CensusIncome

Sariq Sahazada

02/01/2021

INTRODUCTION:

This is a part of HarvardX PH125.9x Data Science: Capstone course as a Final Project. In this project we will build a model that will predict if the income of any individual in the US is greater than or less than USD 50,000 based on the data available about that individual.

This Census Income dataset was collected by Barry Becker in 1994 and given to the public site (http://archive.ics.uci.edu/ml/datasets/Census+Income). This data set will help you understand how the income of a person varies depending on various factors such as the education background, occupation, marital status, geography, age, number of working hours/week, etc.

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

```
## Loading required package: tidyverse

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

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

```
## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(gridExtra)) install.packages("gridExtra", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: gridExtra

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

IMPORTING THE DATA:

```r
train_file = "adult.data"; test_file = "adult.test"

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

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

If you take a look at the training data, you'll notice that the predictor variables are not labelled. Therefore, in the below code, I've assigned variable names to each predictor variable and to make the data more readable, I've gotten rid of unnecessary white spaces.

```r
#Assigning column names
colNames = c ("age", "workclass", "fnlwgt", "education",
"educationnum", "maritalstatus", "occupation",
"relationship", "race", "sex", "capitalgain",
"capitalloss", "hoursperweek", "nativecountry",
"incomelevel")

#Reading training data
train_set = read.table (train_file, header = FALSE, sep = ",",
strip.white = TRUE, col.names = colNames,
na.strings = "?", stringsAsFactors = TRUE)
#Reading testing data
test_set = read.table (test_file, header = FALSE, sep = ",",
strip.white = TRUE, col.names = colNames,
na.strings = "?", fill = TRUE, stringsAsFactors = TRUE)
test_set$age = as.integer(as.character(test_set$age))
```

```
## Warning: NAs introduced by coercion
```

```
test_set = na.omit(test_set)
```

Now in order to study the structure of our data sets, we call the str() method. This gives us a descriptive summary of all the predictor variables present in the data set:

```
#Display structure of the data
str(train_set)
```

```
## 'data.frame':    32561 obs. of  15 variables:
##  $ age          : int  39 50 38 53 28 37 49 52 31 42 ...
##  $ workclass    : Factor w/ 8 levels "Federal-gov",..: 7 6 4 4 4 4 4 6 4 4 ...
##  $ fnlwgt       : int  77516 83311 215646 234721 338409 284582 160187 209642 45781 159449 ...
##  $ education    : Factor w/ 16 levels "10th","11th",..: 10 10 12 2 10 13 7 12 13 10 ...
##  $ educationnum : int  13 13 9 7 13 14 5 9 14 13 ...
##  $ maritalstatus: Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 5 3 1 3 3 3 4 3 5 3 ...
##  $ occupation   : Factor w/ 14 levels "Adm-clerical",..: 1 4 6 6 10 4 8 4 10 4 ...
##  $ relationship : Factor w/ 6 levels "Husband","Not-in-family",..: 2 1 2 1 6 6 2 1 2 1 ...
##  $ race         : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 5 3 3 5 3 5 5 5 ...
##  $ sex          : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 1 1 2 1 2 ...
##  $ capitalgain  : int  2174 0 0 0 0 0 0 0 14084 5178 ...
##  $ capitalloss  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hoursperweek : int  40 13 40 40 40 40 16 45 50 40 ...
##  $ nativecountry: Factor w/ 41 levels "Cambodia","Canada",..: 39 39 39 39 5 39 23 39 39 39 ...
##  $ incomelevel  : Factor w/ 2 levels "<=50K",">50K": 1 1 1 1 1 1 1 2 2 2 ...
```

```
str(test_set)
```

```
## 'data.frame':    15060 obs. of  15 variables:
##  $ age          : int  25 38 28 44 34 63 24 55 65 36 ...
##  $ workclass    : Factor w/ 9 levels "","Federal-gov",..: 5 5 3 5 5 7 5 5 5 2 ...
##  $ fnlwgt       : int  226802 89814 336951 160323 198693 104626 369667 104996 184454 212465 ...
##  $ education    : Factor w/ 17 levels "","10th","11th",..: 3 13 9 17 2 16 17 7 13 11 ...
##  $ educationnum : int  7 9 12 10 6 15 10 4 9 13 ...
##  $ maritalstatus: Factor w/ 8 levels "","Divorced",..: 6 4 4 4 6 4 6 4 4 4 ...
##  $ occupation   : Factor w/ 15 levels "","Adm-clerical",..: 8 6 12 8 9 11 9 4 8 2 ...
##  $ relationship : Factor w/ 7 levels "","Husband","Not-in-family",..: 5 2 2 2 3 2 6 2 2 2 ...
##  $ race         : Factor w/ 6 levels "","Amer-Indian-Eskimo",..: 4 6 6 4 6 6 6 6 6 6 ...
##  $ sex          : Factor w/ 3 levels "","Female","Male": 3 3 3 3 3 3 2 3 3 3 ...
##  $ capitalgain  : int  0 0 0 7688 0 3103 0 0 6418 0 ...
##  $ capitalloss  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hoursperweek : int  40 50 40 40 30 32 40 10 40 40 ...
##  $ nativecountry: Factor w/ 41 levels "","Cambodia",..: 39 39 39 39 39 39 39 39 39 39 ...
##  $ incomelevel  : Factor w/ 3 levels "","<=50K.",">50K.": 2 2 3 3 2 3 2 2 3 2 ...
##  - attr(*, "na.action")= 'omit' Named int [1:1222] 1 6 8 15 21 24 37 67 77 85 ...
##   ..- attr(*, "names")= chr [1:1222] "1" "6" "8" "15" ...
```

DATA WRANGLING:

The data wrangling stage is considered to be one of the most time-consuming tasks in Data Science. This stage includes removing NA values, getting rid of redundant variables and any inconsistencies in the data.

We'll begin the data wrangling by checking if our data observations have any missing values:

```r
table(complete.cases(train_set))
```

```
##
## FALSE  TRUE
##  2399 30162
```

```r
table(complete.cases(test_set))
```

```
##
##  TRUE
## 15060
```

The above code indicates that 2399 sample cases have NA values. In order to fix this, let's look at the summary of all our variables and analyze which variables have the greatest number of null values. The reason why we must get rid of NA values is that they lead to wrongful predictions and hence decrease the accuracy of our model.

```r
summary(train_set[!complete.cases(train_set),])
```

```
##       age                   workclass        fnlwgt              education
##  Min.   :17.00   Private          : 410   Min.   : 12285   HS-grad     :661
##  1st Qu.:22.00   Self-emp-inc     :  42   1st Qu.:121804   Some-college:613
##  Median :36.00   Self-emp-not-inc:  42   Median :177906   Bachelors   :311
##  Mean   :40.39   Local-gov        :  26   Mean   :189584   11th        :127
##  3rd Qu.:58.00   State-gov        :  19   3rd Qu.:232669   10th        :113
##  Max.   :90.00   (Other)          :  24   Max.   :981628   Masters     : 96
##                  NA's             :1836                    (Other)     :478
##   educationnum                 maritalstatus           occupation
##  Min.   : 1.00   Divorced             :229   Prof-specialty : 102
##  1st Qu.: 9.00   Married-AF-spouse    :  2   Other-service  :  83
##  Median :10.00   Married-civ-spouse   :911   Exec-managerial:  74
##  Mean   : 9.57   Married-spouse-absent: 48   Craft-repair   :  69
##  3rd Qu.:11.00   Never-married        :957   Sales          :  66
##  Max.   :16.00   Separated            : 86   (Other)        : 162
##                  Widowed              :166   NA's           :1843
##           relationship                 race          sex         capitalgain
##  Husband       :730   Amer-Indian-Eskimo:  25   Female: 989   Min.   :    0.0
##  Not-in-family :579   Asian-Pac-Islander: 144   Male  :1410   1st Qu.:    0.0
##  Other-relative: 92   Black             : 307                 Median :    0.0
##  Own-child     :602   Other             :  40                 Mean   :  897.1
##  Unmarried     :234   White             :1883                 3rd Qu.:    0.0
##  Wife          :162                                           Max.   :99999.0
##
##   capitalloss       hoursperweek        nativecountry  incomelevel
##  Min.   :   0.00   Min.   : 1.00   United-States:1666   <=50K:2066
##  1st Qu.:   0.00   1st Qu.:25.00   Mexico       :  33   >50K : 333
##  Median :   0.00   Median :40.00   Canada       :  14
##  Mean   :  73.87   Mean   :34.23   Philippines  :  10
##  3rd Qu.:   0.00   3rd Qu.:40.00   Germany      :   9
##  Max.   :4356.00   Max.   :99.00   (Other)      :  84
##                                    NA's         : 583
```

From the above summary, it is observed that three variables have a good amount of NA values:

Workclass = 1836 Occupation = 1843 Nativecountry = 583

These three variables must be cleaned since they are significant variables for predicting the income level of an individual.

```
#Removing NAs
train_set = train_set[!is.na (train_set$workclass) & !is.na (train_set$occupation), ]
train_set = train_set[!is.na (train_set$nativecountry), ]

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

Once we've gotten rid of the NA values, our next step is to get rid of any unnecessary variable that isn't essential for predicting our outcome. It is important to get rid of such variables because they only increase the complexity of the model without improving its efficiency.

One such variable is the 'fnlwgt' variable, which denotes the population totals derived from CPS by calculating "weighted tallies" of any particular socio-economic characteristics of the population.

This variable is removed from our data set since it does not help to predict our resultant variable:

```
#Removing unnecessary variables

train_set$fnlwgt = NULL
test_set$fnlwgt = NULL
```

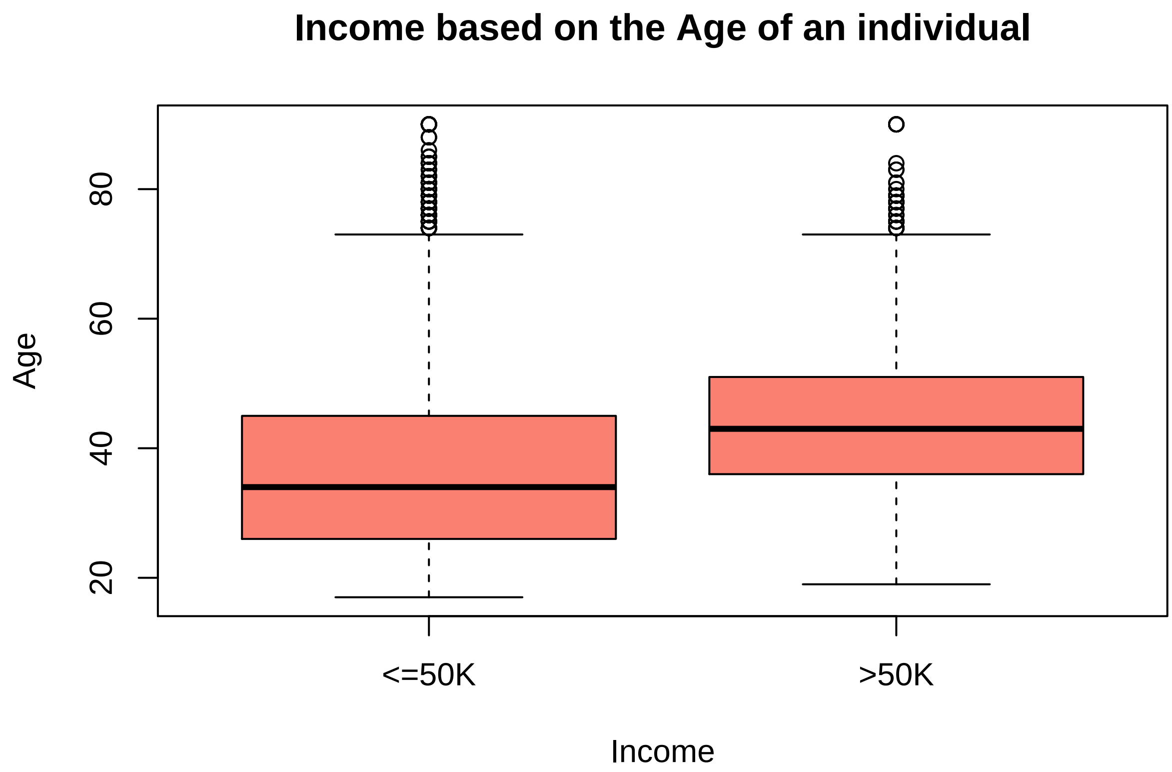
DATA VISUALIZATION:

Data Visualization involves analyzing each feature variable to check if the variables are significant for building the model.

```
#Data Visualization
#Visualizing the age variable

summary (train_set$age)
```

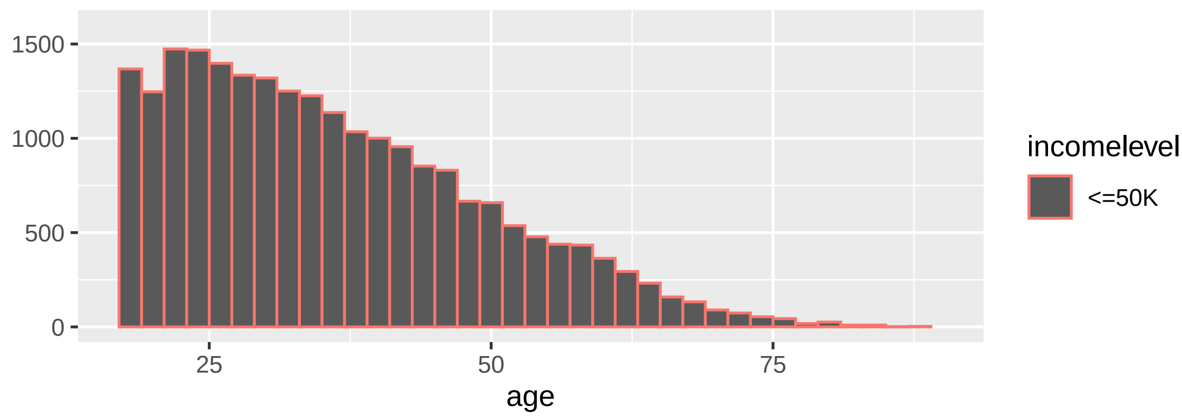
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   17.00   28.00   37.00   38.44   47.00   90.00
```

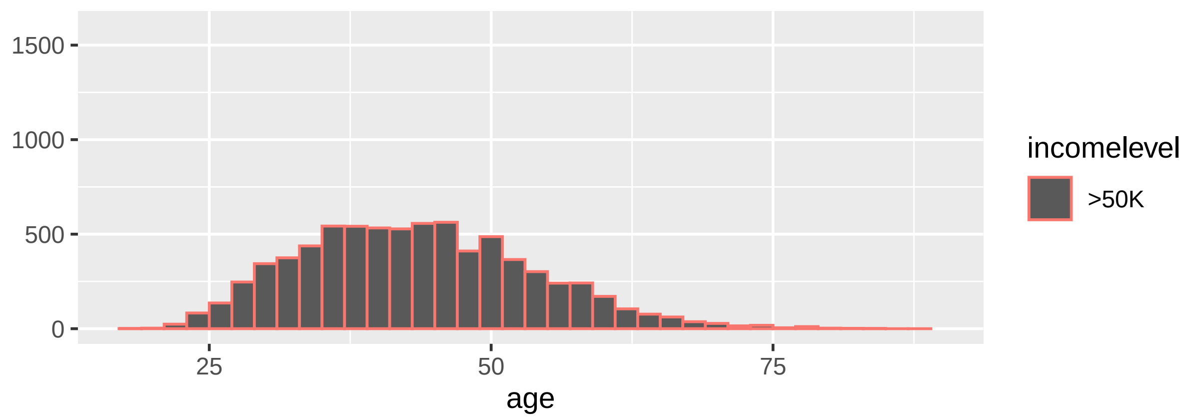
```
#Boxplot for age variable
boxplot (age ~ incomelevel, data = train_set,
main = "Income based on the Age of an individual",
xlab = "Income", ylab = "Age", col = "salmon")
```




```r
#Histogram for age variable
incomeBelow50K = (train_set$incomelevel == "<=50K")
xlimit = c (min (train_set$age), max (train_set$age))
ylimit = c (0, 1600)

hist1 = qplot(age, data = train_set[incomeBelow50K,], margins = TRUE,
binwidth = 2, xlim = xlimit, ylim = ylimit, colour = incomelevel)

hist2 = qplot(age, data = train_set[!incomeBelow50K,], margins = TRUE,
binwidth = 2, xlim = xlimit, ylim = ylimit, colour = incomelevel)

grid.arrange(hist1, hist2, nrow = 2)
```

```
## Warning: Removed 1 rows containing missing values (geom_bar).

## Warning: Removed 1 rows containing missing values (geom_bar).
```





The above illustrations show that the age variable is varying with the level of income and hence it is a strong predictor variable.

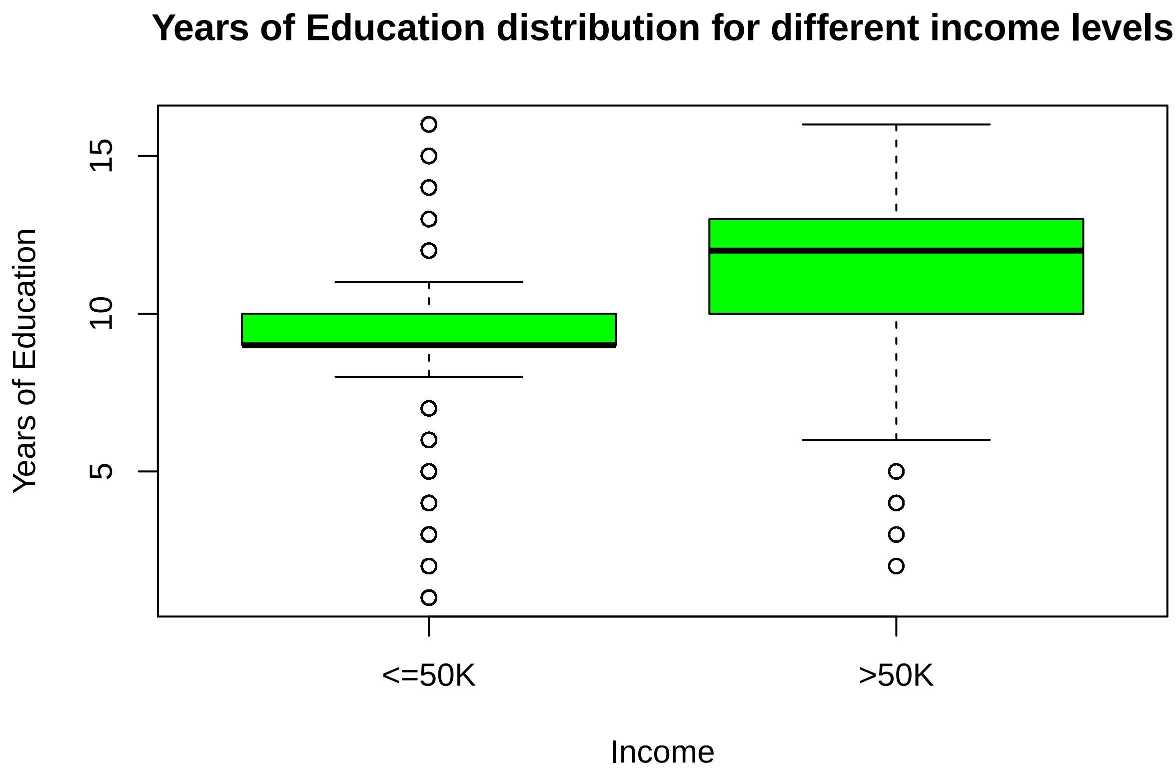
Visualizing the 'educationnum' variable:

This variable denotes the number of years of education of an individual. Let's see how the 'educationnum' variable varies with respect to the income levels:

```
summary (train_set$educationnum)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    9.00   10.00   10.12   13.00   16.00
```

```
#Boxplot for education-num variable
boxplot (educationnum ~ incomelevel, data = train_set,
main = "Years of Education distribution for different income levels",
xlab = "Income", ylab = "Years of Education", col = "green")
```



The above illustration depicts that the 'educationnum' variable varies for income levels <=50k and >50k, thus proving that it is a significant variable for predicting the outcome.

Visualizing capital-gain and capital-loss variable:

After studying the summary of the capital-gain and capital-loss variable for each income level, it is clear that their means vary significantly, thus indicating that they are suitable variables for predicting the income level of an individual.

```
summary (train_set[train_set$incomelevel == "<=50K",
                        c("capitalgain", "capitalloss")])
```

```
##   capitalgain        capitalloss
##  Min.   :    0.0   Min.   :   0.00
##  1st Qu.:    0.0   1st Qu.:   0.00
##  Median :    0.0   Median :   0.00
##  Mean   :  148.9   Mean   :  53.45
##  3rd Qu.:    0.0   3rd Qu.:   0.00
##  Max.   :41310.0   Max.   :4356.00
```

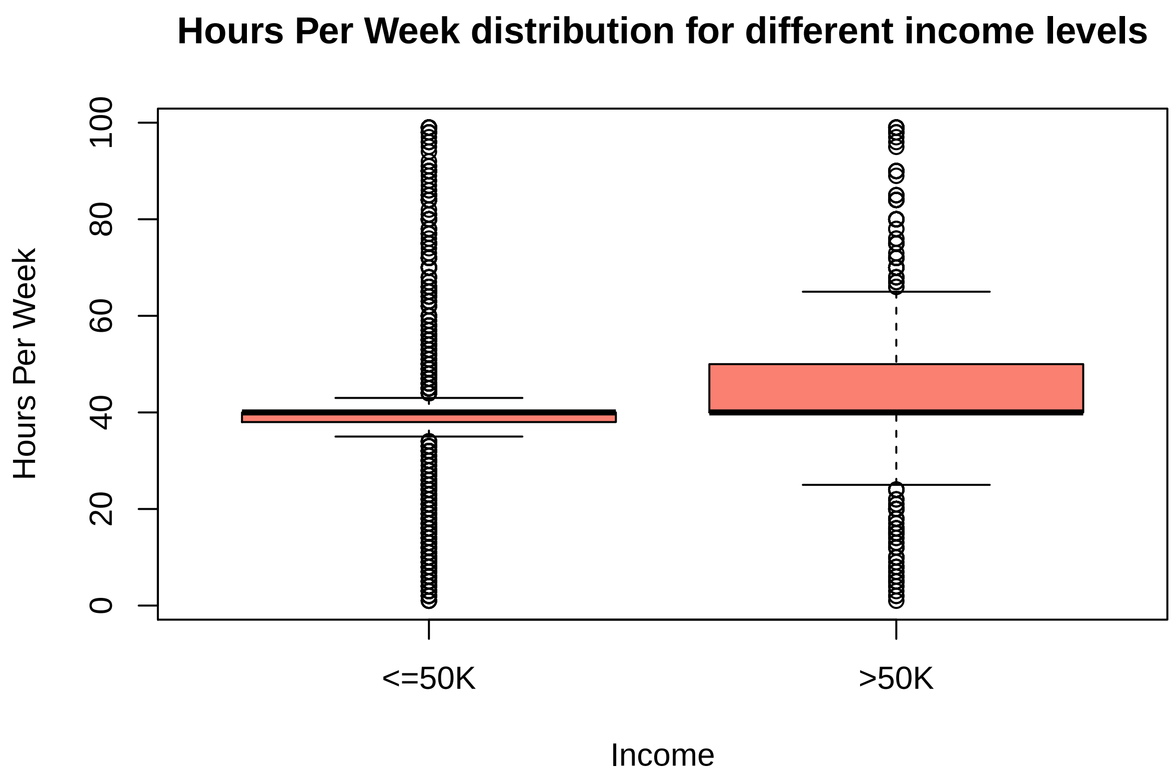
Exploring hours/week variable:

Similarly, the 'hoursperweek' variable is evaluated to check if it is a significant predictor variable.

```r
summary (train_set$hoursperweek)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   40.00   40.00   40.93   45.00   99.00
```

```r
boxplot (hoursperweek ~ incomelevel, data = train_set,
main = "Hours Per Week distribution for different income levels",
xlab = "Income", ylab = "Hours Per Week", col = "salmon")
```

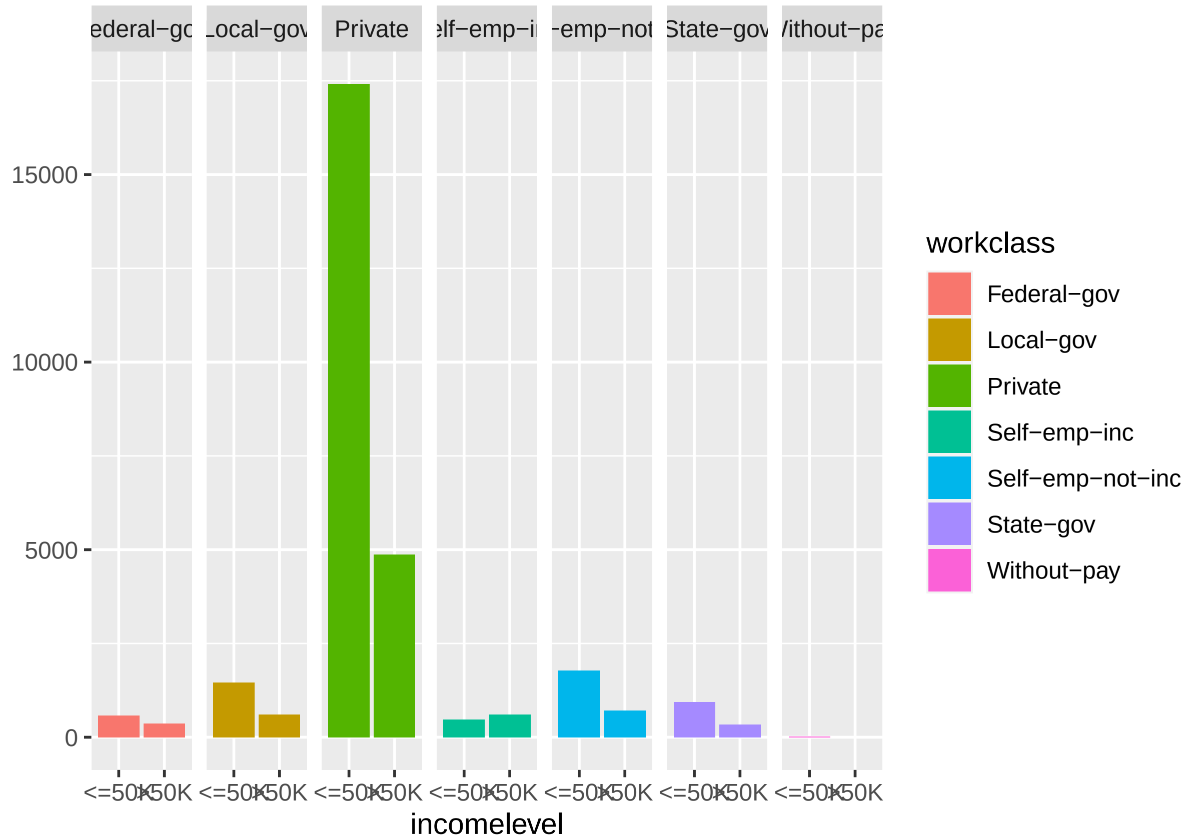


The boxplot shows a clear variation for different income levels which makes it an important variable for predicting the outcome.

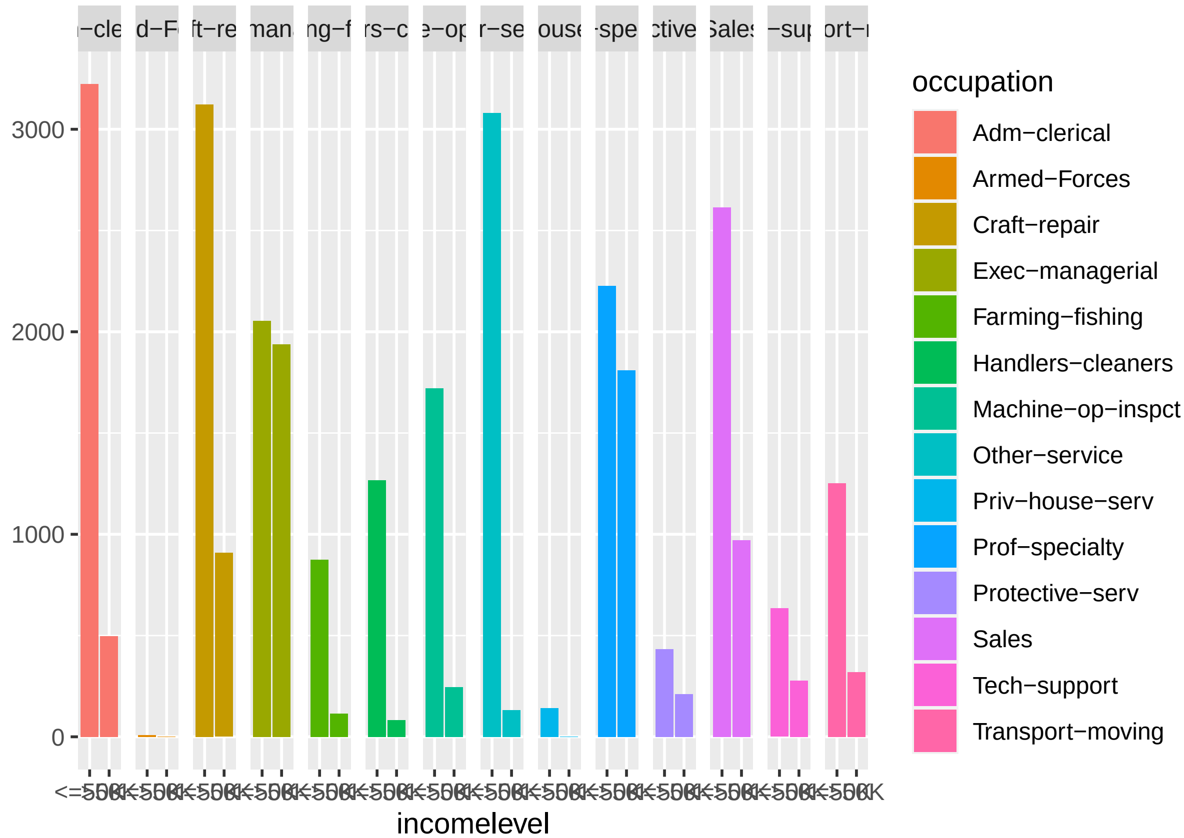
Similarly, we'll be evaluating categorical variables as well. In the below section I've created qplots for each variable and after evaluating the plots, it is clear that these variables are essential for predicting the income level of an individual.

Exploring work-class variable:

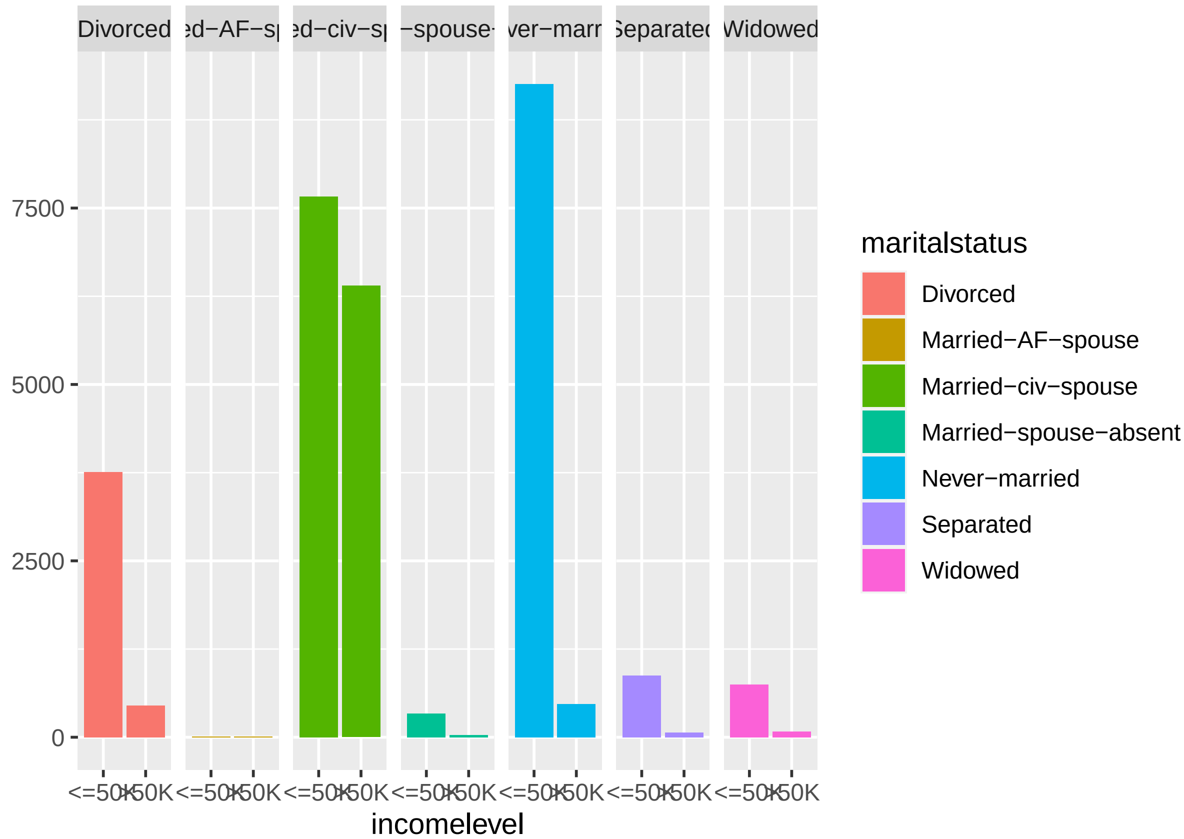
```r
#Evaluating work-class variable
qplot (incomelevel, data = train_set, fill = workclass) + facet_grid (. ~ workclass)
```



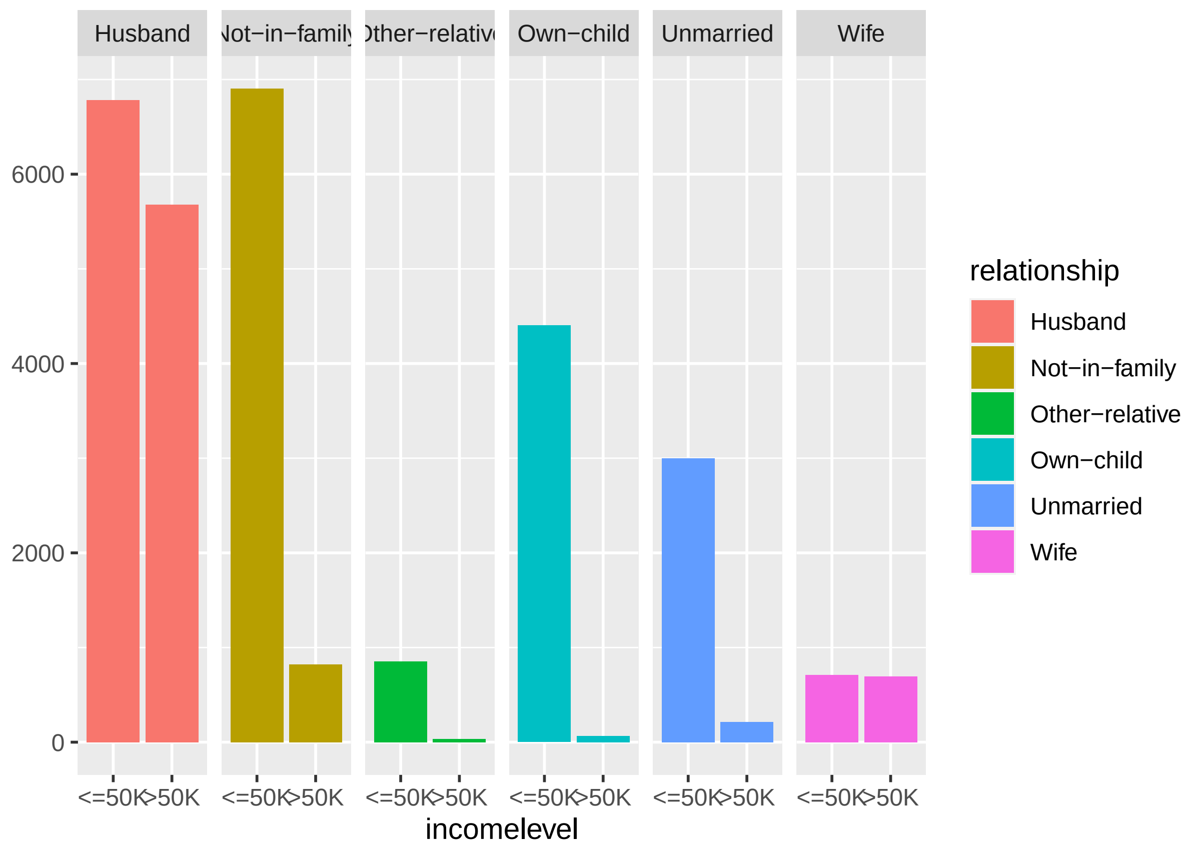

```r
#Evaluating occupation variable
qplot (incomelevel, data = train_set, fill = occupation) + facet_grid (. ~ occupation)
```




```r
#Evaluating marital-status variable
qplot (incomelevel, data =train_set, fill = maritalstatus) + facet_grid (. ~ maritalstatus)
```




```r
#Evaluating relationship variable
qplot (incomelevel, data = train_set, fill = relationship) + facet_grid (. ~ relationship)
```



All these graphs show that these set of predictor variables are significant for building our predictive model.

BUILDING MODEL:

So, after evaluating all our predictor variables, it is finally time to perform Predictive analytics. In this stage, we'll build a predictive model that will predict whether an individual earns above USD 50,000 or not based on the predictor variables that we evaluated in the previous section.

To build this model I've made use of the boosting and Random Forest algorithm since we have to classify an individual into either of the two classes, i.e:

Income level <= USD 50,000

Income level > USD 50,000

```
#Building the model: Boosting

set.seed (32323, sample.kind = "Rounding")
```

```
## Warning in set.seed(32323, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
trCtrl = trainControl(method = "cv", number = 10)

boostFit = train(incomelevel ~ age + workclass + education + educationnum +
```

```r
maritalstatus + occupation + relationship +
race + capitalgain + capitalloss + hoursperweek +
nativecountry, trControl = trCtrl,
method = "gbm", data = train_set, verbose = FALSE)
```

```
## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 69: nativecountryHoland-Netherlands has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 69: nativecountryHoland-Netherlands has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 69: nativecountryHoland-Netherlands has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.
```

```
## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.
```

```
## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =
## "bernoulli", : variable 3: workclassNever-worked has no variation.
```

Checking the Accuracy.

```
confusionMatrix(train_set$incomelevel, predict (boostFit, train_set))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50K  >50K
##      <=50K 21445  1209
##      >50K   2937  4571
##
##                Accuracy : 0.8625
##                  95% CI : (0.8586, 0.8664)
##     No Information Rate : 0.8084
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6017
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8795
##             Specificity : 0.7908
##          Pos Pred Value : 0.9466
##          Neg Pred Value : 0.6088
##              Prevalence : 0.8084
##          Detection Rate : 0.7110
##    Detection Prevalence : 0.7511
##       Balanced Accuracy : 0.8352
##
##        'Positive' Class : <=50K
##
```

```
#Building the model: Random Forest

set.seed(14, sample.kind = "Rounding")
```

```
## Warning in set.seed(14, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
train_rf <- train(incomelevel ~ age + workclass + education + educationnum +
                  maritalstatus + occupation + relationship +
                  race + capitalgain + capitalloss + hoursperweek +
                  nativecountry, method = "rf", ntree = 10,
                  tuneGrid = data.frame(mtry = seq(1:5)),
                  data = train_set)
```

Cheching the Accuracy

```
confusionMatrix (train_set$incomelevel, predict(train_rf, train_set))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50K  >50K
##      <=50K 21593  1061
##      >50K   3327  4181
##
##                Accuracy : 0.8545
##                  95% CI : (0.8505, 0.8585)
##     No Information Rate : 0.8262
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5673
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8665
##             Specificity : 0.7976
##          Pos Pred Value : 0.9532
##          Neg Pred Value : 0.5569
##              Prevalence : 0.8262
##          Detection Rate : 0.7159
##    Detection Prevalence : 0.7511
##       Balanced Accuracy : 0.8320
##
##        'Positive' Class : <=50K
##
```

TESTING THE MODEL:

Since with Boosting algorithm we got the highest accuracy. The test data is applied to the predictive model to validate the efficiency of the model.

```
#Testing model
test_set$predicted = predict(boostFit, test_set)
table(test_set$incomelevel, test_set$predicted)
```

```
##
##          <=50K  >50K
##              0     0
##   <=50K. 10735   625
##   >50K.   1463  2237
```

```
actuals_preds <- data.frame(cbind(actuals=test_set$incomelevel, predicted=test_set$predicted)) # make a
correlation_accuracy <- cor(actuals_preds)
head(actuals_preds)
```

```
##   actuals predicted
## 1       2         1
## 2       2         1
## 3       3         1
## 4       3         2
## 5       2         1
## 6       3         2
```

Defining RMSE i.e. Root Mean Square Error

```r
# Defining RMSE:
RMSE <- function(true, predicted){
  sqrt(mean((true - predicted)^2))
}
```

Now calculating the RMSE:

```r
RMSE(actuals_preds$actuals, actuals_preds$predicted)
```

```
## [1] 1.118004
```

CONCLUSION:

From the RMSEs and Accuracy of models, we can see that Boosting Algorithm improved the accuracy of the prediction.

US Census Income is a classical dataset which represents a challenge for development of better machine learning algorithm. In this project, the Random Forest model only gives an accuracy of 85%, and the Boosting model could improved it to 86%. In conclusion, Boosting algorithm appears to be a very powerful technique. The Ensemble method should also be considered in the future to apply on the US Census Income data set, in order to combine the advantages of various models and enhance the overall performance of prediction.