [1:30162] 9 4 10 9 6 16 9 16 15 13 ...
##  $ marital.status: Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 7 1 6 1 6 5 1 1 5 7 ...
##  $ occupation    : Factor w/ 14 levels "Adm-clerical",..: 4 7 10 8 1 10 10 10 10 8 ...
##  $ relationship  : Factor w/ 6 levels "Husband","Not-in-family",..: 2 5 4 5 5 3 2 5 2 2 ...
##  $ race          : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 5 5 5 5 5 3 5 5 ...
##  $ sex           : Factor w/ 2 levels "Female","Male": 1 1 1 1 2 1 1 1 2 1 ...
##  $ capital.gain  : num [1:30162] 0 0 0 0 0 0 0 0 0 0 ...
##  $ capital.loss  : num [1:30162] 4356 3900 3900 3770 3770 ...
##  $ hours.per.week: num [1:30162] 18 40 40 45 40 20 40 35 45 20 ...
##  $ native.country: Factor w/ 41 levels "Cambodia","Canada",..: 39 39 39 39 39 39 39 39 39 39 ...
##  $ income        : Factor w/ 2 levels "<=50K",">50K": 1 1 1 1 1 2 1 2 2 2 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   workclass = col_character(),
##   ..   fnlwgt = col_double(),
##   ..   education = col_character(),
##   ..   education.num = col_double(),
##   ..   marital.status = col_character(),
##   ..   occupation = col_character(),
##   ..   relationship = col_character(),
##   ..   race = col_character(),
##   ..   sex = col_character(),
##   ..   capital.gain = col_double(),
##   ..   capital.loss = col_double(),
##   ..   hours.per.week = col_double(),
##   ..   native.country = col_character(),
##   ..   income = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Ok now our data is really clean.

Splitting the data

We can now proceed to split the data into a training and test set.

```
#splitting the data
set.seed(123)
test_index <- createDataPartition(y = clean_dataset$income, times = 1, p = 0.2, list = FALSE)
trainning_set <- clean_dataset[-test_index,]
validation_set <- clean_dataset[test_index,]
```

Data Exploration

Descriptive statistic

Now it is time for data exploration my favorite part. With each variable we examine and each visualization we create, we uncover hidden patterns and relationships within the data.

```r
summary(trainning_set)
```

```
##       age                    workclass         fnlwgt                education
##  Min.   :17.00   Federal-gov     :  751   Min.   :  14878   HS-grad     :7887
##  1st Qu.:28.00   Local-gov       : 1660   1st Qu.: 117556   Some-college:5348
##  Median :37.00   Private         :17808   Median : 178312   Bachelors   :4027
##  Mean   :38.44   Self-emp-inc    :  870   Mean   : 189873   Masters     :1293
##  3rd Qu.:47.00   Self-emp-not-inc: 2007   3rd Qu.: 237811   Assoc-voc   :1046
##  Max.   :90.00   State-gov       : 1025   Max.   :1484705   11th        : 838
##                  Without-pay     :    8                     (Other)     :3690
##  education.num                  marital.status            occupation
##  Min.   : 1.00   Divorced             : 3377   Prof-specialty :3232
##  1st Qu.: 9.00   Married-AF-spouse    :   17   Exec-managerial:3197
##  Median :10.00   Married-civ-spouse   :11225   Craft-repair   :3191
##  Mean   :10.12   Married-spouse-absent:  294   Adm-clerical   :2954
##  3rd Qu.:13.00   Never-married        : 7801   Sales          :2875
##  Max.   :16.00   Separated            :  748   Other-service  :2614
##                  Widowed              :  667   (Other)        :6066
##          relationship                 race           sex
##  Husband       :9966   Amer-Indian-Eskimo:  229   Female: 7860
##  Not-in-family :6173   Asian-Pac-Islander:  681   Male  :16269
##  Other-relative: 710   Black             : 2286
##  Own-child     :3584   Other             :  182
##  Unmarried     :2588   White             :20751
##  Wife          :1108
##
##   capital.gain    capital.loss     hours.per.week        native.country
##  Min.   :    0   Min.   :   0.00   Min.   : 1.00   United-States:22055
##  1st Qu.:    0   1st Qu.:   0.00   1st Qu.:40.00   Mexico       :  479
##  Median :    0   Median :   0.00   Median :40.00   Philippines  :  143
##  Mean   : 1085   Mean   :  88.92   Mean   :40.89   Germany      :   94
##  3rd Qu.:    0   3rd Qu.:   0.00   3rd Qu.:45.00   El-Salvador  :   86
##  Max.   :99999   Max.   :4356.00   Max.   :99.00   Puerto-Rico  :   85
##                                                    (Other)      : 1187
##    income
##  <=50K:18123
##  >50K : 6006
##
##
##
##
##
```

Interesting we can see that most of our data set contain people with a revenue of less than 50k.

This is a summary statistic of the data set.

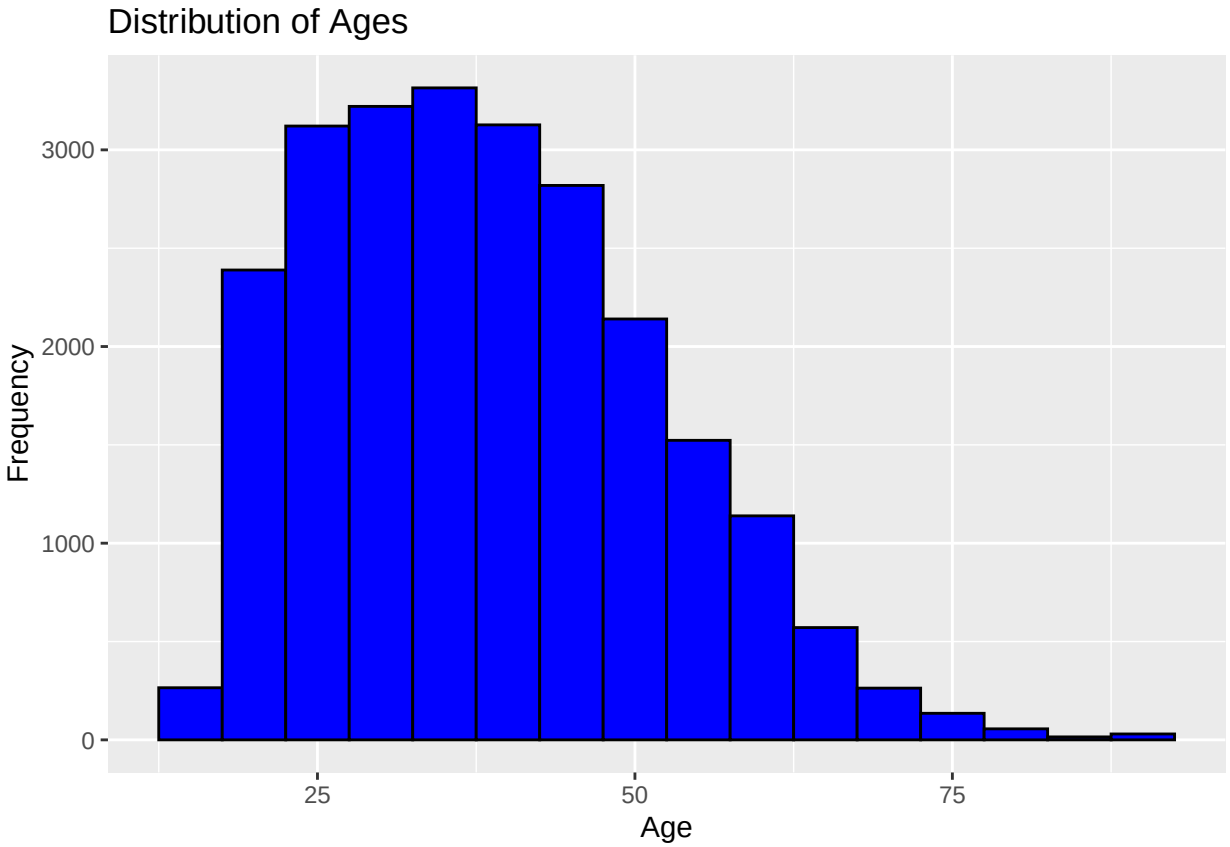
We can see that most person in the data set did earn less than 50k in 1994.

Next.

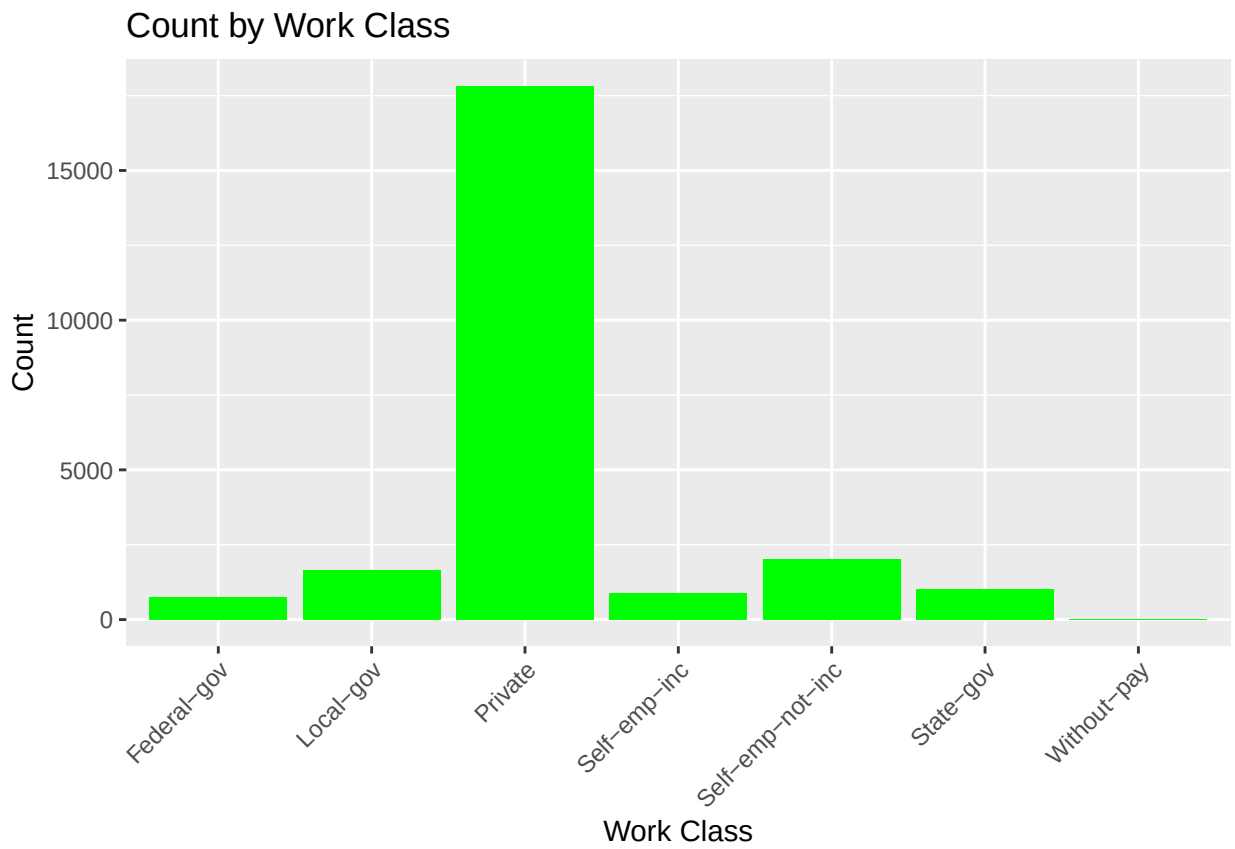
Data Visualization

In this part we are gonna go more visual.

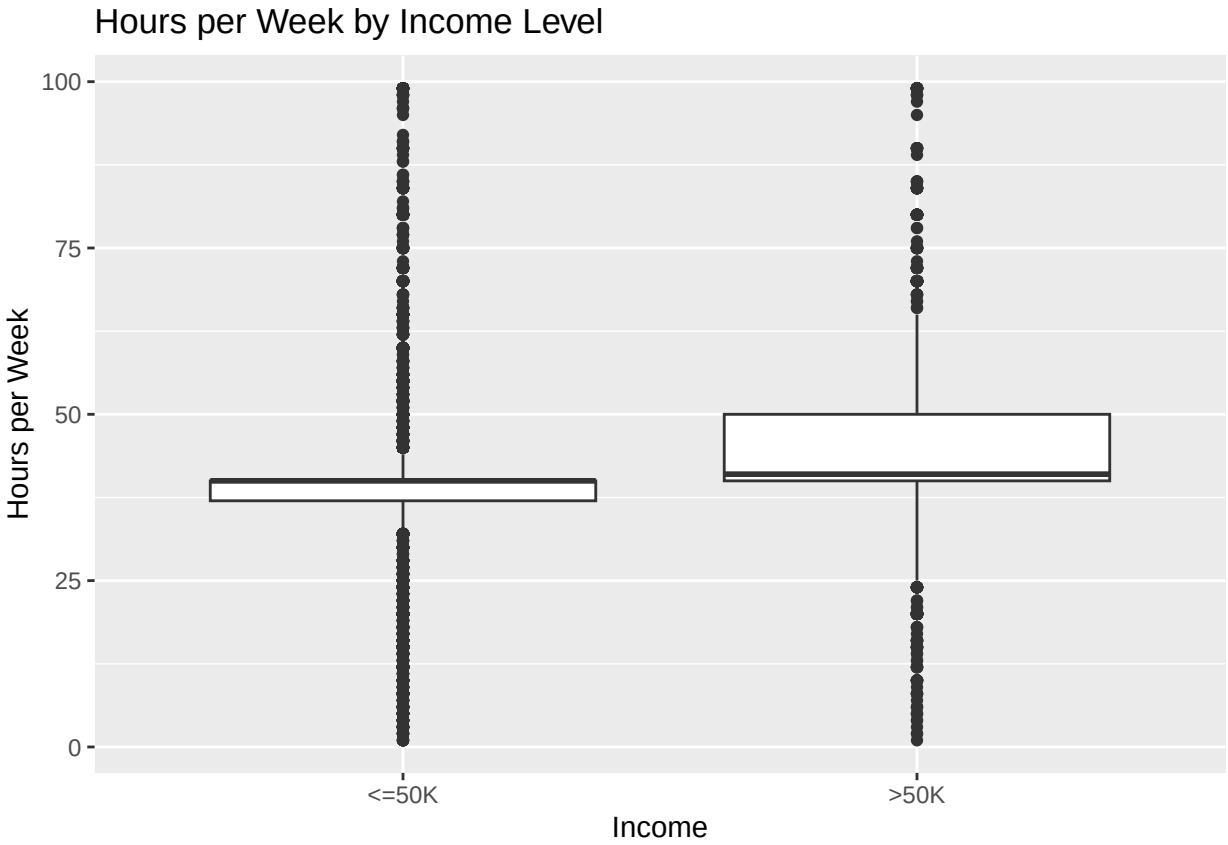
Distribution plot



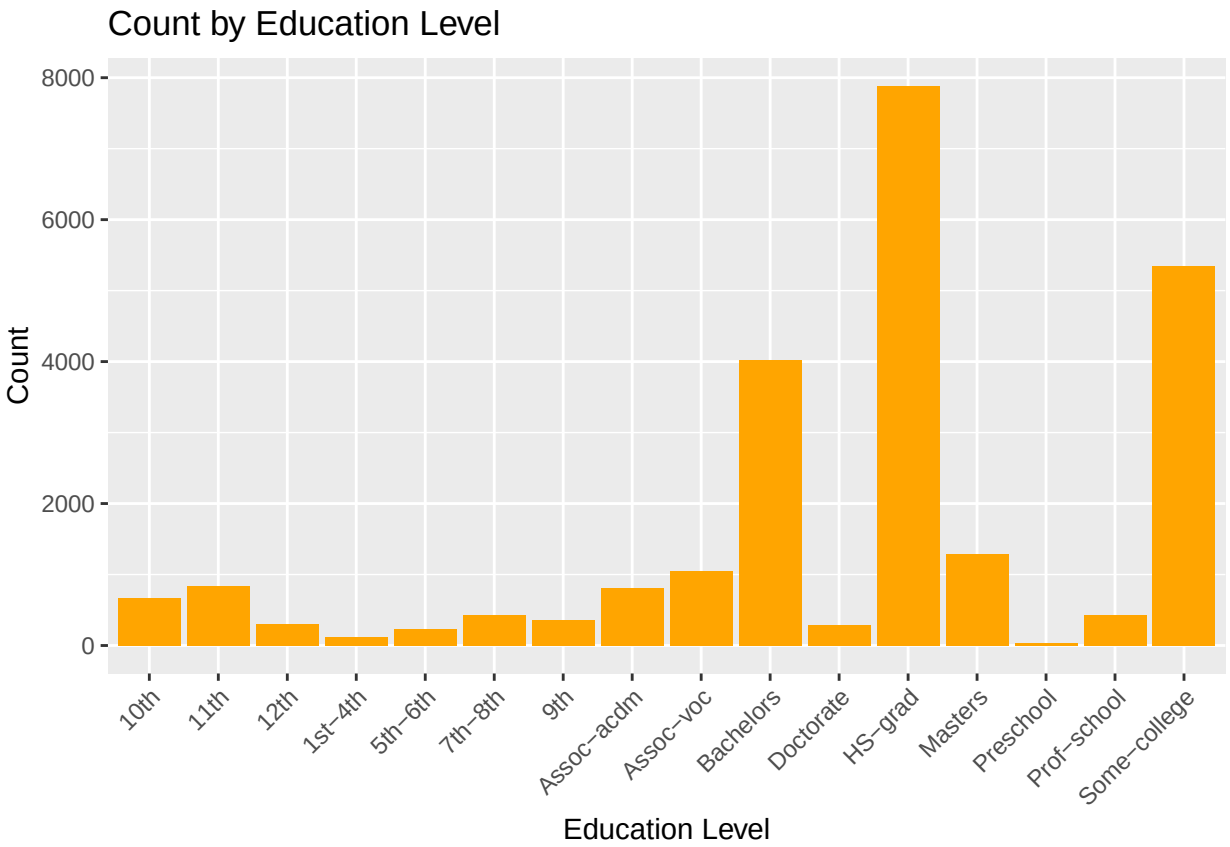
Here we see the distribution of the age of the dataset. We can see that most people are in the range of 25-45 years.



Here we see that most people work in the private sector.

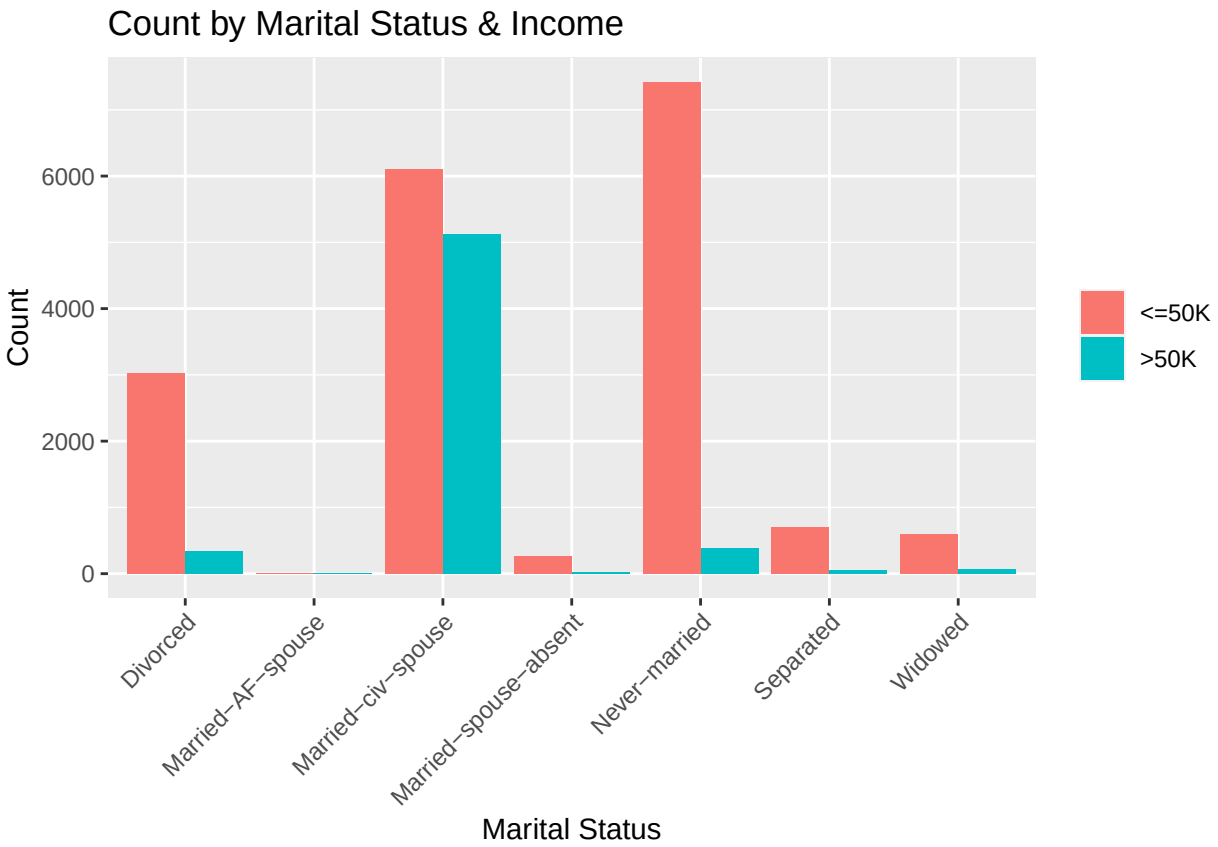


This boxplot show us that people who make more than 50k im annual revenu work more hour on average.



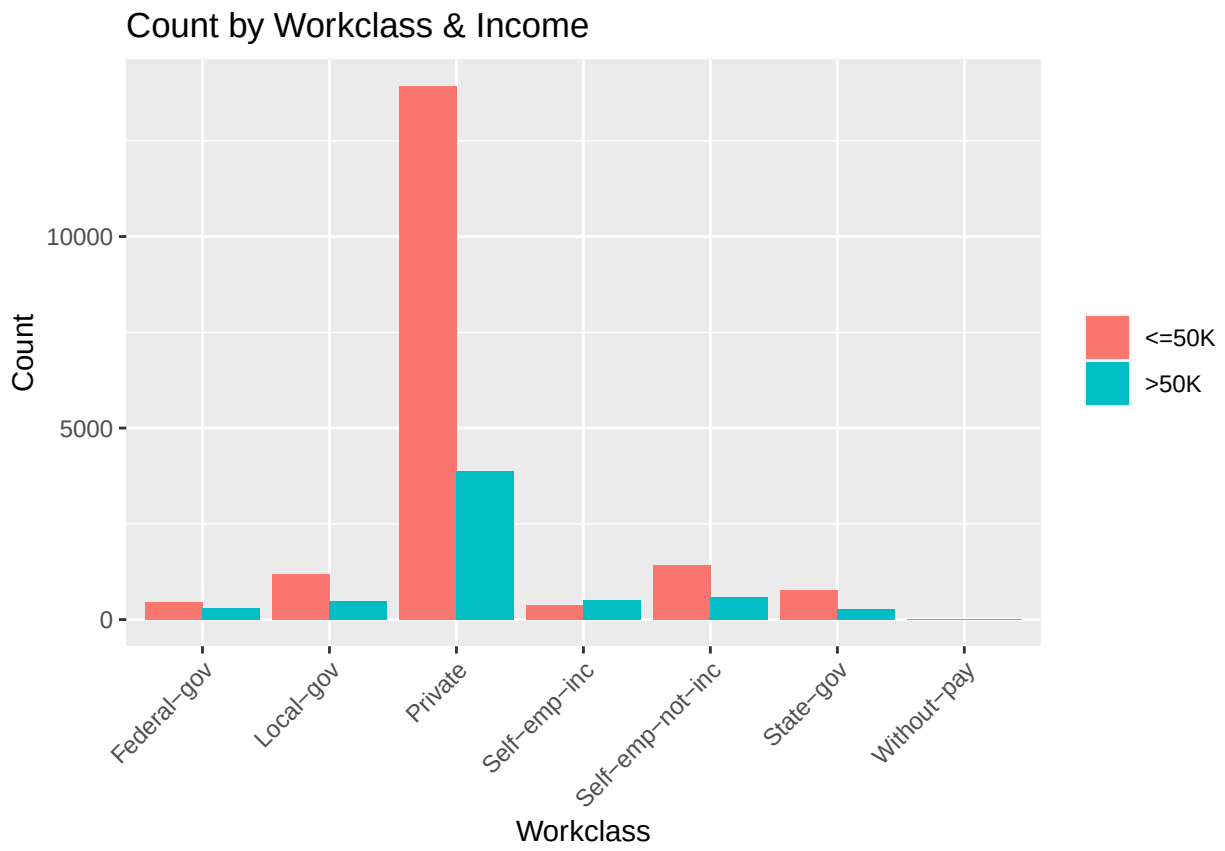
We can see here what is the education level of the dataset. Most repondant are high school diploma, then bachelor then some college.Very few have a doctorate which is normal.

This density plot is really interesting. We see that the peak of people earning less than 50k per year is around 25 years old. Thus, the form of the curve for people earning more than 50k a year have 2 peaks, around 35 years old for the first peak and 50 years old for the second peak.

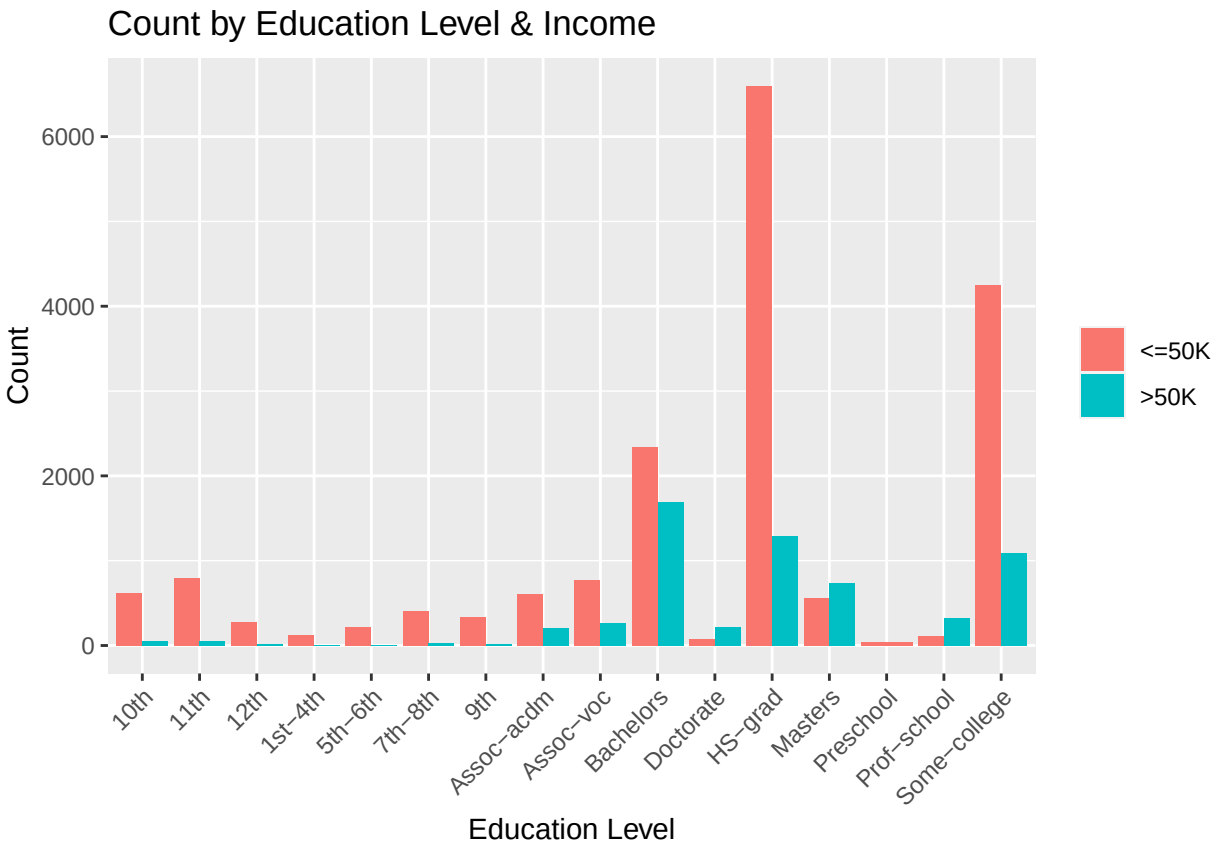


Not surprising the people who are divorced earn less than 50k per year. Very few earn more than 50.Most people who were never married don't earn more than 50k which I don't really know why, what could explain this ?

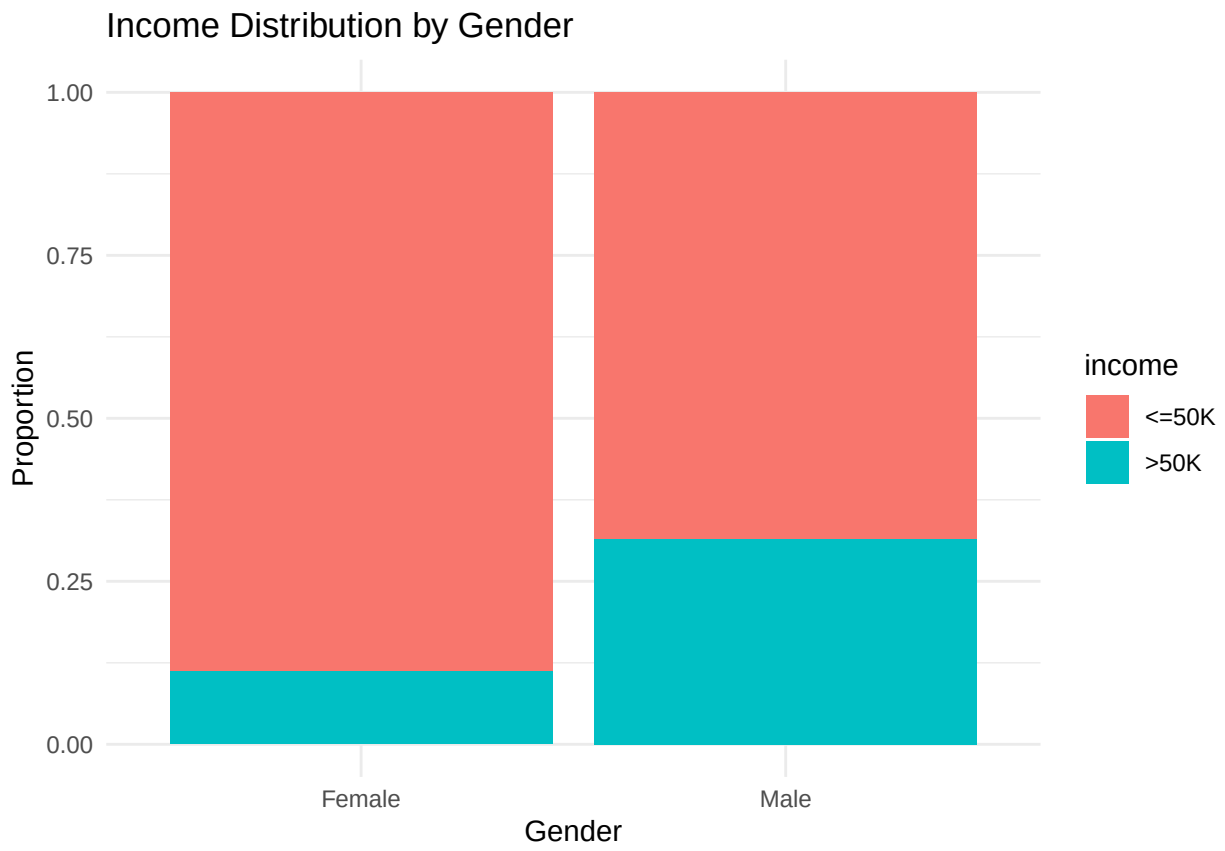
There is more men by a lot earning more than 50k then there is woman earning more than 50k. Also we you look on the left both woman and men earning less than 50k have a dataset right skewed.



People who are self employed earn more than 50k. People who earn more than 50k in the private sector are much rarer.

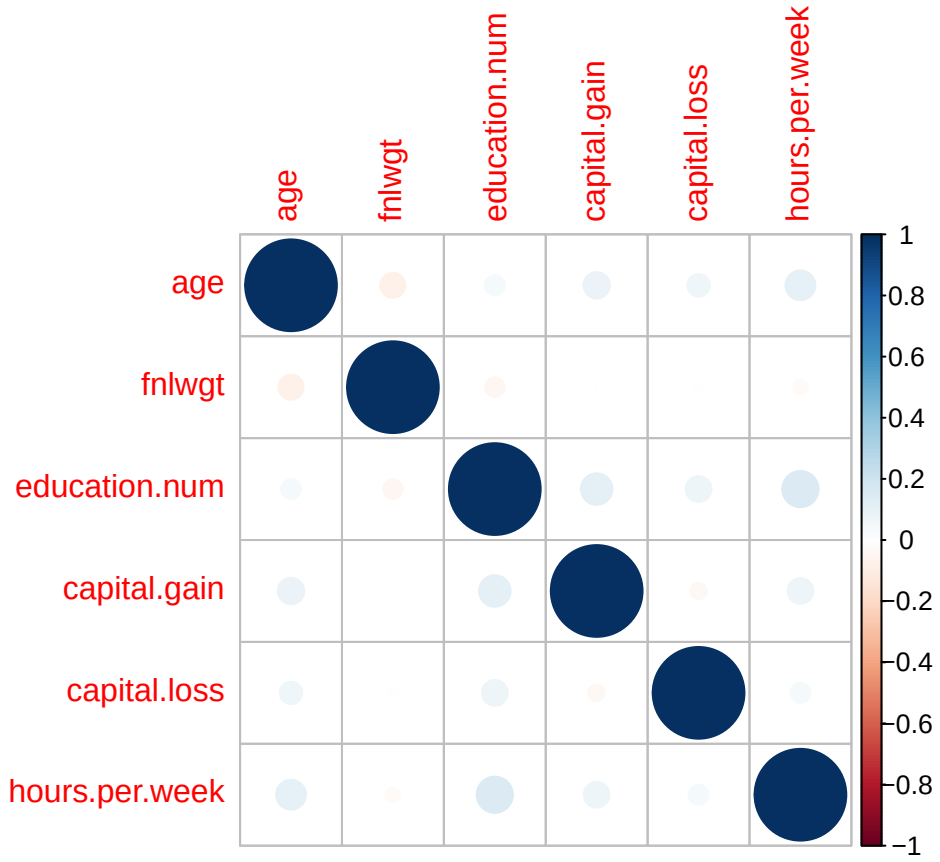


It is more frequent to see people who have a master earning more than 50k per year. It is also frequent to see people earning less than 50k who have no degree.



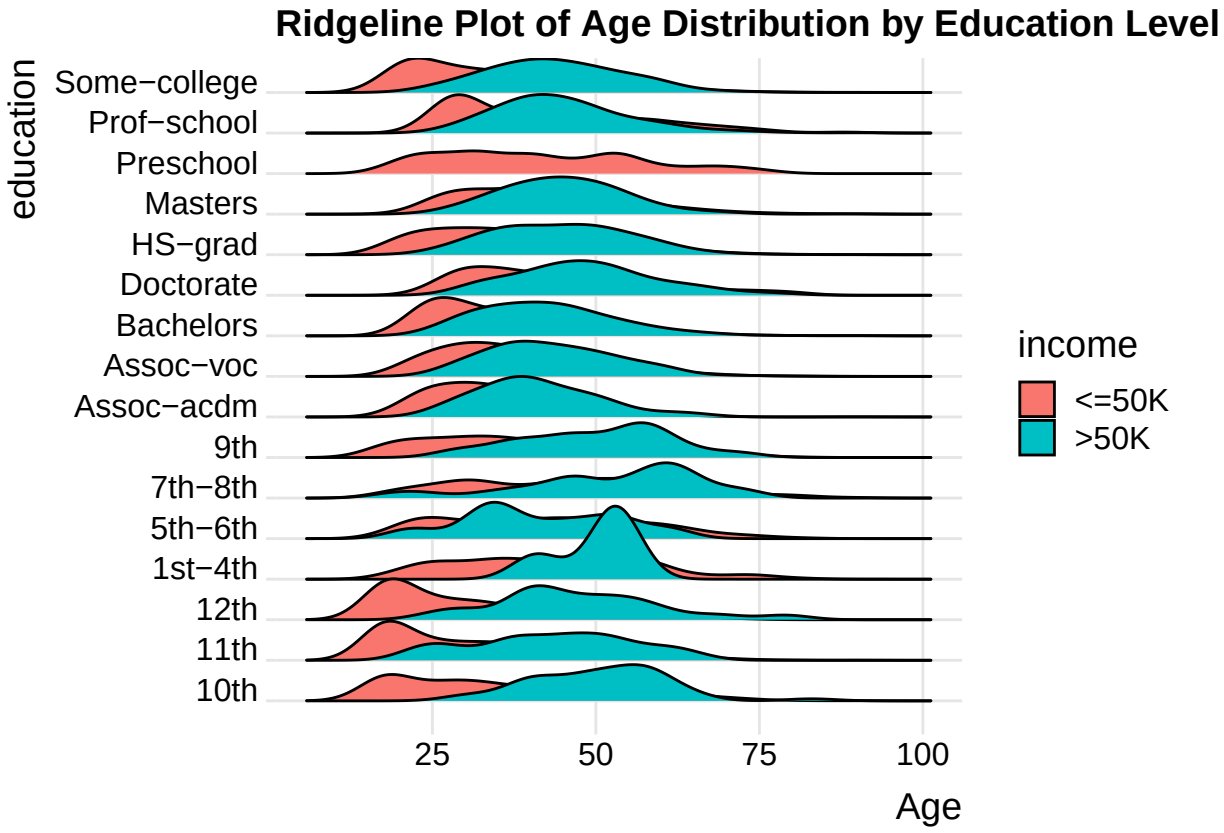
Here we see inequality of gender, female earn less than men in 1994, sadly.

No surprise executive and professor earn good salary.

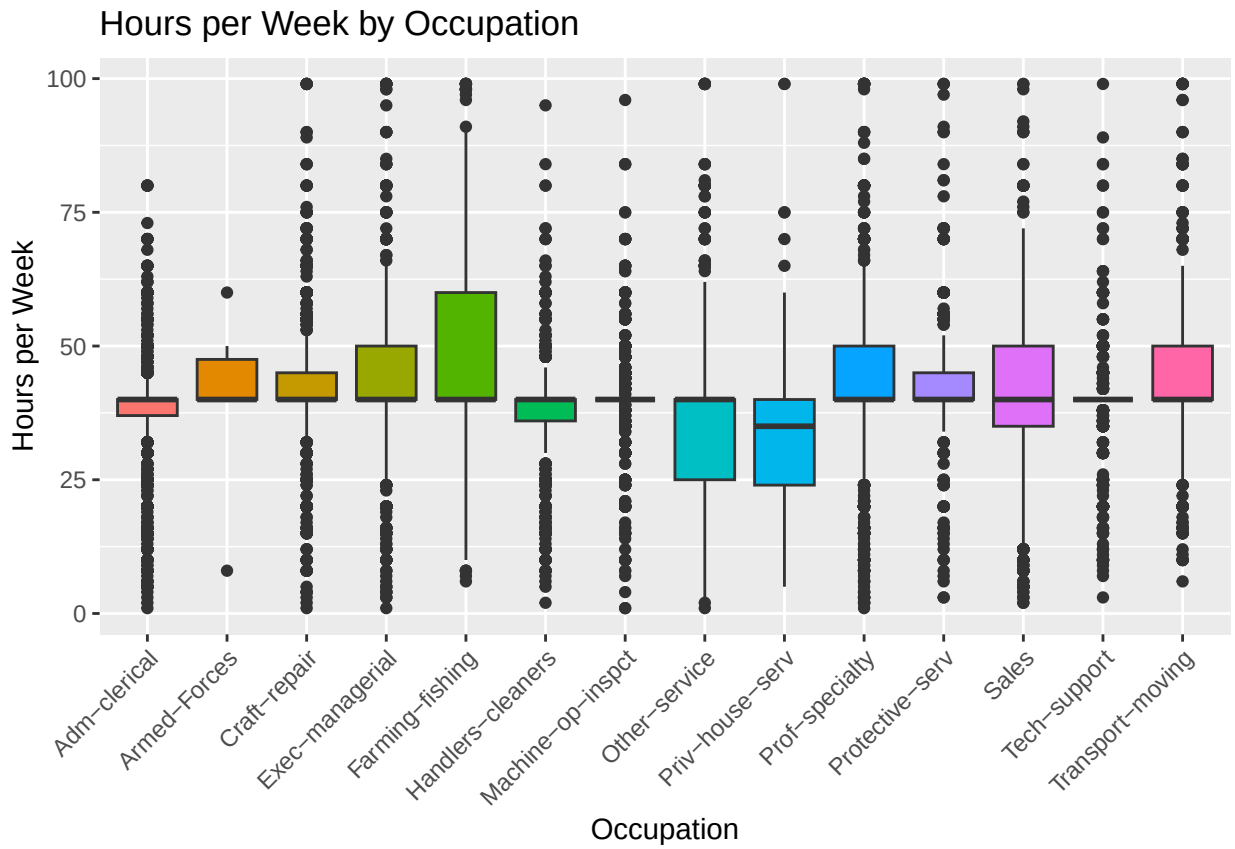


Some of the variable have correlation but no strong correlation were found.

```
## Picking joint bandwidth of 3.74
```

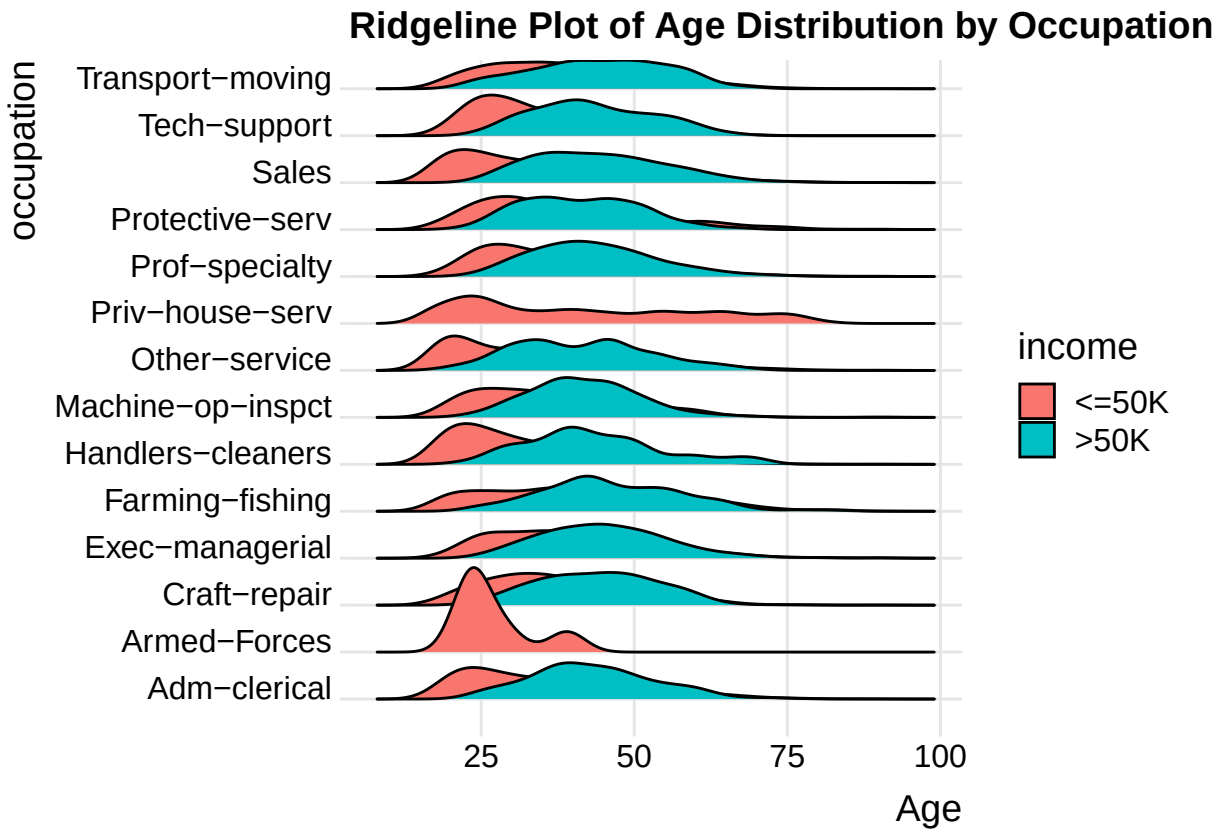


People who earn more than 50k and are aged less than 25 years old are really really rare.

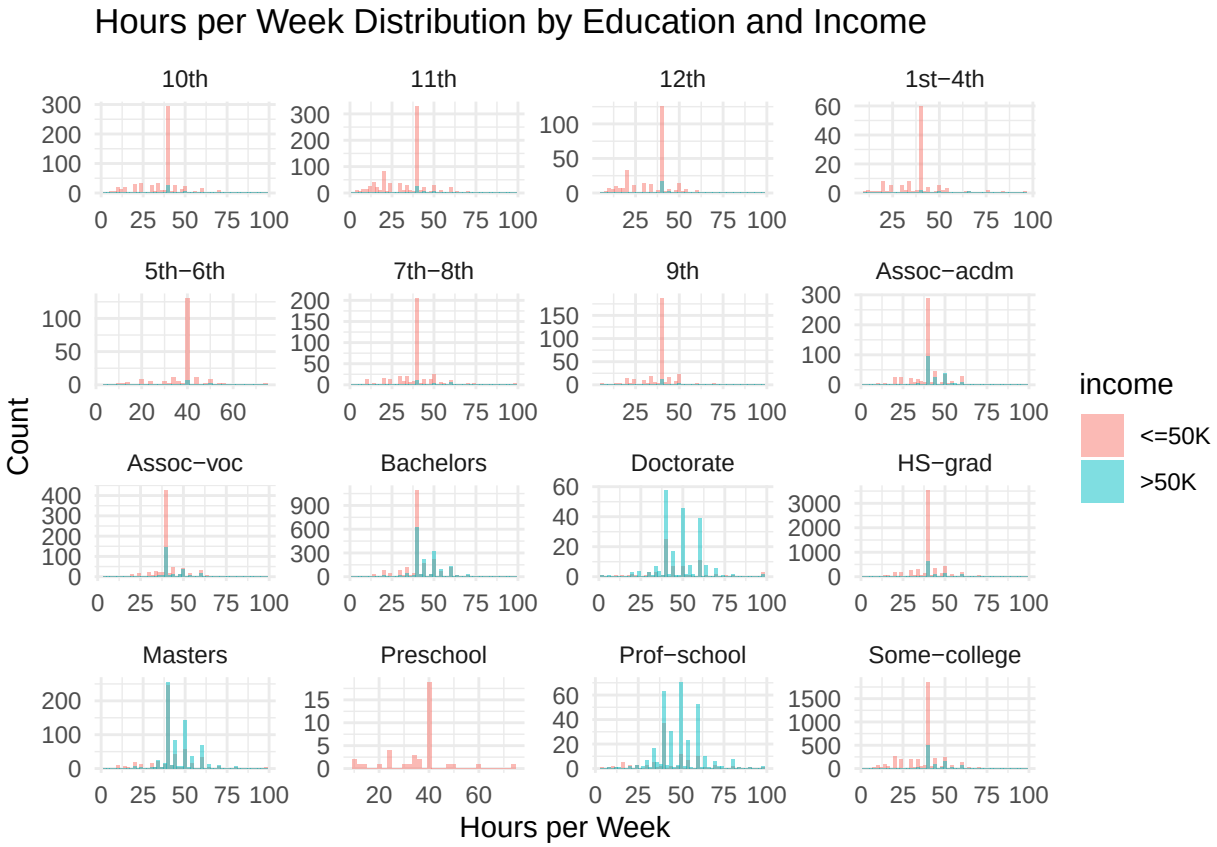


Farmer are the one who work the most.

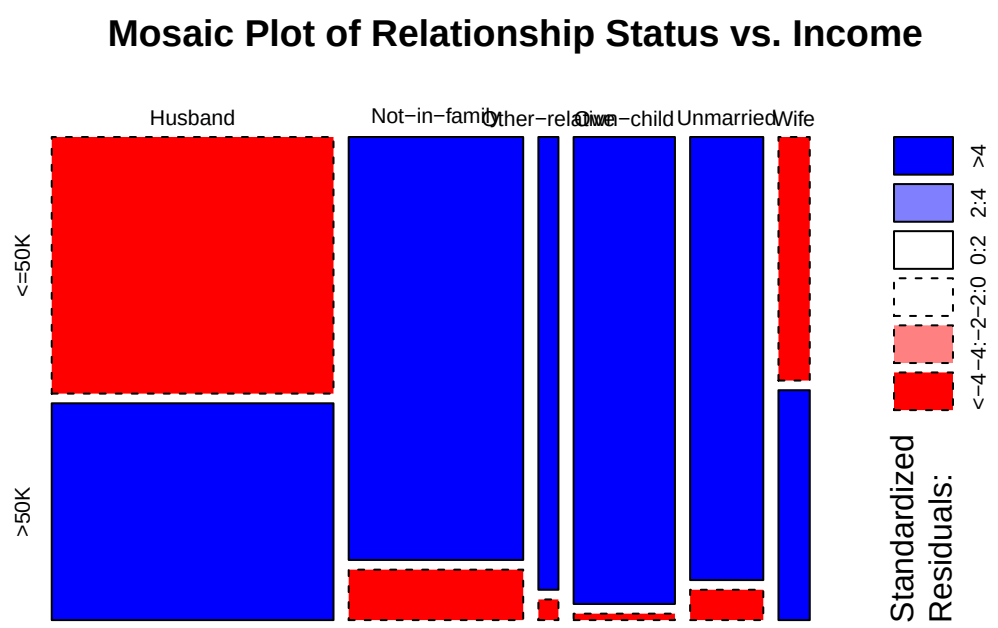
```
## Picking joint bandwidth of 3
```



People who protect our country are the one who earn the less. Crazy ! Doesn't make sense.



Occupation	Mean Hours per Week
Adm-clerical	37.63135
Armed-Forces	39.66667
Craft-repair	42.26073
Exec-managerial	45.09884
Farming-fishing	46.83709
Handlers-cleaners	37.89954
Machine-op-inspct	40.71586
Other-service	34.35425
Priv-house-serv	34.39496
Prof-specialty	42.38366
Protective-serv	42.71565
Sales	40.66122
Tech-support	39.32819
Transport-moving	44.43444



Result

Modeling Approach

Choice of Model

First model : Logistic Regression

In our initial approach to predicting income, we employed the Logistic Regression model. This statistical method is especially apt for binary outcomes, making it a fitting choice for our task of discerning between two income categories: those earning above $50K and those earning below. To implement this, the glm() function from R was utilized with a binomial family and a logit link function. After training the model on our training dataset, predictions were made on a separate validation set. The outcome of these predictions is dichotomized at the threshold of 0.5: values greater than this threshold are classified as incomes ">50K" while values below are designated as "<=50K". The confusionMatrix() function then offers a consolidated view of the model's performance, showing how often it correctly or incorrectly predicted each income category.

```
#model 1 : logistic regression
model_logistic <- glm(income ~ ., family=binomial(link='logit'), data=trainning_set)
```

```
## Warning: glm.fit: des probabilités ont été ajustées numériquement à 0 ou 1
```

```r
pred_logistic <- predict(model_logistic, newdata=validation_set, type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## les prédictions venant d'un modèle de rang faible peuvent être trompeuses
```

```r
result_logistic <- ifelse(pred_logistic > 0.5, ">50K", "<=50K")
confusionMatrix(factor(result_logistic), validation_set$income)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50K >50K
##      <=50K  4195  574
##      >50K    336  928
##
##                Accuracy : 0.8492
##                  95% CI : (0.8399, 0.8581)
##     No Information Rate : 0.751
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5741
##
##  Mcnemar's Test P-Value : 3.951e-15
##
##             Sensitivity : 0.9258
##             Specificity : 0.6178
##          Pos Pred Value : 0.8796
##          Neg Pred Value : 0.7342
##              Prevalence : 0.7510
##          Detection Rate : 0.6953
##    Detection Prevalence : 0.7905
##       Balanced Accuracy : 0.7718
##
##        'Positive' Class : <=50K
##
```

Second Model : Random Forest

For our next predictive strategy, we turned to the Random Forest algorithm. This method is essentially an ensemble of decision trees where each tree votes for a particular outcome, ensuring a more balanced and robust prediction. It's particularly effective for complex datasets with many features, as it can capture intricate patterns and interactions. With the randomForest() function in R, we constructed a forest consisting of 100 trees (ntree=100). After training on our dataset, predictions were generated for the validation set. The model's performance in terms of its accuracy in predicting the income categories is presented using the confusionMatrix(). The inherent strength of the Random Forest in handling overfitting and its capability to process a mix of categorical and numeric features made it a prime choice for our project.

```r
#model 2 : random forest
set.seed(123)
model_rf <- randomForest(income ~ ., data=trainning_set, ntree=100)
pred_rf <- predict(model_rf, newdata=validation_set)
confusionMatrix(pred_rf, validation_set$income)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50K >50K
##      <=50K  4184  515
##      >50K    347  987
##
##                Accuracy : 0.8571
##                  95% CI : (0.848, 0.8659)
##     No Information Rate : 0.751
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6031
##
##  Mcnemar's Test P-Value : 1.285e-08
##
##             Sensitivity : 0.9234
##             Specificity : 0.6571
##          Pos Pred Value : 0.8904
##          Neg Pred Value : 0.7399
##              Prevalence : 0.7510
##          Detection Rate : 0.6935
##    Detection Prevalence : 0.7789
##       Balanced Accuracy : 0.7903
##
##        'Positive' Class : <=50K
##
```

Third Model : XGBOOST

Our third modeling approach utilized the powerful gradient boosting framework, XGBOOST. This technique builds trees one at a time, where each new tree helps to correct errors made by the previously trained tree. By converting our data into a DMatrix, a special data structure that XGBOOST uses to boost its performance and efficiency, we prepped our dataset for the algorithm. We configured the model to use a gbtree booster for tree-based models and set the objective to "binary:logistic" given our two-class prediction problem. With parameters like eta determining the learning rate and depth controlling the depth of the trees, the model was trained over 100 rounds. Post-training, predictions on the validation set were derived and evaluated with a confusionMatrix(). The choice of XGBOOST for this dataset is motivated by its reputation for high performance and capability to optimize large-scale and complex data structures, making it an excellent fit for our project's requirements.

```
#model 3 : XGBOOST
dtrain <- xgb.DMatrix(data = model.matrix(~.-1, data=trainning_set[,!names(trainning_set) %in% c('income
dtest <- xgb.DMatrix(data = model.matrix(~.-1, data=validation_set[,!names(validation_set) %in% c('incom

params <- list(booster = "gbtree", objective = "binary:logistic", eta=0.3, depth=6)
set.seed(123)
model_xgb <- xgb.train(params = params, data = dtrain, nrounds=100)
```

```
## [18:14:58] WARNING: src/learner.cc:767:
## Parameters: { "depth" } are not used.
```

```r
pred_xgb <- predict(model_xgb, newdata=dtest)
result_xgb <- ifelse(pred_xgb > 0.5, ">50K", "<=50K")
confusionMatrix(factor(result_xgb), validation_set$income)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50K >50K
##      <=50K  4217  494
##      >50K    314 1008
##
##                Accuracy : 0.8661
##                  95% CI : (0.8572, 0.8746)
##     No Information Rate : 0.751
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6269
##
##  Mcnemar's Test P-Value : 3.031e-10
##
##             Sensitivity : 0.9307
##             Specificity : 0.6711
##          Pos Pred Value : 0.8951
##          Neg Pred Value : 0.7625
##              Prevalence : 0.7510
##          Detection Rate : 0.6990
##    Detection Prevalence : 0.7809
##       Balanced Accuracy : 0.8009
##
##        'Positive' Class : <=50K
##
```

Fourth Model : Support vector machine

For the fourth modeling endeavor, we employed the Support Vector Machine (SVM), a renowned algorithm tailored for classification problems. The SVM algorithm works by finding a hyperplane that best divides the dataset into classes. In our case, we used the 'C-classification' type, which is standard for two-class classification problems, and opted for a 'radial' kernel, which is versatile in handling non-linear data. Once the model was trained on our training set, we used it to predict incomes on the validation set. These predictions were subsequently assessed using a confusionMatrix(). The choice of SVM for this task is underpinned by its ability to efficiently handle high-dimensional data and its proficiency in classifying non-linear data, which aligns well with the intricacies of our dataset.

```r
#Model 4 : Support vector machine
model_svm <- svm(income ~ ., data=trainning_set, type='C-classification', kernel='radial')
pred_svm <- predict(model_svm, newdata=validation_set)
confusionMatrix(pred_svm, validation_set$income)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50K >50K
```

```
##      <=50K  4238  638
##      >50K    293  864
##
##                Accuracy : 0.8457
##                  95% CI : (0.8363, 0.8547)
##     No Information Rate : 0.751
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.553
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9353
##             Specificity : 0.5752
##          Pos Pred Value : 0.8692
##          Neg Pred Value : 0.7468
##              Prevalence : 0.7510
##          Detection Rate : 0.7025
##    Detection Prevalence : 0.8082
##       Balanced Accuracy : 0.7553
##
##        'Positive' Class : <=50K
##
```

Fifth Model : K-Nearest-Neighbors

For our fifth and final model, we delved into the K-Nearest Neighbors (KNN) algorithm, an instance-based learning method rooted in simplicity and intuitiveness. KNN operates by classifying an observation based on the majority class of its 'k' nearest observations in the dataset. Here, we set k to 21, indicating that for each person in the validation set, the algorithm looks at the 21 nearest individuals in the training set to decide the income category. To accommodate the algorithm's demands, we first transformed our data into matrices, ensuring it could be efficiently processed. Post prediction, the model's outcomes were evaluated using the confusionMatrix(). The choice of KNN is justified by its non-parametric nature, which means it makes no assumptions about the underlying data distribution. This capability renders it particularly fitting for datasets where the relationship between variables may be more complex or non-linear.

```
#Model 5 : K-Nearest Neighbors

train_data_matrix <- as.matrix(model.matrix(income ~ ., trainning_set)[, -1])
val_data_matrix <- as.matrix(model.matrix(income ~ ., validation_set)[, -1])
pred_knn <- knn(train_data_matrix, val_data_matrix, cl = trainning_set$income, k = 21)
confusionMatrix(pred_knn, validation_set$income)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50K >50K
##      <=50K  4462 1175
##      >50K     69  327
##
##                Accuracy : 0.7938
##                  95% CI : (0.7834, 0.8039)
##     No Information Rate : 0.751
```

```
##     P-Value [Acc > NIR] : 2.536e-15
##
##                   Kappa : 0.2686
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9848
##             Specificity : 0.2177
##          Pos Pred Value : 0.7916
##          Neg Pred Value : 0.8258
##              Prevalence : 0.7510
##          Detection Rate : 0.7396
##    Detection Prevalence : 0.9344
##       Balanced Accuracy : 0.6012
##
##        'Positive' Class : <=50K
##
```

Final table result

```
#TABLE OF EACH MODEL-------------------------------------
# Store the model predictions
predictions <- list(
  logistic = result_logistic,
  rf = pred_rf,
  xgboost = result_xgb,
  svm = pred_svm,
  knn = pred_knn
)
# Ensure the income variable in the validation set is a factor with specified levels
validation_set$income <- factor(validation_set$income, levels = c("<=50K", ">50K"))

# Convert the predictions to factors with the same levels as validation_set$income
predictions$logistic <- factor(predictions$logistic, levels = c("<=50K", ">50K"))
predictions$rf <- factor(predictions$rf, levels = c("<=50K", ">50K"))
predictions$xgboost <- factor(predictions$xgboost, levels = c("<=50K", "> 50K"))
predictions$svm <- factor(predictions$svm, levels = c("<=50K", ">50K"))
predictions$knn <- factor(predictions$knn, levels = c("<=50K", ">50K"))

# Initialize an empty dataframe to store results
model_performance <- data.frame(
  Model = character(),
  Accuracy = numeric(),
  F1_Score = numeric(),
  stringsAsFactors = FALSE
)

# Compute accuracy and F1 scores based on the predictions
for(model_name in names(predictions)) {
  cm <- confusionMatrix(predictions[[model_name]], validation_set$income)

  accuracy <- sum(diag(cm$table)) / sum(cm$table)
```

```
  recall <- cm$byClass['Recall']
  precision <- cm$byClass['Precision']

  f1 <- 2 * (precision * recall) / (precision + recall)

  model_performance <- rbind(model_performance, data.frame(Model = model_name, Accuracy = accuracy, F1_S
}
```

```
## Warning in confusionMatrix.default(predictions[[model_name]],
## validation_set$income): The data contains levels not found in the data, but
## they are empty and will be dropped.

## Warning in confusionMatrix.default(predictions[[model_name]],
## validation_set$income): Levels are not in the same order for reference and
## data. Refactoring data to match.
```

```
kable(model_performance,caption = "Table of all model result")
```

Table 2: Table of all model result

	Model	Accuracy	F1_Score
Precision	logistic	0.8491629	0.9021505
Precision1	rf	0.8571192	0.9066089
Precision2	xgboost	0.8951390	0.9446685
Precision3	svm	0.8456821	0.9010311
Precision4	knn	0.7938008	0.8776554

XGBoost has the highest accuracy of 0.8660699 and the highest F1 score of 0.9125730.

Therefore, XGBoost appears to be the best model among the ones tested, based on both accuracy and F1 score !

Conclusion

In conclusion, the models assessed, XGBoost emerged as the most promising, boasting an accuracy of 86.6%. Its gradient-boosted framework proved adept at managing our dataset's nuances, offering a harmonious balance between bias and variance. However, while this accuracy is commendable, it is crucial to remember that every model has its limitations.

Our analysis was bounded by the constraints of the dataset and the features at our disposal. Real-world scenarios might introduce variables and complexities not captured here. Additionally, an accuracy of 86.6% also indicates potential misclassifications, highlighting the continual need for model refinement and the incorporation of more comprehensive data to enhance predictive accuracy further. Future endeavors could delve into feature engineering, more advanced algorithms, or even ensemble methods to further bolster the predictive prowess.

References

UCI Machine Learning (2023). Adult Census Income. Kaggle. Retrieved from https://www.kaggle.com/datasets/uciml/adult-census-income.