# HavardX PH125.9x Adult Census Income - Data Science Capstone CYO Project Report

*Ravi Jha*

*12/18/2019*

## Contents

# 1. Introduction

### 1.1 Overview

This report is prepared to fulfill the completion requirement of the **HavardX PH125.9x Data Science Capstone** course. In this project, an income classification system is created using the US Census 1994 dataset. A classification system is a subclass of an information filtering system that seeks to classify people using demographics to predict whether a person will earn an annual income over 50K dollars or not.

### 1.2 Project Goal

The goal of this project is to train machine learning algorithms (beyond standard linear regression) that predict whether income exceeds $50K per annum based on census data with an accuracy more than 80% and then compares their performance with other models.

### 1.3 Key steps performed

Following are the key steps performed in this project:

- Importing the dataset and setup
- Data wrangling
- Exploratory data analysis and visualization
- Spliting the curated dataset and processing
- Creating and tuning predective models
- Reporting results

# 2. Data and setup

The first step in the whole process is to import and setup data, before performing any data wrangling.

### 2.1 Adult Census Income DataSet

Let's start with the dataset, as per course instruction, Machine Learning analyses friendly **Adult Census Income** dataset, provided by UCI is used. The data was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics). The dataset is then cleaned considering the following criteria:

- age > 16
- final weighting > 1
- adjusted gross income (AGI) > $100
- working hours per week > 0

This dataset can be found and downloaded here:

- Adult Census Income dataset: http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data

### 2.2 Data Import and Initial Setup

To assist with our data import and initial setup, several packages from *CRAN* is utilized and loaded. These will be automatically downloaded and installed during code execution.

```
#######################################################
# to import and setup data
#######################################################

# Install requested packages if not found
if (!require(tidyverse))
  install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if (!require(caret))
  install.packages("caret", repos = "http://cran.us.r-project.org")
if (!require(data.table))
  install.packages("data.table", repos = "http://cran.us.r-project.org")
if (!require(e1071))
  install.packages("e1071", dependencies=TRUE,
                   repos = "http://cran.us.r-project.org")
if (!require(rpart))
  install.packages("rpart", repos = "http://cran.us.r-project.org")
if (!require(ranger))
  install.packages("ranger", dependencies=TRUE,
                   repos = "http://cran.us.r-project.org")
if (!require(dummies))
  install.packages("dummies", repos = "http://cran.us.r-project.org")
if (!require(dplyr))
  install.packages("dplyr")
if (!require(tidyr))
  install.packages("tidyr")

# Load libraries
library(tidyverse) # to untidy data
library(caret) # Classification And Regression
library(e1071) # SVM
library(ggplot2) # for plotting graphs
```

```r
library(dummies) # dummy variables (one hot encoder)
library(ranger) # fast Random Forest
library(rpart) # Decision Tree
library(class) # KNN
library(hexbin) # plotting
library(lubridate) # date
library(knitr) # LaTeX pdf
library(kableExtra) # LaTeX pdf

# Adult Census Income dataset
# http://archive.ics.uci.edu/ml/machine-learning-databases/adult

# to store objects from memory to the filesystem
datafile <- "Income.RData"

# to check if datafile is already downloaded
if (!file.exists("Income.RData"))
{
  fileURL <-
    "http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data"

  destfile_adult <- "./dataset/adult.data"

  # Download only if destination data file - adult.data
  # is not found on the dataset directory folder
  if (!file.exists(destfile_adult))
  {
    # Create dataset folder file if it doesn't exists
    if (!dir.exists("./dataset"))
    {
      dir.create("./dataset")
    }

    # Download file from url to temp file
    download.file(fileURL, destfile_adult,  quiet = TRUE)

    print("Downloading file to the dataset directory")

  }
  # to name dataset columns
  c_names <-
    c("age", "workclass", "fnlwgt", "education", "education-num",
      "marital-status", "occupation", "relationship", "race", "sex",
      "capital-gain", "capital-loss", "hours-per-week", "native-country", "income")


  # to read the data from the file and store in the census_data object
  census_data <-
    read.csv(file = "./dataset/adult.data", header = FALSE,
             col.names = c_names, stringsAsFactors = T, strip.white = TRUE)

  # replace . in the columns names
  colnames(census_data) <- str_replace_all(colnames(census_data), "\\.", "_")
```

```
    # to Save census_data object to datafile
    save(census_data, file = datafile)

    # Remove unused objects from the memory
    rm(c_names, fileURL, datafile, destfile_adult)

  } else {
    # lo Load the datafile if it already exists
    load(datafile)
  }
```

# 3. Data Wrangling

The next step is to understand the data through a brief data review and then do data wrangling as per the necessity.

### 3.1 Initial exploration

Before proceeding further with analysis, examining the data is important. The first step in it, is to understand the format and the contents by looking at the few rows.

```
    # Display first few rows to understand the data structure
    glimpse(census_data)
```

```
## Observations: 32,561
## Variables: 15
## $ age            <int> 39, 50, 38, 53, 28, 37, 49, 52, 31, 42, 37, 30,...
## $ workclass      <fct> State-gov, Self-emp-not-inc, Private, Private, ...
## $ fnlwgt         <int> 77516, 83311, 215646, 234721, 338409, 284582, 1...
## $ education      <fct> Bachelors, Bachelors, HS-grad, 11th, Bachelors,...
## $ education_num  <int> 13, 13, 9, 7, 13, 14, 5, 9, 14, 13, 10, 13, 13,...
## $ marital_status <fct> Never-married, Married-civ-spouse, Divorced, Ma...
## $ occupation     <fct> Adm-clerical, Exec-managerial, Handlers-cleaner...
## $ relationship   <fct> Not-in-family, Husband, Not-in-family, Husband,...
## $ race           <fct> White, White, White, Black, Black, White, Black...
## $ sex            <fct> Male, Male, Male, Male, Female, Female, Female,...
## $ capital_gain   <int> 2174, 0, 0, 0, 0, 0, 0, 0, 14084, 5178, 0, 0, 0...
## $ capital_loss   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ hours_per_week <int> 40, 13, 40, 40, 40, 40, 16, 45, 50, 40, 80, 40,...
## $ native_country <fct> United-States, United-States, United-States, Un...
## $ income         <fct> <=50K, <=50K, <=50K, <=50K, <=50K, <=50K, <=50K...
```

The dataset has 32,561 records and 15 attributes.

#### Dataset Summary

Summry of the dataset can be seen by the following snippet:

```
    # Summary of the dataset
    summary(census_data)
```

```
##       age                    workclass         fnlwgt
##  Min.   :17.00   Private         :22696   Min.   :  12285
##  1st Qu.:28.00   Self-emp-not-inc: 2541   1st Qu.: 117827
##  Median :37.00   Local-gov       : 2093   Median : 178356
##  Mean   :38.58   ?               : 1836   Mean   : 189778
##  3rd Qu.:48.00   State-gov       : 1298   3rd Qu.: 237051
##  Max.   :90.00   Self-emp-inc    : 1116   Max.   :1484705
##                  (Other)         :  981
##         education     education_num                  marital_status
##  HS-grad     :10501   Min.   : 1.00   Divorced             : 4443
##  Some-college: 7291   1st Qu.: 9.00   Married-AF-spouse    :   23
##  Bachelors   : 5355   Median :10.00   Married-civ-spouse   :14976
##  Masters     : 1723   Mean   :10.08   Married-spouse-absent:  418
##  Assoc-voc   : 1382   3rd Qu.:12.00   Never-married        :10683
##  11th        : 1175   Max.   :16.00   Separated            : 1025
##  (Other)     : 5134                   Widowed              :  993
##            occupation           relationship                   race
```

```
##  Prof-specialty :4140   Husband       :13193   Amer-Indian-Eskimo:  311
##  Craft-repair   :4099   Not-in-family : 8305   Asian-Pac-Islander: 1039
##  Exec-managerial:4066   Other-relative:  981   Black             : 3124
##  Adm-clerical   :3770   Own-child     : 5068   Other             :  271
##  Sales          :3650   Unmarried     : 3446   White             :27816
##  Other-service  :3295   Wife          : 1568
##  (Other)        :9541
##      sex         capital_gain    capital_loss    hours_per_week
##  Female:10771   Min.   :    0   Min.   :   0.0   Min.   : 1.00
##  Male  :21790   1st Qu.:    0   1st Qu.:   0.0   1st Qu.:40.00
##                 Median :    0   Median :   0.0   Median :40.00
##                 Mean   : 1078   Mean   :  87.3   Mean   :40.44
##                 3rd Qu.:    0   3rd Qu.:   0.0   3rd Qu.:45.00
##                 Max.   :99999   Max.   :4356.0   Max.   :99.00
##
##        native_country    income
##  United-States:29170   <=50K:24720
##  Mexico       :  643   >50K : 7841
##  ?            :  583
##  Philippines  :  198
##  Germany      :  137
##  Canada       :  121
##  (Other)      : 1709
```

### Column types and class

Let's check column types and their class.

```
# To display dataset column types and summary
col_typ <- as.data.frame(lapply(census_data, class)) %>%
  .[1, ] %>%
  gather(variable, class , 1:ncol(.)) %>%
  mutate(data_type = ifelse(
    class == "factor",
    "categorical",
    ifelse(variable == "education_num", "categorical", "continuous")
  ))
col_typ %>%
  # to format table with theme
  kable() %>%
  kable_styling(latex_options = c("striped", "hover", "condensed")) %>%
  row_spec(0, bold = T)
```

| variable | class | data_type |
|---|---|---|
| age | integer | continuous |
| workclass | factor | categorical |
| fnlwgt | integer | continuous |
| education | factor | categorical |
| education_num | integer | categorical |
| marital_status | factor | categorical |
| occupation | factor | categorical |
| relationship | factor | categorical |
| race | factor | categorical |
| sex | factor | categorical |
| capital_gain | integer | continuous |
| capital_loss | integer | continuous |
| hours_per_week | integer | continuous |
| native_country | factor | categorical |
| income | factor | categorical |

### 3.2 Tidying data

Based on the initial understanding further data analysis is required to identify data cleaning necessities. Following are the steps to tidy the data:

#### Duplicate Records

Let's first check for any duplicate record. The following table shows there are 24 duplicate rows.

```
# to check for duplicate records
census_data %>%
  summarize(record_count = n(),
            distinct_records = n_distinct(.)) %>%
  mutate(duplicate_records = record_count - distinct_records) %>%
  mutate(duplicate_percent =
           paste0(round(duplicate_records / record_count * 100, 1), " %")) %>%
  # to format table with theme
  kable() %>%
  kable_styling(latex_options = c("striped", "hover", "condensed")) %>%
  row_spec(0, bold = T)
```

| record_count | distinct_records | duplicate_records | duplicate_percent |
|---|---|---|---|
| 32561 | 32537 | 24 | 0.1 % |

#### Missing Data

By closely looking at the data, it can be observed that there are columns with value "?". This is also supported by the dataset details available on UCI website that **unknown** values are labeled as **?**.

As it can be seen from the below table, there are 2,399 incomplete records. Precisely, there are missing values in `workclass`, `occupation`, and `native_country` columns. The table below displays the missing columns and record statistics.

```
#to check if any explicit missing value marked with NA
  anyNA(census_data)
```

```
## [1] FALSE
```

```r
# to check missing values labeled as "?"

# first check for the records with any missing value (?)
missing_count_tbl <-
  map_df(census_data, ~ str_detect(., pattern = "\\?")) %>%
  rowSums() %>%
  tbl_df() %>%
  filter(value > 0) %>%
  summarize(missing_count = n()) %>%
  # to create a table to display the missing count and percent
  mutate(missing_percent =
           paste0(round(missing_count / nrow(census_data) * 100, 1), "%"),
         Column = "Record with any missing columns") %>%

  select(Column, missing_count, missing_percent)

  # now check for individual columns with missing values
  # and bind it with the existing table

  map_df(census_data, ~ sum(str_detect(., pattern = "\\?"))) %>%

  gather(Column, missing_count, 1:15) %>%
  mutate(missing_percent =
           paste0(round(missing_count / nrow(census_data) * 100, 1), "%")) %>%
  bind_rows(missing_count_tbl) %>%
  # filter-out columns with no missing value
  filter(missing_count > 1) %>%
  # arrange rows by missing count
  arrange(desc(missing_count)) %>%
  # to format table with theme
  kable() %>%
  kable_styling(latex_options = c("striped", "hover", "condensed")) %>%
  row_spec(0, bold = T)
```

| Column | missing_count | missing_percent |
|---|---|---|
| Record with any missing columns | 2399 | 7.4% |
| occupation | 1843 | 5.7% |
| workclass | 1836 | 5.6% |
| native_country | 583 | 1.8% |

### Impute missing values

To prepare the dataset for analysis, getting rid if missing values in an important step. The easiest method is to replace the missing values with the mean for numerical variables; however, it makes more sense to remove records with missing values for categorical attribtes using the following code:

```r
# first to convert missing values ? to NA
# for workclass, occupation, and native_country
census_data$workclass<-ifelse(census_data$workclass=='?',
                              NA, as.character(census_data$workclass))
census_data$occupation<-ifelse(census_data$occupation=='?',
                               NA, as.character(census_data$occupation))
census_data$native_country<-ifelse(census_data$native_country=='?',
                                   NA, as.character(census_data$native_country))
```

```
# next remove rows with missing values
census_data <- na.omit(census_data)
```

### Combine Marital Status column catagories

Next, collalte `marital_status` column values into 3 categories - *Married, Not-married, Never-married* using the following code:

```
# to combine the column values catagories
census_data$marital_status <- as.character(census_data$marital_status)

# to create two new categories, Never-married is already present
Married <- c("Married-AF-spouse", "Married-civ-spouse", "Married-spouse-absent")
Notmarried <- c("Divorced","Separated", "Widowed")

# to update the marital_status column with the new catagories
census_data$marital_status[census_data$marital_status %in% Married] <-
  "Married"
census_data$marital_status[census_data$marital_status %in% Notmarried] <-
  "Not-married"

# to display the table
census_data %>%
  group_by(marital_status) %>%
  summarise(count =n(),
            proportion = paste0(round(count/nrow(census_data)*100, digits = 2),"%")) %>%
  arrange(desc(count)) %>%
  kable() %>%
  kable_styling(latex_options = c("striped", "hover", "condensed"))  %>%
  row_spec(0, bold = T)
```

| marital_status | count | proportion |
|---|---|---|
| Married | 14456 | 47.93% |
| Never-married | 9726 | 32.25% |
| Not-married | 5980 | 19.83% |

### Remove Education_Num column

`education` and `education_num` variables represent the same data with `education_num` as ordinal representation. We can get rid of `education_num` to clean the data using the following code:

```
# to remove education_num column
census_data <- census_data %>%
  select(-education_num)
```

### Combine Capital Gain and Capital Loss columns

Next, the `capital_gain` and `capital_loss` columns can be combined into a single column `capital`. Further, `capital_gain` and `capital_loss` columns can be removed to aid the analysis, using the following code:

```
# to combine capital gain and capital loss columns
# into a single column Capital,
# a positive value represents gain and negative represents loss
census_data <- census_data %>%
  mutate(capital = capital_gain - capital_loss) %>%
```

```
  # then remove capital_gain and capital_loss
  select(-capital_gain, -capital_loss)
```

### Remove Final Weight column

The final weight is represented by `fnlwgt` column, adjusts the weight as per population size of the number of people in the census data. For example, a record gets a high weight if the proportion of such samples is comparatively small in the overall population and vice-versa. This is not useful in the analysis, hence, can be removed.

```
# to remove fnlwgt column
census_data <- census_data %>%
  select(-fnlwgt)
```

### Combine Native Country column catagories

There are too many categories in `native_country` column, it can be reduced to their respective regions and key countries can be kept. The countries are categorized among their respective regions and leaving key countries such as the *US* to see if native to the countries make any difference to the income classification.

```
# to combine the column values catagories

# following are the new catogories
LatinAmerica <- c("Dominican-Republic","Guatemala","Haiti","Honduras",
                  "Jamaica","Mexico","Nicaragua", "Outlying-US(Guam-USVI-etc)",
                  "Puerto-Rico","Trinadad&Tobago","Cuba")

SouthAmerica <- c("Peru","Ecuador","El-Salvador","Columbia")

Europe <- c("France","Germany","Greece","Holand-Netherlands",
            "Hungary","Italy","Poland", "Portugal","Puerto-Rico",
            "South","Ireland","Yugoslavia")

Asia <- c("China","Hong","India","Japan","Iran")

SE_Aisa <- c("Vietnam","Cambodia","Thailand","Laos","Philippines","Taiwan")

US <- c("United-States")
UK <- c("England","Scotland")
Canada <- c("Canada")

# to update the native_country column with the new catagories
census_data$native_country[census_data$native_country %in% LatinAmerica] <-
  "Latin America"
census_data$native_country[census_data$native_country %in% Asia] <-
  "Asia"
census_data$native_country[census_data$native_country %in% SE_Aisa] <-
  "South East Aisa"
census_data$native_country[census_data$native_country %in% SouthAmerica] <-
  "South America"
census_data$native_country[census_data$native_country %in% Europe] <-
  "Europe"
census_data$native_country[census_data$native_country %in% US] <-
  "US"
census_data$native_country[census_data$native_country %in% UK] <-
```

```
  "UK"
census_data$native_country[census_data$native_country %in% Canada] <-
  "Canada"

# to display the data in a tabular form
  census_data %>%
  group_by(native_country) %>%
  summarise(count =n(),
            proportion = paste0(round(count/nrow(census_data)*100, digits = 2),"%")) %>%
  arrange(desc(count)) %>%
  kable() %>%
  kable_styling(latex_options = c("striped", "hover", "condensed"))  %>%
  row_spec(0, bold = T)
```

| native_country | count | proportion |
|---|---|---|
| US | 27504 | 91.19% |
| Latin America | 1140 | 3.78% |
| Europe | 467 | 1.55% |
| South East Aisa | 346 | 1.15% |
| Asia | 288 | 0.95% |
| South America | 213 | 0.71% |
| Canada | 107 | 0.35% |
| UK | 97 | 0.32% |

### Combine Education column catagories

Similarly, categories in `education` column can be combined into a fewer columns to make the modeling more efficient.

```
# to combine the column values catagories

# No-college and Associates are the new additonal categories
census_data$education = gsub("^10th", "No-college", census_data$education)
census_data$education = gsub("^11th", "No-college", census_data$education)
census_data$education = gsub("^12th", "No-college", census_data$education)
census_data$education = gsub("^1st-4th", "No-college", census_data$education)
census_data$education = gsub("^5th-6th", "No-college", census_data$education)
census_data$education = gsub("^7th-8th", "No-college", census_data$education)
census_data$education = gsub("^9th", "No-college", census_data$education)
census_data$education = gsub("^Assoc-acdm", "Associates", census_data$education)
census_data$education = gsub("^Assoc-voc", "Associates", census_data$education)

census_data$education = gsub("^Bachelors", "Bachelors", census_data$education)
census_data$education = gsub("^Doctorate", "Doctorate", census_data$education)
census_data$education = gsub("^HS-Grad", "HS-Graduate", census_data$education)
census_data$education = gsub("^Masters", "Masters", census_data$education)
census_data$education = gsub("^Preschool", "No-college", census_data$education)
census_data$education = gsub("^Prof-school", "Prof-School", census_data$education)
census_data$education = gsub("^Some-college", "Some-college", census_data$education)

# to display the data in a tabular form
  census_data %>%
  group_by(education) %>%
  summarise(count =n(),
```

```
            proportion = paste0(round(count/nrow(census_data)*100, digits = 2),"%")) %>%
  arrange(desc(count)) %>%
  kable() %>%
  kable_styling(latex_options = c("striped", "hover", "condensed"))  %>%
  row_spec(0, bold = T)
```

| education | count | proportion |
|---|---|---|
| HS-grad | 9840 | 32.62% |
| Some-college | 6678 | 22.14% |
| Bachelors | 5044 | 16.72% |
| No-college | 3741 | 12.4% |
| Associates | 2315 | 7.68% |
| Masters | 1627 | 5.39% |
| Prof-School | 542 | 1.8% |
| Doctorate | 375 | 1.24% |

Please note: There is a weird space "ˆ" is observed in education column values, so *gsub* is used to do string replace for values.

### Combine Workclass column categories

Next, categories in `workclass` column can be combined into fewer categories as follows:

```
# to combine the column values catagories
# Not-Working and Self-Employed are two additional categories
census_data$workclass = gsub("^Without-pay", "Not-Working", census_data$workclass)
census_data$workclass = gsub("^Never-worked", "Not-Working", census_data$workclass)
census_data$workclass = gsub("^Self-emp-not-inc", "Self-Employed", census_data$workclass)
census_data$workclass = gsub("^Self-emp-inc", "Self-Employed", census_data$workclass)

# to display the data in a tabular form
  census_data %>%
  group_by(workclass) %>%
  summarise(count =n(),
            proportion = paste0(round(count/nrow(census_data)*100, digits = 2),"%")) %>%
  arrange(desc(count)) %>%
  kable() %>%
  kable_styling(latex_options = c("striped", "hover", "condensed"))  %>%
  row_spec(0, bold = T)
```

| workclass | count | proportion |
|---|---|---|
| Private | 22286 | 73.89% |
| Self-Employed | 3573 | 11.85% |
| Local-gov | 2067 | 6.85% |
| State-gov | 1279 | 4.24% |
| Federal-gov | 943 | 3.13% |
| Not-Working | 14 | 0.05% |

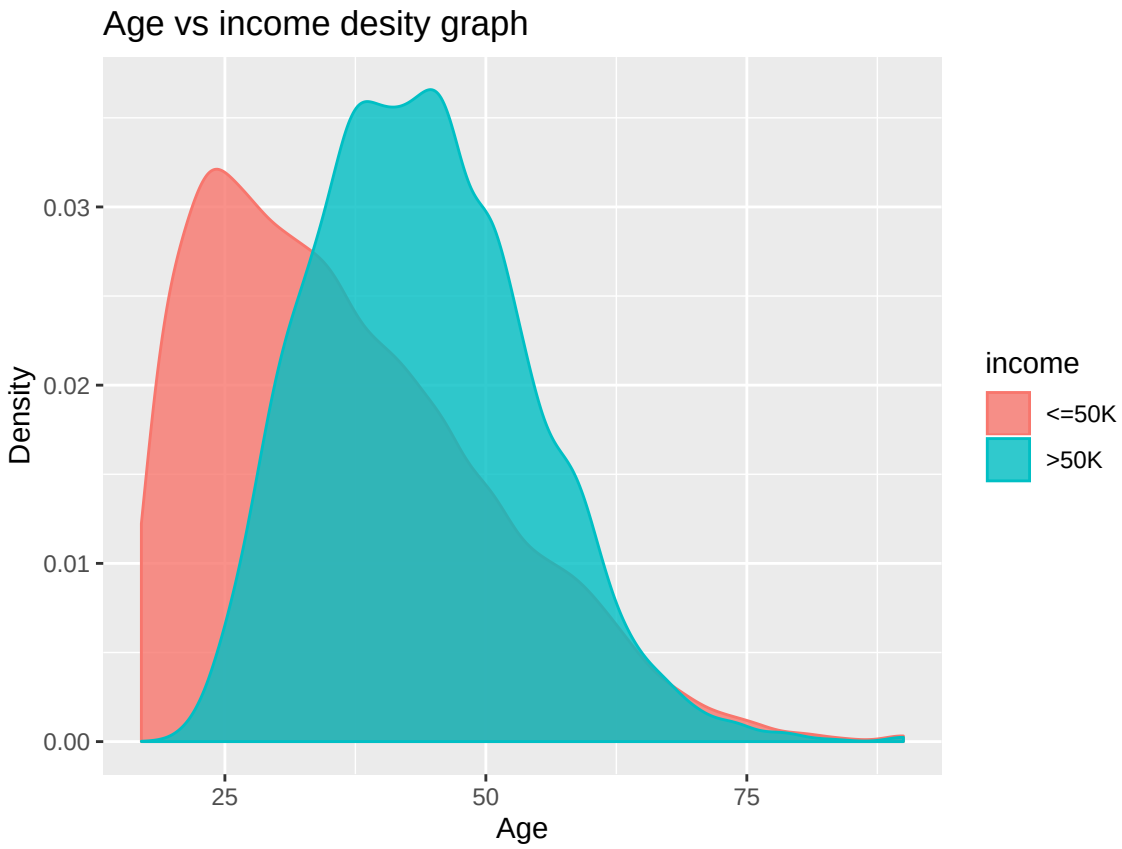
# 4. Exploratory Data Analysis and Visualization

The next step is to perform exploratory data analysis on the tidy data by utilizing visualization techniques.

### Age vs income

The chart shows that people who are older tends to earn more, specifically with the age group between 32 to 40 years. On the other hand, majority of the people with an age around 25 years earns less than 50k per annum.

```
# to plot density graph for age and income
ggplot(census_data, aes(x = age, color = income, fill = income)) +
  geom_density(alpha = 0.8) +
  # to provide a title, x, and y axis labels
  labs(x = "Age", y = "Density",
    title = "Age vs income desity graph"
  )
```




```
# to plot a table with median age for both income categories
census_data %>%
  group_by(income) %>%
  summarise(Median_age = median(age)) %>%
  kable("latex", booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hover", "condensed"))
```

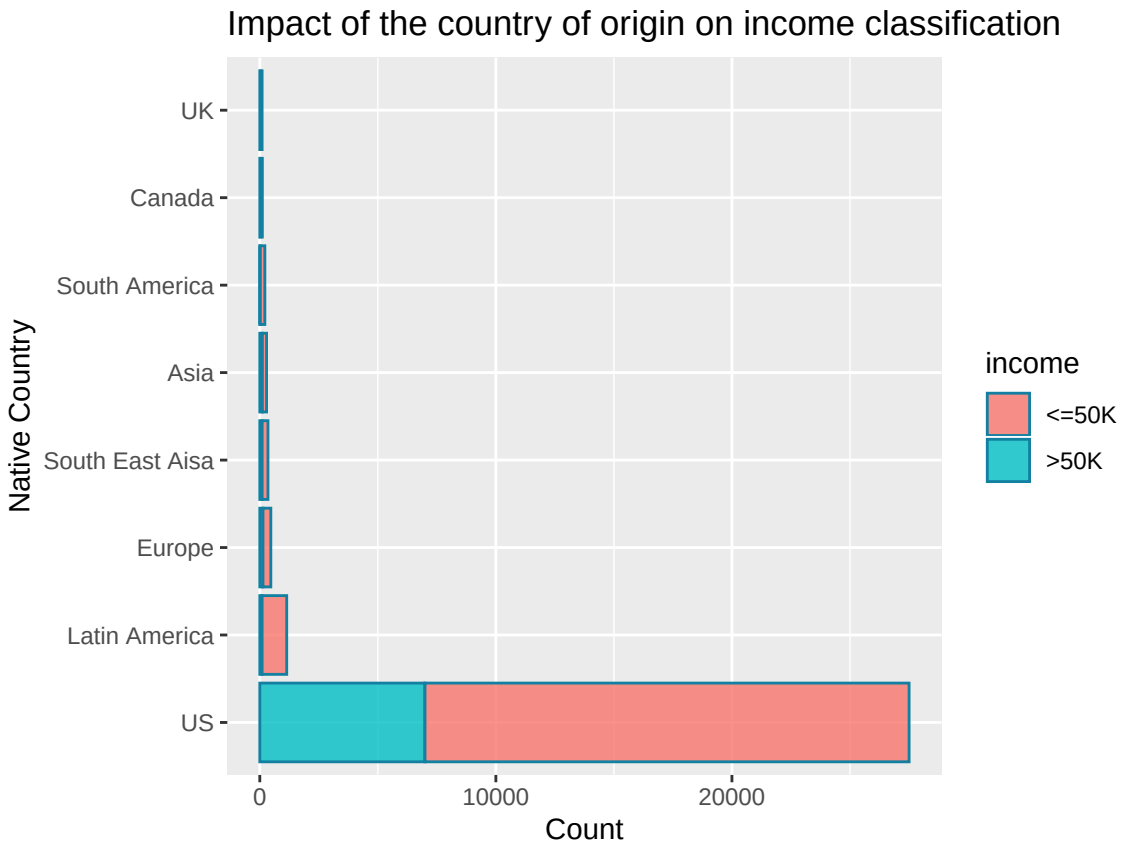
| income | Median_age |
|---|---|
| <=50K | 34 |
| >50K | 43 |

### Native Country vs income

It is evident from the graph that the country of origin has a bias in the income group. People with native country as the US has the highest proportion of both income groups. All other countries/ regions has negligible proportion, this may be due to the fact that majority of people in the population are native to the US. However, people from Asia, Canada, and the UK have higher percent of people earning USD 50k/year in their respective category.

```
# to plot native_country vs income

# first create a plot order as per native_country bar lengths
plot_order <-
  reorder(census_data$native_country, census_data$native_country, length)
plot_order <-
  factor(plot_order, levels = rev(levels(plot_order)))
# plot grap using ggplot
ggplot(census_data, aes(plot_order)) +
  geom_bar(aes(fill = income), color = "#1380A1", alpha = 0.8) +
  # to flip coordinate to adust the text
  coord_flip() +
  # to provide a title, x, and y axis labels
  labs(x = "Native Country", y = "Count",
      title = "Impact of the country of origin on income classification")
```




```
    # to display category proportion as per income distribution
    round(prop.table(table(census_data$native_country,
                                census_data$income), 1) * 100, digits = 2)
```

```
##
##                  <=50K  >50K
##   Asia           62.85 37.15
##   Canada         66.36 33.64
##   Europe         71.95 28.05
##   Latin America  91.75  8.25
##   South America  92.02  7.98
##   South East Aisa 72.25 27.75
##   UK             67.01 32.99
##   US             74.57 25.43
```

### Occupation vs income

It can also be observed from the graph that the people in managerial and professional specialty have highest proportion of earning greater than USD 50k / annum. Also, from the table below in both the categories more than 44% earn higher package. It is then followed by Sales and Craft-repair, whereas, the Armed Forces personal has the least proportion earning more than $50k / year.

```
    # to plot occupation vs income

    # first create a plot order as per occupation bar lengths
    plot_order <-
      reorder(census_data$occupation, census_data$occupation, length)
```

```
plot_order <-
  factor(plot_order, levels = rev(levels(plot_order)))
# plot grap using ggplot
ggplot(census_data, aes(plot_order)) +
  geom_bar(aes(fill = income), color = "#1380A1", alpha = 0.8) +
  # to flip coordinate to adust the text
  coord_flip() +
  # to provide a title, x, and y axis labels
  labs(x = "Occupation", y = "Count",
       title = "Biased in the income based on occupation")
```


```
# to display category proportion as per income distribution
round(prop.table(table(census_data$occupation,
                       census_data$income), 1) * 100, digits = 2)
```

```
##
##                     <=50K  >50K
##   Adm-clerical      86.62 13.38
##   Armed-Forces      88.89 11.11
##   Craft-repair      77.47 22.53
##   Exec-managerial   51.48 48.52
##   Farming-fishing   88.37 11.63
##   Handlers-cleaners 93.85  6.15
##   Machine-op-inspct 87.54 12.46
##   Other-service     95.89  4.11
##   Priv-house-serv   99.30  0.70
```

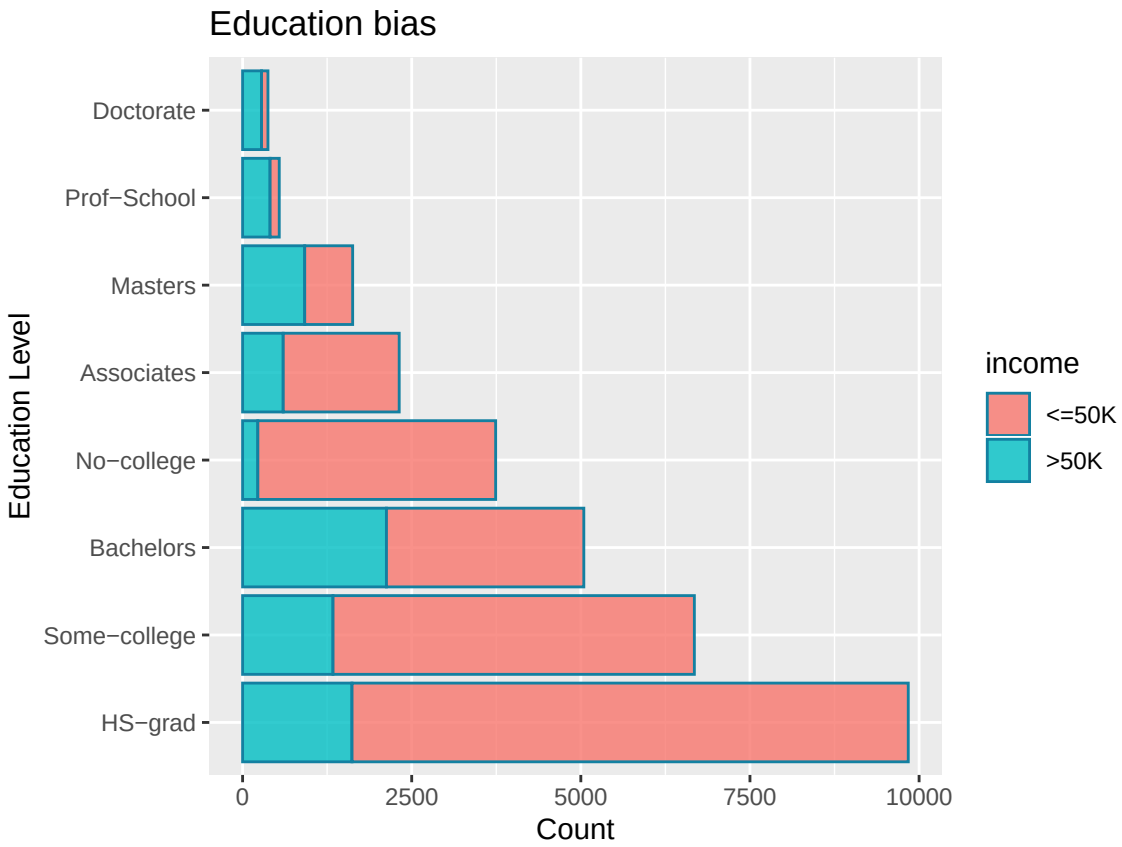
```
##   Prof-specialty    55.15 44.85
##   Protective-serv   67.39 32.61
##   Sales             72.94 27.06
##   Tech-support      69.52 30.48
##   Transport-moving  79.71 20.29
```

### Education vs income

It can be certainly observed from the below graph that more educated the person is, the chances of earning $50/annum is higher. People with Prof-School, Doctorate, and Masters education have 74.9%, 74.7%, and 56% proportion in the categories respectively. However, people with No-college level education has least proportion earning more than USD 50k/year.

```
# to plot education vs income

# first create a plot order as per education bar lengths
plot_order <-
  reorder(census_data$education, census_data$education, length)
plot_order <-
  factor(plot_order, levels = rev(levels(plot_order)))
# plot grap using ggplot
ggplot(census_data, aes(plot_order)) +
  geom_bar(aes(fill = income), color = "#1380A1", alpha = 0.8) +
  # to flip coordinate to adust the text
  coord_flip() +
  # to provide a title, x, and y axis labels
  labs(x = "Education Level", y = "Count",
       title = "Education bias")
```




```
    # to display category proportion as per income distribution
    round(prop.table(table(census_data$education,
                                census_data$income), 1) * 100, digits = 2)
```

```
##
##                <=50K  >50K
##   Associates   74.08 25.92
##   Bachelors    57.85 42.15
##   Doctorate    25.33 74.67
##   HS-grad      83.57 16.43
##   Masters      43.58 56.42
##   No-college   93.99  6.01
##   Prof-School  25.09 74.91
##   Some-college 79.99 20.01
```

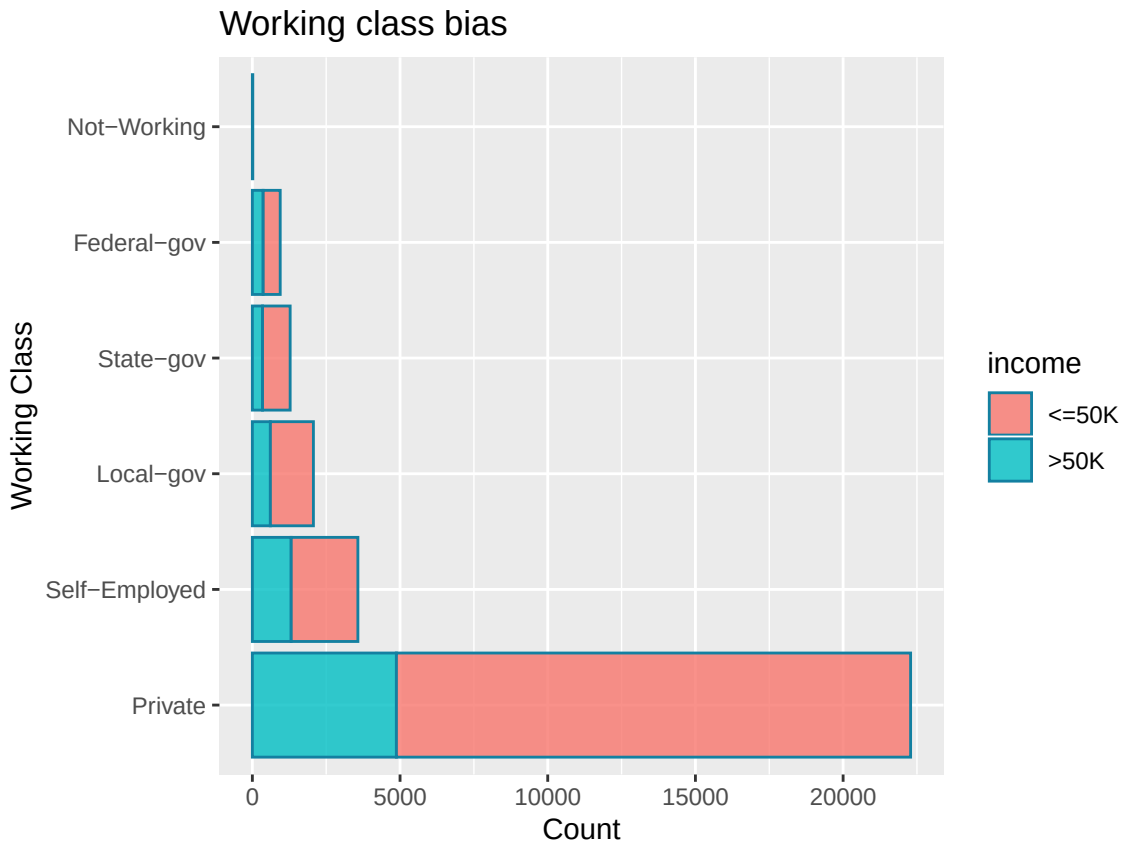
### Workclass vs income

Next, let's check what would be the impact of people's working class on their income, using the following graph. People in Fedral-gov and Self employed have higher percentage in their categories and has a bias on the income groups.

```
    # to plot workclass vs income

    # first create a plot order as per workclass bar lengths
    plot_order <-
      reorder(census_data$workclass, census_data$workclass, length)
    plot_order <-
```

```
    factor(plot_order, levels = rev(levels(plot_order)))
  # plot grap using ggplot
  ggplot(census_data, aes(plot_order)) +
    geom_bar(aes(fill = income), color = "#1380A1", alpha = 0.8) +
    # to flip coordinate to adust the text
    coord_flip() +
    # to provide a title, x, and y axis labels
    labs(x = "Working Class", y = "Count",
      title = "Working class bias")
```




```
  # to display category proportion as per income distribution
  round(prop.table(table(census_data$workclass,
                         census_data$income), 1) * 100, digits = 2)
```

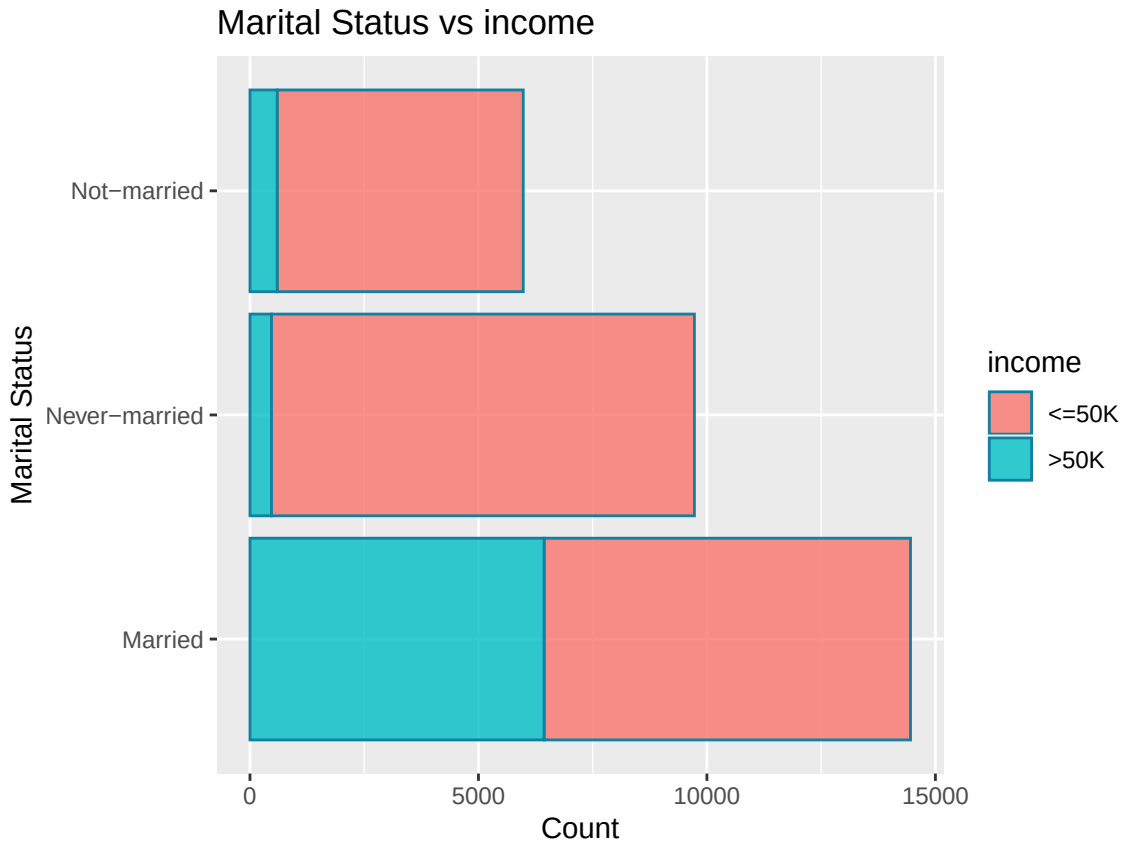
```
##
##                  <=50K   >50K
##   Federal-gov    61.29  38.71
##   Local-gov      70.54  29.46
##   Not-Working   100.00   0.00
##   Private        78.12  21.88
##   Self-Employed  63.22  36.78
##   State-gov      73.10  26.90
```

**Marital Status vs income**

As it can be observed from the below graph the the married people tend to earn more. It is followd by Not-married and Never-married categories.

```r
# to plot marital_status vs income

# first create a plot order as per marital_status bar lengths
plot_order <-
  reorder(census_data$marital_status, census_data$marital_status, length)
plot_order <-
  factor(plot_order, levels = rev(levels(plot_order)))
# plot grap using ggplot
ggplot(census_data, aes(plot_order)) +
  geom_bar(aes(fill = income), color = "#1380A1", alpha = 0.8) +
  # to flip coordinate to adust the text
  coord_flip() +
  # to provide a title, x, and y axis labels
  labs(x = "Marital Status", y = "Count",
       title = "Marital Status vs income")
```




```r
# to display category proportion as per income distribution
round(prop.table(table(census_data$marital_status,
                       census_data$income), 1) * 100, digits = 2)
```

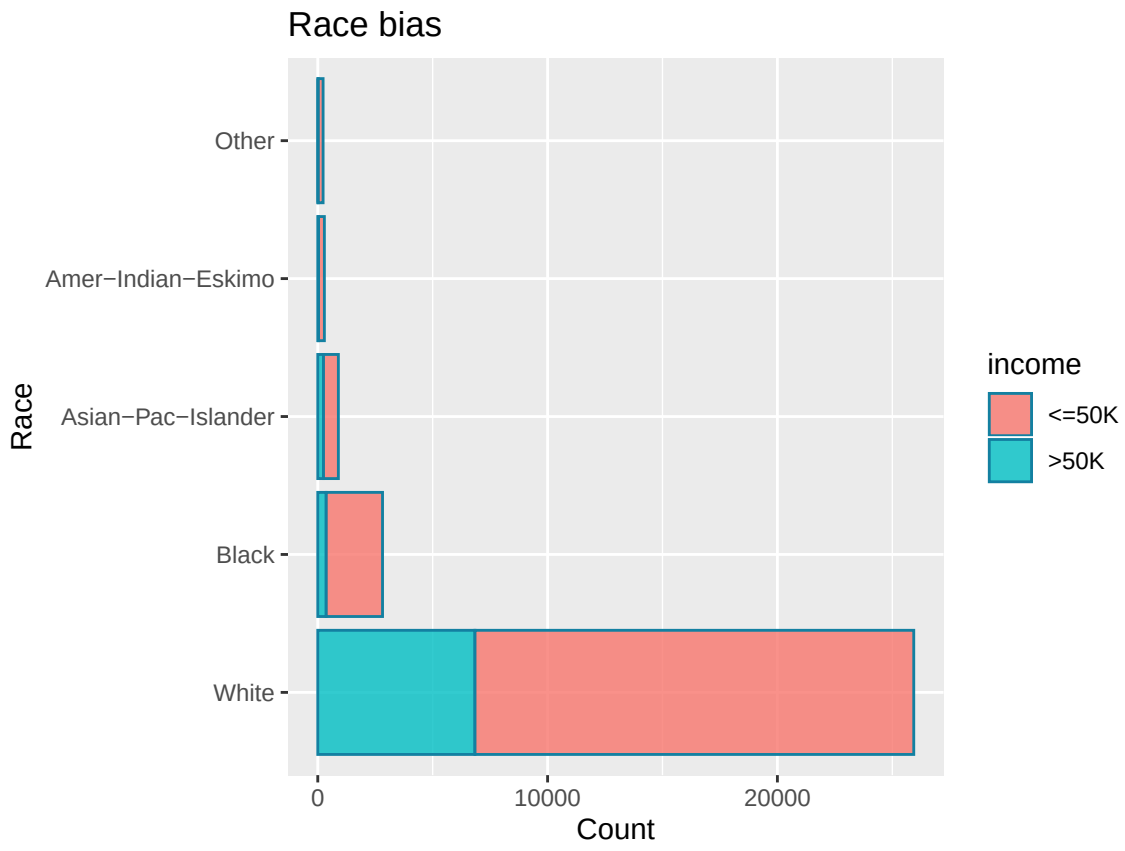
```
##
##                 <=50K  >50K
##   Married       55.45 44.55
##   Never-married 95.17  4.83
##   Not-married   90.00 10.00
```

### Race vs income

People with Asian and White as their race groups have major proportions making more that 50k in an year. However, overall proportion of people with White race have more count than combined together.

```
    # to plot race vs income

    # first create a plot order as per race bar lengths
    plot_order <-
      reorder(census_data$race, census_data$race, length)
    plot_order <-
      factor(plot_order, levels = rev(levels(plot_order)))
    # plot grap using ggplot
    ggplot(census_data, aes(plot_order)) +
      geom_bar(aes(fill = income), color = "#1380A1", alpha = 0.8) +
      # to flip coordinate to adust the text
      coord_flip() +
      # to provide a title, x, and y axis labels
     labs(x = "Race", y = "Count",
          title = "Race bias")
```




```
    # to display category proportion as per income distribution
    round(prop.table(table(census_data$race,
                           census_data$income), 1) * 100, digits = 2)
```

```
##
##                       <=50K  >50K
##   Amer-Indian-Eskimo 88.11 11.89
```

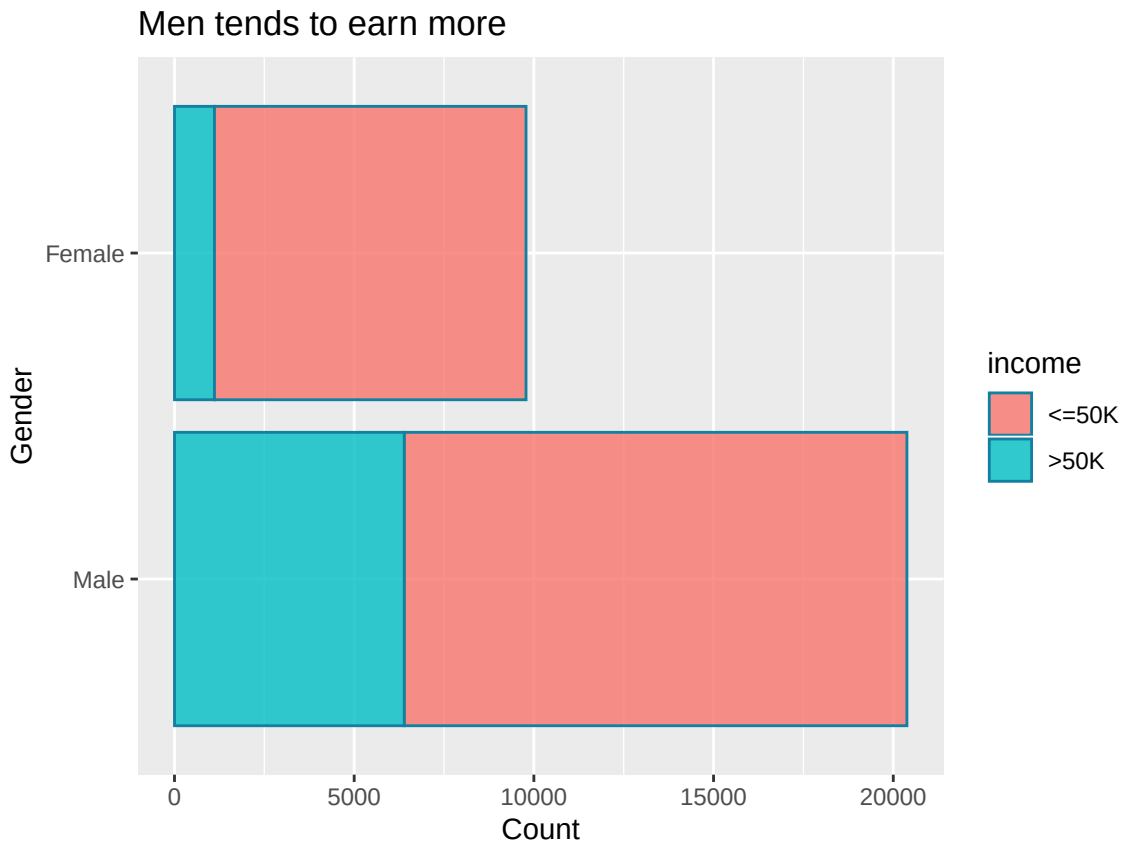
```
##   Asian-Pac-Islander 72.29 27.71
##   Black              87.01 12.99
##   Other              90.91  9.09
##   White              73.63 26.37
```

### Gender vs income

The below table depicts that the men makes more money than female, it is certainly because of higher proportion of the men population.

```
# to plot sex vs income

# first create a plot order as per sex bar lengths
plot_order <-
  reorder(census_data$sex, census_data$sex, length)
plot_order <-
  factor(plot_order, levels = rev(levels(plot_order)))
# plot grap using ggplot
ggplot(census_data, aes(plot_order)) +
  geom_bar(aes(fill = income), color = "#1380A1", alpha = 0.8) +
  # to flip coordinate to adust the text
  coord_flip() +
  # to provide a title, x, and y axis labels
  labs(x = "Gender", y = "Count", title = "Men tends to earn more")
```




```
# to display category proportion as per income distribution
round(prop.table(table(census_data$sex,
```

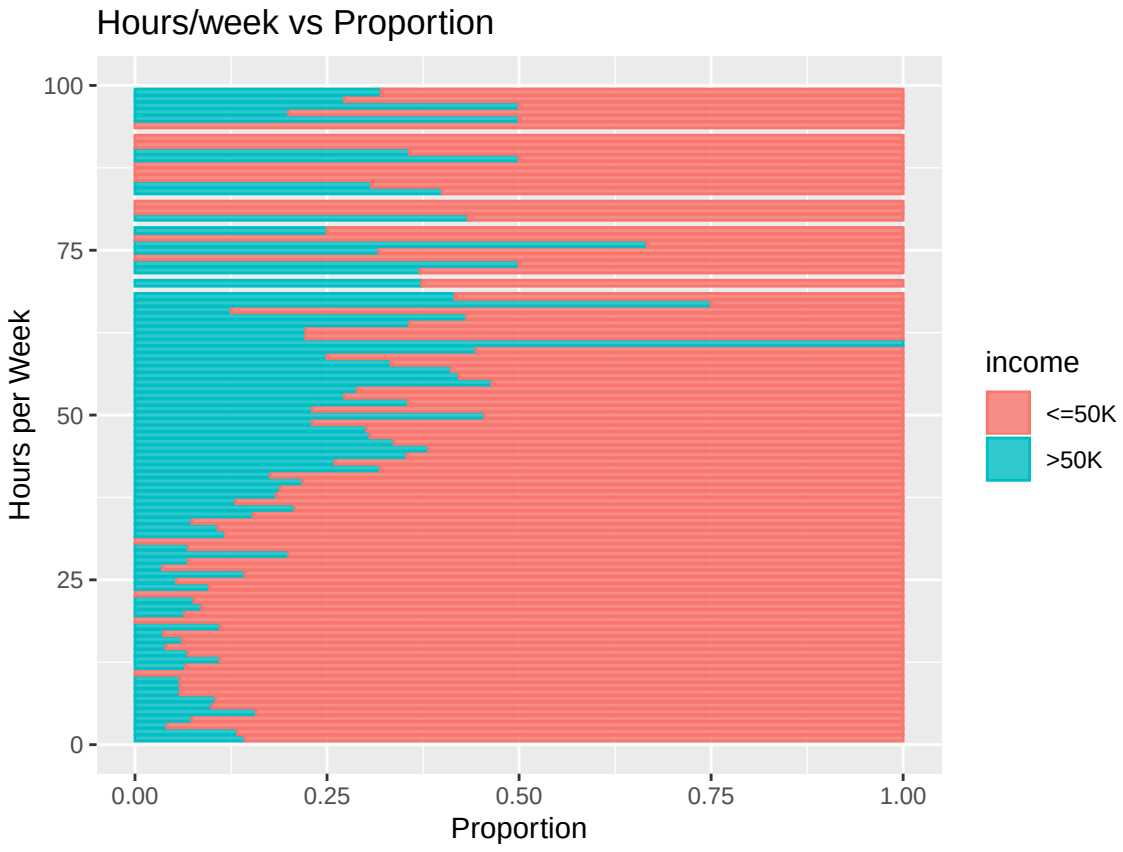
```
                                    census_data$income), 1) * 100, digits = 2)
```

```
##
##           <=50K  >50K
##    Female 88.63 11.37
##    Male   68.62 31.38
```

### Working hours vs income

The below graphs shows that the majority of people works between 35 to 65 hours; with the mean of ~ 45 hours for the people making more than 50k/year.

```
    # plot grap using ggplot
    ggplot(census_data, aes(x = hours_per_week, fill = income, color = income)) +
       geom_bar(alpha = 0.8, position = "fill") +
      # to flip coordinate to adust the text
      coord_flip() +
      # to provide a title, x, and y axis labels
      labs(x = "Hours per Week", y = "Proportion", title = "Hours/week vs Proportion")
```




```
    # to plot a table with mean working hours for both income categories
    census_data %>%
      group_by(income) %>%
      summarise(hours = mean(hours_per_week)) %>%
      kable("latex", booktabs = T) %>%
      kable_styling(latex_options = c("striped", "hover", "condensed"))
```

| income | hours |
|---|---|
| <=50K | 39.34859 |
| >50K | 45.70658 |

### Outlier Data points

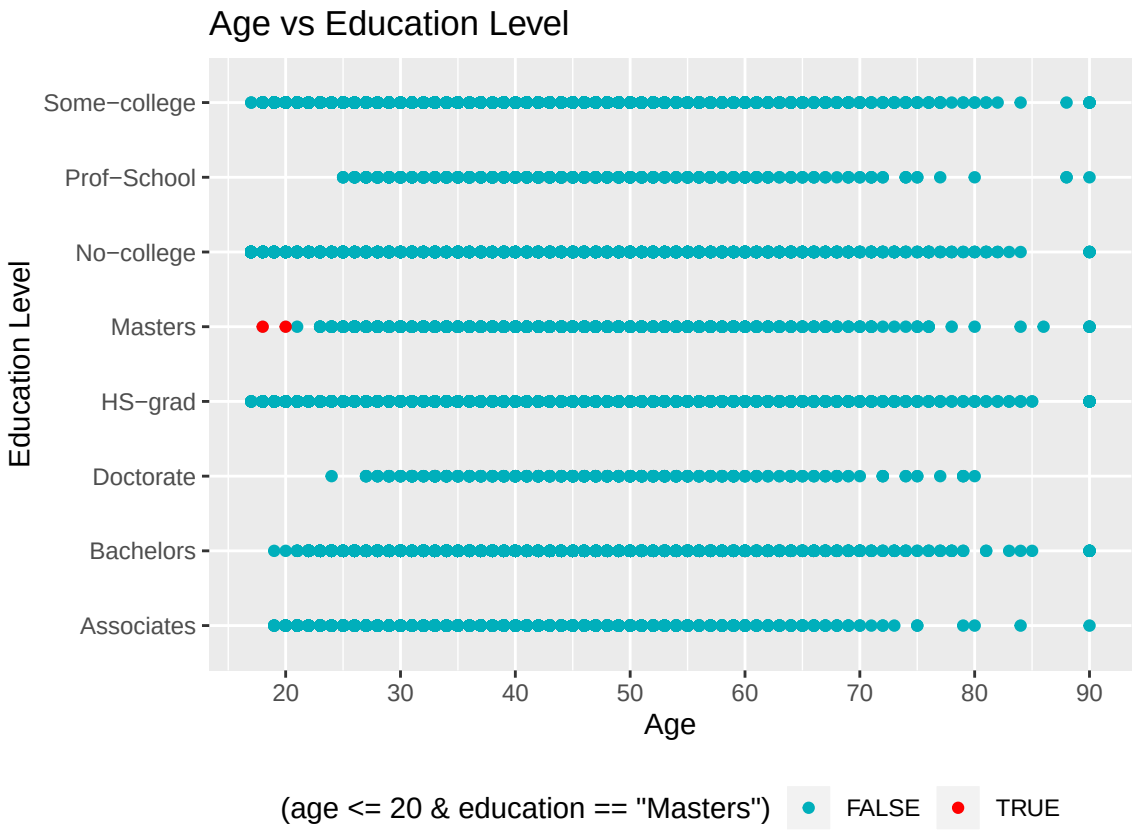
Outliers are the datapoints that differ considerably from other observations. It can be attributed to calculation uncertainty or errors. The larger outliers have an excessively large impact on model performance; thus, removing them from the dataset is advisable. Common attributes will be considered in the following sections.

### Outliers: Education level

As it can be seen from the point graph below that there are a few data points (marked with red) are outlier datapoints. It is highly unlikely that the people with age of 20 years and less can complete Masters degree.

```
census_data %>%

# to map age and education level variables to aesthetics of ggplot function
ggplot(aes(age, education)) +
# to plot scattered data using geom_point
geom_point( alpha = 0.3, col = "#00AFBB") +
# to highlight obscure outlier points with different color
geom_point(aes(col = (age<= 20 & education == 'Masters')), alpha = 1) +
scale_colour_manual(values = setNames(c('red','#00AFBB'),c(T, F)))+
# to move legend at the end of the figure
theme(legend.position="bottom") +
# to scale the axis values
scale_x_continuous(breaks = seq(0,100,10))+
# to provide a title, x, and y axis labels
xlab("Age") +
ylab("Education Level") +
ggtitle("Age vs Education Level")
```




These obscure data points with age less than or equal to 20 years and education as Masters can be removed using the code below.

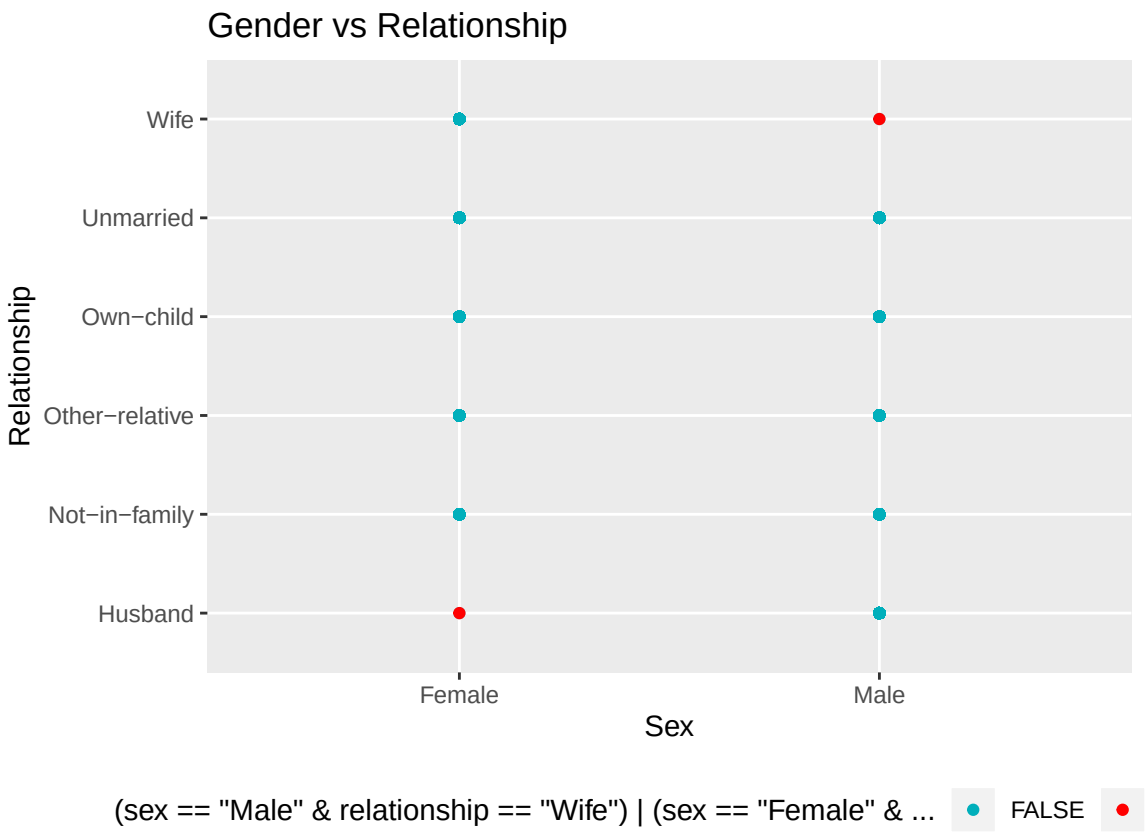
```
census_data <- census_data %>%
  filter(!(age <= 20 & education == 'Masters'))
```

### Outliers: Sex and Relationship

Simlarly, there are ponits in the graph below (marked in red) which are highly unlikely (however, could be possible). For the sake of making analysis simpler these can be removed from the dataset.

```
census_data %>%
# to map sex and relationship variables to aesthetics of ggplot function
ggplot(aes(sex, relationship)) +
# to plot scattered data using geom_point
geom_point( alpha = 0.5, col = "#00AFBB") +
# to highlight obscure outlier points with different color
geom_point(aes(col = (sex == 'Male' & relationship == 'Wife')
               | (sex == 'Female' & relationship == 'Husband')), alpha = 1)+
scale_colour_manual(values = setNames(c('red','#00AFBB'),c(T, F)))+
# to move legend at the end of the figure
theme(legend.position="bottom") +

# to provide a title, x, and y axis labels
xlab("Sex") +
ylab("Relationship") +
ggtitle("Gender vs Relationship")
```




These obscure data points with Sex as male and relationship as wife and vice-versa can be removed using the code below.

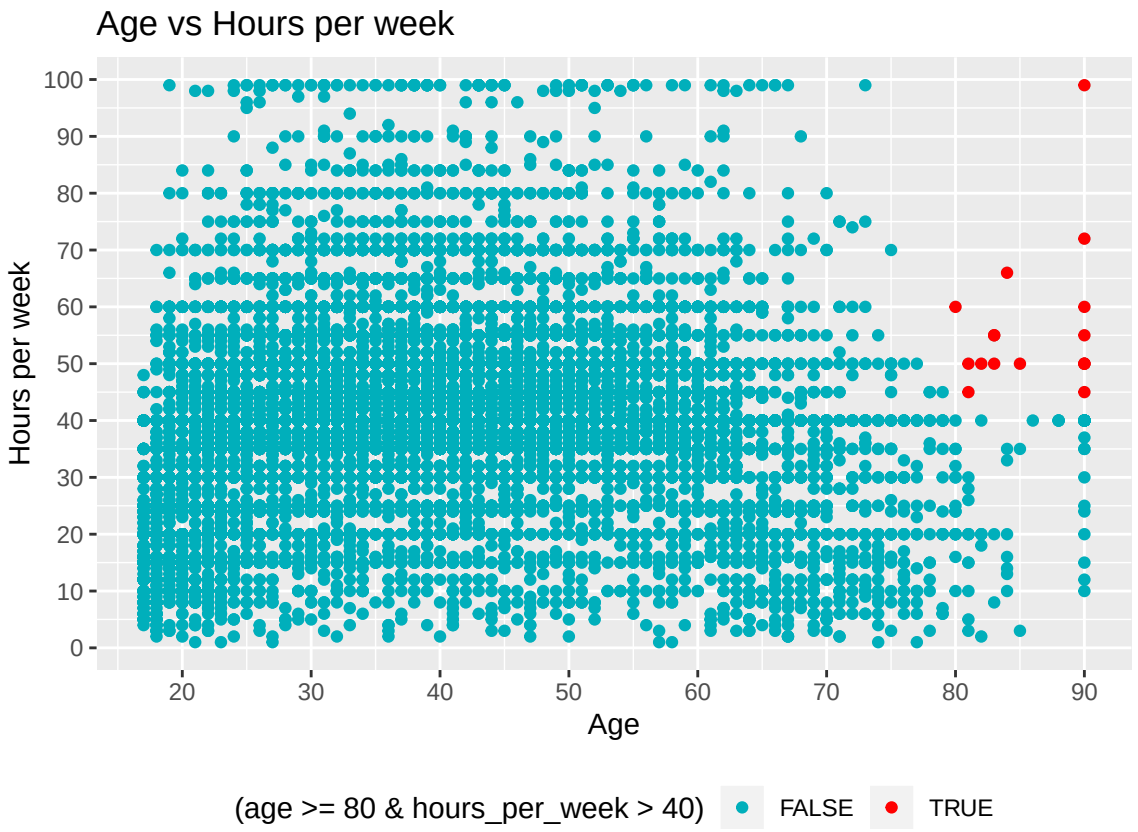
```
census_data <- census_data %>%
  filter(!((sex == 'Male' & relationship == 'Wife')
              | (sex == 'Female' & relationship == 'Husband')))
```

### Outliers: Hours per week

In fact, reports of working hours for people over 80 years of age are more than 40 hours/week, which is highly unlikely. Even for people with 90 years of age, there are records of 100 hours per week. To assist in research and modelling, these documents should be deleted.

```
census_data %>%
# to map age and hours per week variables to aesthetics of ggplot function
ggplot(aes(age, hours_per_week)) +
# to plot scattered data using geom_point
geom_point( alpha = 0.3, col = "#00AFBB") +
# to highlight obscure outlier points with different color
geom_point(aes(col = (age >= 80 & hours_per_week >40)),alpha = 1)+
scale_colour_manual(values = setNames(c('red','#00AFBB'),c(T, F)))+
# to scale the axis values
scale_x_continuous(breaks = seq(0,100,10))+
scale_y_continuous(breaks = seq(0,100,10))+
# to move legend at the end of the figure
theme(legend.position="bottom") +
```

```
  # to provide a title, x, and y axis labels
  xlab("Age") +
  ylab("Hours per week") +
  ggtitle("Age vs Hours per week")
```




These outlier data points with age more than and equal to 80 years and working hours greater than 40 hours can be removed using the code below.

```
  census_data <- census_data %>%
    filter(!(age >= 80 & hours_per_week >40))
```

### Outliers: Capital

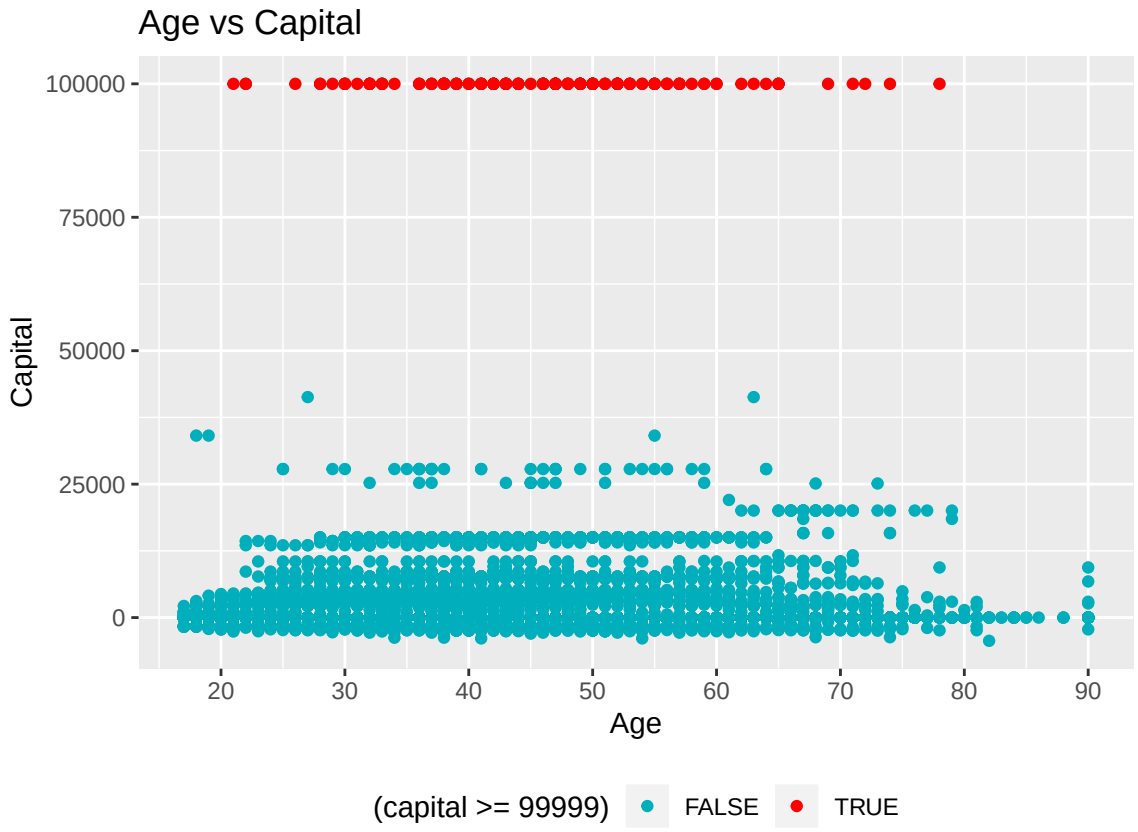
Moreover, for the Captial attribute it seems as data error, there are many points with 99999 as the Capital which seems suspicious and hence, should be eliminated from the analysis.

```
  census_data %>%
  # to map age and capital variables to aesthetics of ggplot function
  ggplot(aes(age, capital)) +

  # to plot scattered data using geom_point
  geom_point(alpha = 0.3, col = "#00AFBB") +
  # to highlight obscure outlier points with different color
  geom_point(aes(col = (capital >= 99999)), alpha = 1)+
  scale_colour_manual(values = setNames(c('red','#00AFBB'),c(T, F)))+
  # to scale the axis values
  scale_x_continuous(breaks = seq(0,100,10))+
```

```
  # to move legend at the end of the figure
  theme(legend.position="bottom") +

  # to provide a title, x, and y axis labels
  xlab("Age") +
  ylab("Capital") +
  ggtitle("Age vs Capital")
```




These erroneous data points with Capital more than and equal to 99999 can be removed using the code below.

```
census_data <- census_data %>%
  filter(!(capital >= 99999))
```

### Dataset processing and splitting

Before be move to modeling there are other activities (such as encoding etc.) needs to be done which were deferred until visualization as it makes the data hard to interpret and visualizations hard to understand.

### Factorizing

The first step is to check if any variable got changed to 'char' from 'factor' during transient analysis and wrangling steps, if any, convert them back to factors aid next steps in the project.

```
    # to check the current state of the data
    glimpse(census_data)
```

```
## Observations: 29,993
## Variables: 12
## $ age            <int> 39, 50, 38, 53, 28, 37, 49, 52, 31, 42, 37, 30,...
## $ workclass      <chr> "State-gov", "Self-Employed", "Private", "Priva...
## $ education      <chr> "Bachelors", "Bachelors", "HS-grad", "No-colleg...
## $ marital_status <chr> "Never-married", "Married", "Not-married", "Mar...
## $ occupation     <chr> "Adm-clerical", "Exec-managerial", "Handlers-cl...
## $ relationship   <fct> Not-in-family, Husband, Not-in-family, Husband,...
## $ race           <fct> White, White, White, Black, Black, White, Black...
## $ sex            <fct> Male, Male, Male, Male, Female, Female, Female,...
## $ hours_per_week <int> 40, 13, 40, 40, 40, 40, 16, 45, 50, 40, 80, 40,...
## $ native_country <chr> "US", "US", "US", "US", "Latin America", "US", ...
## $ income         <fct> <=50K, <=50K, <=50K, <=50K, <=50K, <=50K, <=50K...
## $ capital        <int> 2174, 0, 0, 0, 0, 0, 0, 0, 14084, 5178, 0, 0, 0...
```

```
    # to factoring variables to exclude the unwanted levels
    census_data$workclass <- factor(census_data$workclass)
    census_data$occupation <- factor(census_data$occupation)
    census_data$native_country <- factor(census_data$native_country)
    census_data$education <- factor(census_data$education)
    census_data$marital_status <- factor(census_data$marital_status)

    # let's check if any NA generated due to cleanup activity.
    anyNA(census_data)
```

```
## [1] FALSE
```

### Encoding: One-Hot Encode Factors

Now, let's do some data processing to make the data suitable (and often essential) for modeling classifier models.One-hot encoding is a process to convert to multiple binarized vectors from categorical vectors. Each 1s and 0s binary vector indicates the presence of the original vector class or levels. Categorical variables are translated into a type in this method that could help make algorithms more efficient; however, the number of columns is expanded.

The below code is used to do encoding and also it can be see that the columns increased to 57 from 12.

```
    # to factoring variables to exclude the unwanted levels
    census_data <- dummy.data.frame(census_data)
    # to check the impact of the encoding
    glimpse(census_data)
```

```
## Observations: 29,993
## Variables: 57
## $ age                            <int> 39, 50, 38, 53, 28, 37, 49, 52...
```

```
## $ `workclassFederal-gov`          <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `workclassLocal-gov`            <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `workclassNot-Working`          <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ workclassPrivate                <int> 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, ...
## $ `workclassSelf-Employed`        <int> 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, ...
## $ `workclassState-gov`            <int> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ educationAssociates             <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ educationBachelors              <int> 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, ...
## $ educationDoctorate              <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `educationHS-grad`              <int> 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, ...
## $ educationMasters                <int> 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, ...
## $ `educationNo-college`           <int> 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, ...
## $ `educationProf-School`          <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `educationSome-college`         <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ marital_statusMarried           <int> 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, ...
## $ `marital_statusNever-married`   <int> 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, ...
## $ `marital_statusNot-married`     <int> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationAdm-clerical`        <int> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationArmed-Forces`        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationCraft-repair`        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationExec-managerial`     <int> 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, ...
## $ `occupationFarming-fishing`     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationHandlers-cleaners`   <int> 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, ...
## $ `occupationMachine-op-inspct`   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationOther-service`       <int> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, ...
## $ `occupationPriv-house-serv`     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationProf-specialty`      <int> 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, ...
## $ `occupationProtective-serv`     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ occupationSales                 <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationTech-support`        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationTransport-moving`    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ relationshipHusband             <int> 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, ...
## $ `relationshipNot-in-family`     <int> 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, ...
## $ `relationshipOther-relative`    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `relationshipOwn-child`         <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ relationshipUnmarried           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ relationshipWife                <int> 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, ...
## $ `raceAmer-Indian-Eskimo`        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `raceAsian-Pac-Islander`        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ raceBlack                       <int> 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, ...
## $ raceOther                       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ raceWhite                       <int> 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, ...
## $ sexFemale                       <int> 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, ...
## $ sexMale                         <int> 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, ...
## $ hours_per_week                  <int> 40, 13, 40, 40, 40, 40, 16, 45...
## $ native_countryAsia              <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ native_countryCanada            <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ native_countryEurope            <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `native_countryLatin America`   <int> 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, ...
## $ `native_countrySouth America`   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `native_countrySouth East Aisa` <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ native_countryUK                <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ native_countryUS                <int> 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, ...
## $ `income<=50K`                   <int> 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, ...
```

```
## $ `income>50K`                  <int> 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, ...
## $ capital                       <int> 2174, 0, 0, 0, 0, 0, 0, 0, 140...
```

### Combine Income columns

It can be observed that the income column is also converted into two columns income>50K and income<=50K. Annual income of >50k or not can easily be represented by 1 and 0 (2 levels) respecively from the 'income>50K' column. So the column income<=50K can be removed and column income>50K can be renamed to income, that will leave the feature/column count to 56.

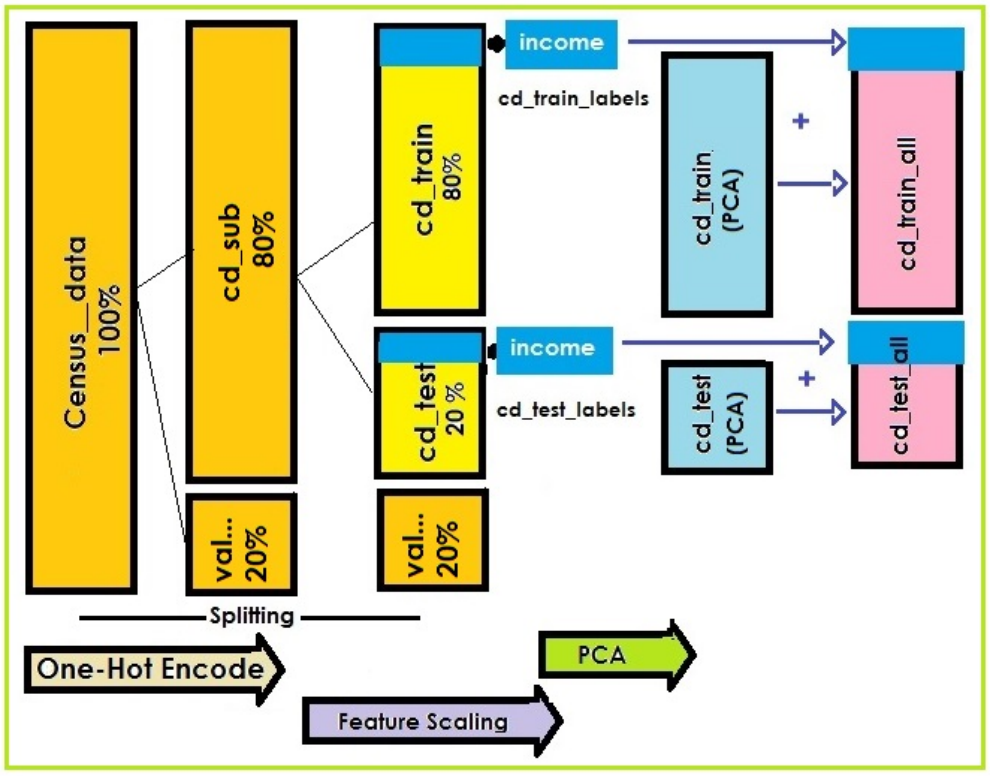
```
# to keep only a single column called income
census_data <- census_data %>%
  mutate(income = census_data$'income>50K')

# then remove census_data$'income>50K' and census_data$'income<=50K'
census_data <-
  census_data[, -which(names(census_data) %in% c('income>50K', 'income<=50K'))]
```

### Spliting dataset: Training and Validation Sets

Let's first split the census_data dataset into **cd_sub** (subset) and **cd_validation** set (20%), and the cd_sub set will then be further split into a **cd_train** and **cd_test** set with the test set being 20%. The algorithms are developed using the cd_sub (cd_train and cd_test); for a final test of the algorithm, income is to be predicted on the **cd_validation** (a separate validation dataset) as if they were unknown.

The splitting scheme is represented as follows:




The following code is used to generate the required (sub) datasets.

```
#############################################################################
# to split datasets into cd_sub, cd_validation, cd_train, and cd_test
```

```
    ##############################################################################

    # Partition the data set into cd_sub and cd_validation dataset.
    # The cd_test set will be 20% of census_data

    set.seed(1, sample.kind="Rounding")
    # if using R 3.5 or earlier, use `set.seed(1)` instead

    # to create the index with 80% train and 20% test
    indextemp <-
      sample(1:nrow(census_data), size = as.integer(nrow(census_data) * 0.8))

    cd_sub <- census_data[indextemp,]
    cd_validation  <- census_data[-indextemp,]

    #  # Now, further split the cd_sub into cd_train and cd_test dataset
    # with respect to dependent variable income
    index <-
      sample(1:nrow(cd_sub), size = as.integer(nrow(cd_sub) * 0.8))

    cd_train <- cd_sub[index,]
    cd_test  <- cd_sub[-index,]
```

### Labels Dataset

Next step is to extract labels (targets) from the test and test datasets, using the following code:

```
    #extract 'income' column of dataset to labels datasets
    cd_train_labels <- cd_train$income
    cd_test_labels <- cd_test$income

    # then remove income label from the train and test dataset
    cd_train <- cd_train %>%
      select(-income)
    cd_test  <- cd_test %>%
      select(-income)
    # to check the dimensions of the datasets
    dim(cd_train)
```

```
## [1] 19195    55
```

```
    dim(cd_test)
```

```
## [1] 4799   55
```

### Feature Scaling

Next, all the features will be scaled-up. It's a technique that scales all the characteristics that speeds up the models' training time as all features are on the same scale. The code used in the scaling of the function follows:

```
    # to scale features between 0 and 1.
    preprocessor <-
      preProcess(cd_train, method = 'range', rangeBounds = c(0, 1))
    cd_train <- predict(preprocessor, cd_train)
    cd_test <- predict(preprocessor, cd_test)
```

```
    # to check the dataset after scaling
    glimpse(cd_train)
```

```
## Observations: 19,195
## Variables: 55
## $ age                              <dbl> 0.24657534, 0.27397260, 0.1917...
## $ `workclassFederal-gov`           <dbl> 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, ...
## $ `workclassLocal-gov`             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `workclassNot-Working`           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ workclassPrivate                 <dbl> 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, ...
## $ `workclassSelf-Employed`         <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `workclassState-gov`             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ educationAssociates              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ educationBachelors               <dbl> 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, ...
## $ educationDoctorate               <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `educationHS-grad`               <dbl> 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ educationMasters                 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `educationNo-college`            <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, ...
## $ `educationProf-School`           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `educationSome-college`          <dbl> 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, ...
## $ marital_statusMarried            <dbl> 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, ...
## $ `marital_statusNever-married`    <dbl> 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, ...
## $ `marital_statusNot-married`      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ...
## $ `occupationAdm-clerical`         <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, ...
## $ `occupationArmed-Forces`         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationCraft-repair`         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationExec-managerial`      <dbl> 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, ...
## $ `occupationFarming-fishing`      <dbl> 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationHandlers-cleaners`    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationMachine-op-inspct`    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationOther-service`        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationPriv-house-serv`      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `occupationProf-specialty`       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ...
## $ `occupationProtective-serv`      <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, ...
## $ occupationSales                  <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, ...
## $ `occupationTech-support`         <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, ...
## $ `occupationTransport-moving`     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ relationshipHusband              <dbl> 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, ...
## $ `relationshipNot-in-family`      <dbl> 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, ...
## $ `relationshipOther-relative`     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `relationshipOwn-child`          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ...
## $ relationshipUnmarried            <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ relationshipWife                 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `raceAmer-Indian-Eskimo`         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `raceAsian-Pac-Islander`         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ raceBlack                        <dbl> 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, ...
## $ raceOther                        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ raceWhite                        <dbl> 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, ...
## $ sexFemale                        <dbl> 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, ...
## $ sexMale                          <dbl> 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, ...
## $ hours_per_week                   <dbl> 0.37755102, 0.14285714, 0.6020...
## $ native_countryAsia               <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ native_countryCanada             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
```

```
## $ native_countryEurope            <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `native_countryLatin America`   <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, ...
## $ `native_countrySouth America`   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `native_countrySouth East Aisa` <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ native_countryUK                <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ native_countryUS                <dbl> 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, ...
## $ capital                         <dbl> 0.09538825, 0.09538825, 0.0953...
```

### Principal Component Analysis (PCA) to reduce dimensionality

Now, let's perform Principal Component Analysis, it's a variable dimensionality reduction method; in other words, this approach combines highly correlated variables together to form a smaller number of an artificial set of variables called "principal components" that account for the most variance in the data. This improves accuracy, reduces runtime of the model and avoids problems such as over-adjustment

```
    # to perform PCA on train and test datasets with threshold of 95% variance
    preprocessor <- preProcess(cd_train, method = 'pca', threshold = 0.95)
    cd_train <- predict(preprocessor, cd_train)
    cd_test <- predict(preprocessor, cd_test)
    # to check the dataset after performing PCA
    glimpse(cd_train)
```

```
## Observations: 19,195
## Variables: 41
## $ PC1  <dbl> -2.4079909, 1.8141568, -1.0747222, -0.1253950, -2.4323001...
## $ PC2  <dbl> -0.4219133, 1.0976086, 1.6844664, -0.9264497, -0.1752024,...
## $ PC3  <dbl> -3.9962345, 0.6862909, -0.6660395, 0.8167939, 1.2123468, ...
## $ PC4  <dbl> 0.6226055, 1.1245308, -0.1304320, -3.0671445, -1.2372272,...
## $ PC5  <dbl> 0.7289721, 0.2161449, 2.2275241, 1.6903515, -0.8139000, -...
## $ PC6  <dbl> 1.28220998, -0.05055428, 2.84176504, 2.31626765, -0.44345...
## $ PC7  <dbl> 1.26093453, 0.83510060, 0.39413077, 0.45772938, -0.950697...
## $ PC8  <dbl> 2.61231928, 1.13969431, -0.46364308, -0.09377399, 0.57492...
## $ PC9  <dbl> -0.4850200, 0.9324981, -2.1584178, 0.1370711, -1.3832276,...
## $ PC10 <dbl> 1.8874943, 1.2553766, 2.4937282, -2.6429794, -0.3637159, ...
## $ PC11 <dbl> 0.79339783, 0.05117561, 0.53404762, 3.70933063, 0.9917008...
## $ PC12 <dbl> -0.2500659, -0.1056328, -0.5166864, 0.3896470, 0.4628248,...
## $ PC13 <dbl> 2.0934829, -0.5057497, -1.1898041, -0.9397197, 0.9182007,...
## $ PC14 <dbl> 1.28432648, 0.58653585, -0.80595984, 1.23916638, -1.55152...
## $ PC15 <dbl> 3.65578172, -0.35108661, -0.36991852, 0.63672556, -1.1120...
## $ PC16 <dbl> -3.69668613, 0.96784223, 1.26897740, 1.50881830, -0.36202...
## $ PC17 <dbl> -1.877562643, 0.291427245, 0.857387172, -0.599646141, 0.4...
## $ PC18 <dbl> 0.76974831, 1.24925875, 2.32532729, -1.71312223, 0.497529...
## $ PC19 <dbl> -1.20316258, 0.37172635, 0.93720519, -0.28457177, 0.17079...
## $ PC20 <dbl> -1.979265333, -0.051657286, -1.327435278, 0.880057971, -0...
## $ PC21 <dbl> -0.78810027, -0.02315620, 0.06462695, -1.57418267, -0.001...
## $ PC22 <dbl> -3.37191817, 0.23282747, -0.22053228, -0.32075630, 0.7600...
## $ PC23 <dbl> -2.71542572, 0.06503982, -0.20326570, -0.59138692, -0.153...
## $ PC24 <dbl> -0.6254273032, 0.3250873882, 0.1594727010, 0.2205887040, ...
## $ PC25 <dbl> 3.3310035952, -0.1096551489, 0.6353265512, -0.2222427729,...
## $ PC26 <dbl> -0.610573553, 0.651582345, 0.547975200, -1.030389881, -0....
## $ PC27 <dbl> -1.5646678, 0.2532156, -0.1107192, 0.8045389, -0.1313393,...
## $ PC28 <dbl> 2.821473215, -0.166345912, -0.021747956, 0.233618220, 0.0...
## $ PC29 <dbl> -0.227072272, -1.106004989, -0.645208041, 0.305365419, 0....
## $ PC30 <dbl> 1.437294095, 0.247171139, -1.200907878, -0.405599067, -0....
## $ PC31 <dbl> -0.30160891, -0.92260300, -0.42878309, 1.68143833, 0.2897...
```

```
## $ PC32 <dbl> -0.61801044, 1.26038815, 1.87940380, -0.53746955, 0.52464...
## $ PC33 <dbl> 0.03711762, -0.21113501, -0.30077225, 0.59174115, 0.63229...
## $ PC34 <dbl> 0.66210249, -1.34190114, -1.14593263, -1.47585649, 0.4810...
## $ PC35 <dbl> -0.296451914, -2.331066819, -2.760620437, 0.890179236, 0....
## $ PC36 <dbl> -0.613794817, 2.254588403, 0.863641401, 1.754962483, -0.0...
## $ PC37 <dbl> 0.69230394, 0.83745427, 0.00780415, -0.84849814, -0.35911...
## $ PC38 <dbl> 0.32470077, -0.22895466, 2.05609989, 4.53541598, 0.196656...
## $ PC39 <dbl> 0.6080260, -0.0449803, 1.2055047, 0.2643386, 0.4389305, 0...
## $ PC40 <dbl> 0.70395307, -2.66029593, -0.42632142, -0.79648140, -0.372...
## $ PC41 <dbl> 0.3222638, -0.8736402, -1.2305654, 3.1339644, -0.2997071,...
```

Now, combine labels in the train and test datasets after performing PCA, using the below code:

```
cd_train_all <-
  cbind(cd_train, cd_train_labels)
 cd_test_all <-
  cbind(cd_test, cd_test_labels)
```

# 5. Modeling Approach

Now, after the analysis, data processing, and splitting we are ready for modeling the classification model, this section describes modeling approaches and insights gained before finalizing a target model.

### 5.2 Model 1: Logistic Regression Model

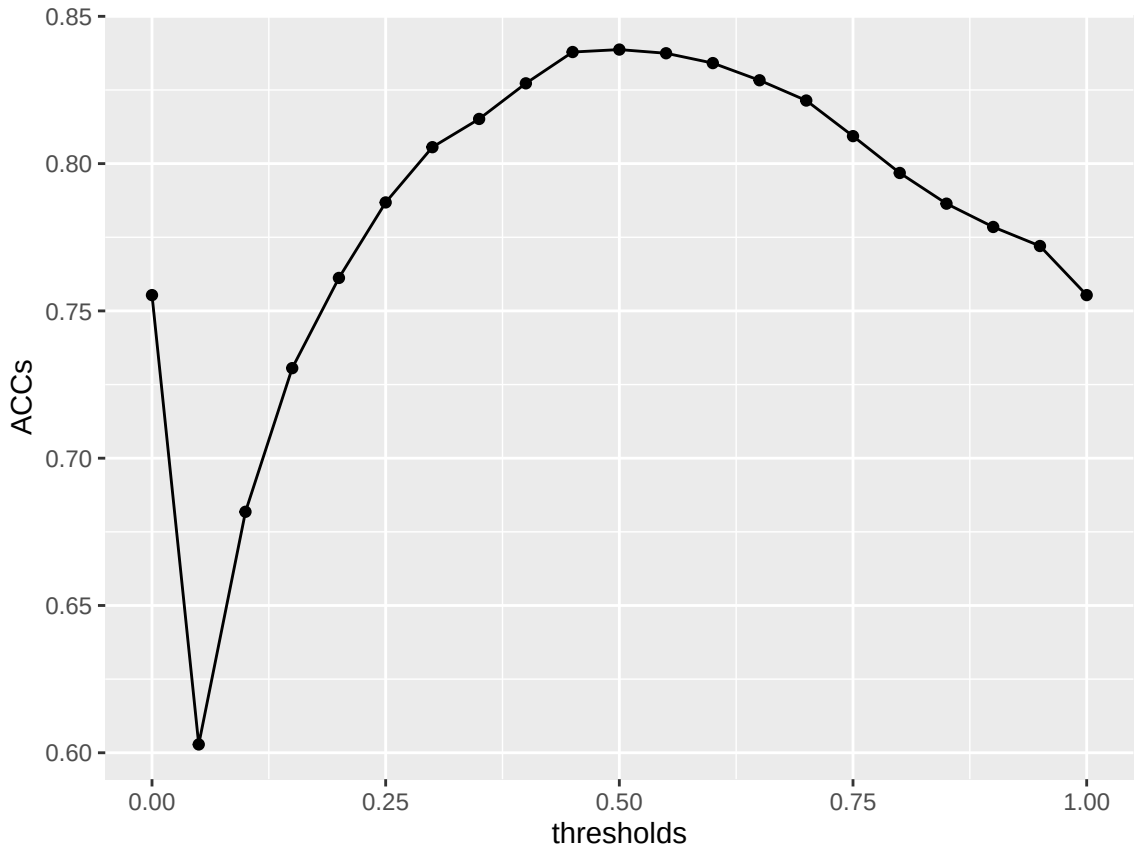
Let's first start with the *Logistic Regression Model*

```
# to create logistic regression model
log_model <- glm(cd_train_labels ~ ., data = cd_train, family = binomial(link='logit'))

#to loop through various threshold to find the optimal threshold
thresholds <- seq(0, 1, 0.05)

Prediction <- predict(log_model, newdata = cd_test, type = "response")
# loop function using sapply
ACCs <- sapply(thresholds, function(threshold) {
    pred_table <- table(cd_test_labels, Prediction >= threshold)
  # return the calculated accuracy
  return((sum(diag(pred_table)) / sum(pred_table)))
})

# to plot threshold point vs accuracy
qplot(thresholds, ACCs, geom = c("point", "line"))
```



```
    # to find the optimal threshhold for which accuracy is maximum
    threshold_log <- thresholds[which.max(ACCs)]

    threshold_log
```

```
## [1] 0.5
```

```
    # accuracy for the optimal threshhold value
    log_acc <- max(ACCs)
```

The below code will display the results in a tabular form for easy comparison.

```
      # to display Model Accuracy results in a tabular form
      model_results <- tibble(Model = "Logistic Regression Model",
                              Dataset = "cd_test",
                              Accuracy = round(log_acc * 100, digits = 2))

      model_results %>%
      # to apply theme to the table
      kable("latex", booktabs = T) %>%
      kable_styling(latex_options = c("striped", "hover", "condensed")) %>%
      row_spec(0, bold = T) %>%
      # to highlight the last row
      row_spec(1:1, bold = T, color = "white", background = "#D7261E")
```

| Model | Dataset | Accuracy |
|---|---|---|
| Logistic Regression Model | cd_test | 83.87 |

### 5.3 Model 2: Decision Tree Model

```
    # to create Decision Tree Model model
    dt_model <-
      train(cd_train_labels ~ ., data = cd_train_all, method = "rpart")

    # to predict using test dataset
    Prediction_dt <-
      predict(dt_model, newdata = cd_test)
    # to calculate model accuracy
    dt_acc <-
      mean(round(as.numeric(Prediction_dt)) == cd_test_labels)

    # append results to the table
    model_results <- bind_rows(model_results,
                               tibble(Model = "Decision Tree Model",
                                      Dataset = "cd_test",
                                      Accuracy = round(dt_acc * 100,
                                                       digits = 2)))
    model_results %>%
          kable("latex", booktabs = T)  %>%
          kable_styling(latex_options = c("striped", "hover", "condensed")) %>%
          row_spec(0, bold = T) %>%
          # to highlight the last row
          row_spec(2:2, bold = T, color = "white", background = "#D7261E")
```

| Model | Dataset | Accuracy |
|---|---|---|
| Logistic Regression Model | cd_test | 83.87 |
| **Decision Tree Model** | **cd_test** | **79.14** |

### 5.4 Model 3: Random Forest

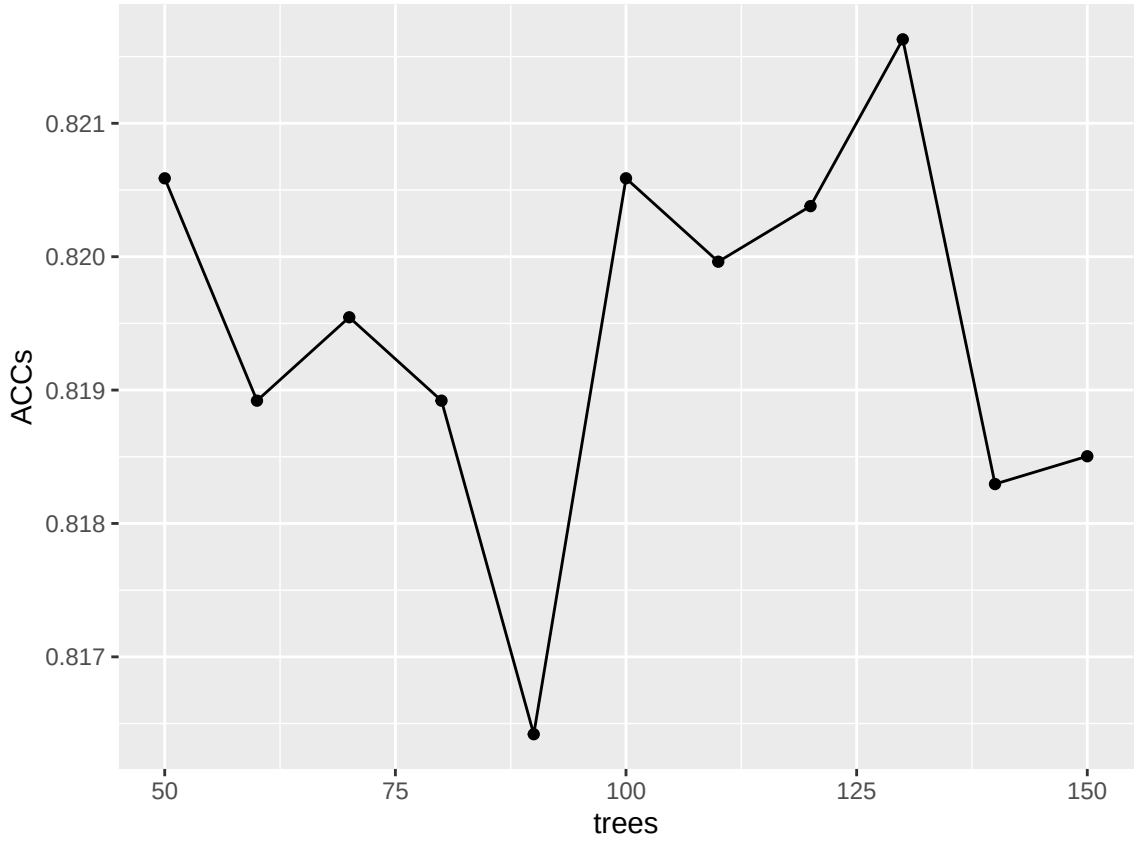
As per a publication by researchgate (details in reference section) the optimal numbers of trees for the model is between 64 and 128, so to find the optimal number of tree the algorithm is looped between 50 to 150 with an interval of 10.

```
#to loop through various threshold to find the optimal threshold (tree no.)
trees <- seq(50, 150, 10)

ACCs <- sapply(trees, function(tree) {
# to create Random Forest model
rf_model <-
  ranger(cd_train_labels ~ ., data = cd_train_all, num.trees = tree)

Prediction_rf <-
  predict(rf_model, cd_test_all)
# to calculate model accuracy
 return( mean(round(as.numeric(Prediction_rf$predictions)) == cd_test_labels))
})

# to plot threshold point vs accuracy
qplot(trees, ACCs, geom = c("point", "line"))
```




```
    # to find the optimal threshhold (no. of trees) for which accuracy is maximum
    tree_rf <- trees[which.max(ACCs)]
    tree_rf
```

```
## [1] 130
```

```
    # accuracy for the optimal threshhold value
    rf_acc <- max(ACCs)

    # append results to the table
    model_results <- bind_rows(model_results,
                               tibble(Model = "Ramdom Forest Model",
                                      Dataset = "cd_test",
                                      Accuracy = round(rf_acc * 100,
                                                       digits = 2)))
    model_results %>%
          kable("latex", booktabs = T)  %>%
          kable_styling(latex_options = c("striped", "hover", "condensed")) %>%
          row_spec(0, bold = T) %>%
          # to highlight the last row
          row_spec(3:3, bold = T, color = "white", background = "#D7261E")
```

| Model | Dataset | Accuracy |
|---|---|---|
| Logistic Regression Model | cd_test | 83.87 |
| Decision Tree Model | cd_test | 79.14 |
| **Ramdom Forest Model** | **cd_test** | **82.16** |

### 5.5 Model 4: Support Vector Machine

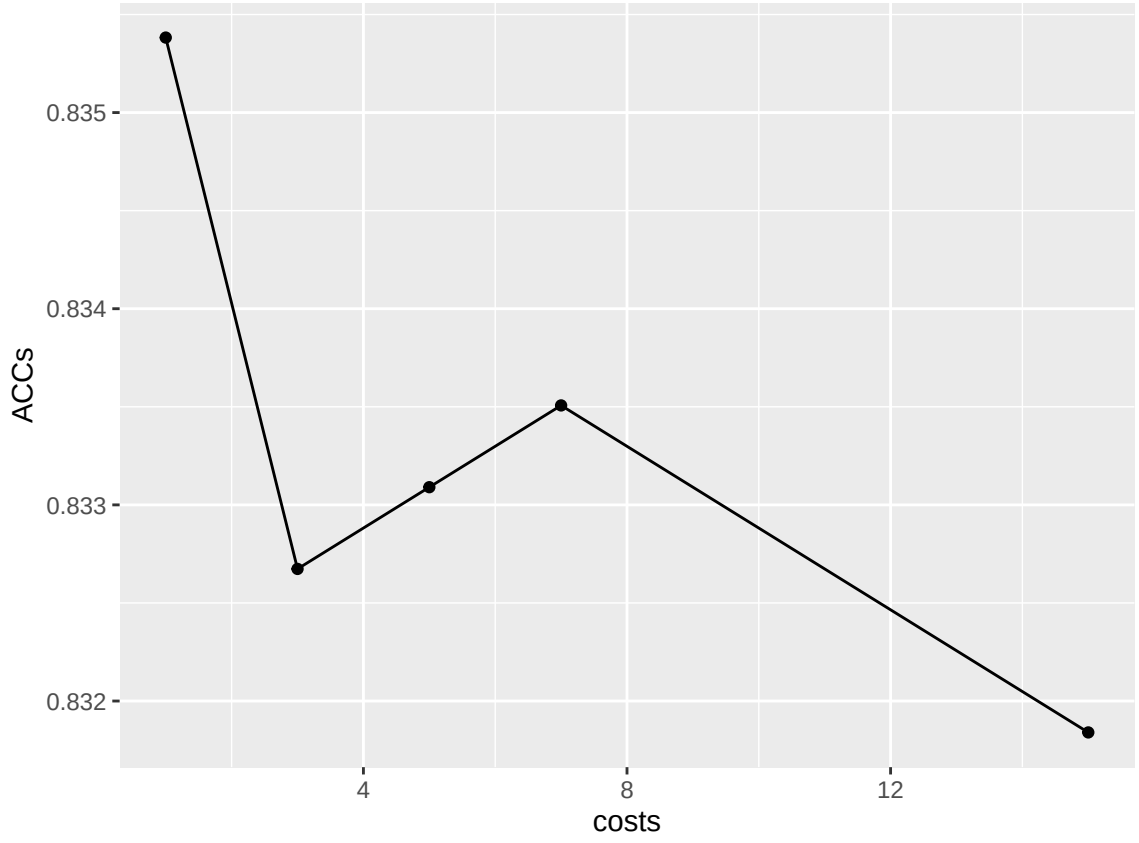
Initially, accuracy was calculated for cost ranging between 0.1 to 100, and cost near 5 gave the maximum accuracy for SVM, so a narrower cost range around 5 is tried.

```
# to create Support Vector Machine model

# loop through costs to find optimal cost for svm
costs = c(1, 3, 5, 7, 15)
ACCs <- sapply(costs, function(cost) {
  svm_model <- svm(cd_train_labels ~ ., data = cd_train, cost = cost)
  pred <- predict(svm_model, cd_test)

  # to calculate model accuracy
  return(mean(round(as.numeric(pred)) == cd_test_labels))
})

# to plot threshold point vs accuracy
qplot(costs, ACCs, geom = c("point", "line"))
```




```
# to find the optimal cost for which accuracy is maximum
cost_svm <- costs[which.max(ACCs)]
cost_svm
```

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

```
# accuracy for the optimal cost value
svm_acc <- max(ACCs)
```

```r
# append results to the table
model_results <- bind_rows(model_results,
                           tibble(Model = "Support Vector Machine",
                                  Dataset = "cd_test",
                                  Accuracy = round(svm_acc * 100,
                                                   digits = 2)))
model_results %>%
      kable("latex", booktabs = T)  %>%
      kable_styling(latex_options = c("striped", "hover", "condensed")) %>%
      row_spec(0, bold = T) %>%
      # to highlight the last row
      row_spec(4:4, bold = T, color = "white", background = "#D7261E")
```

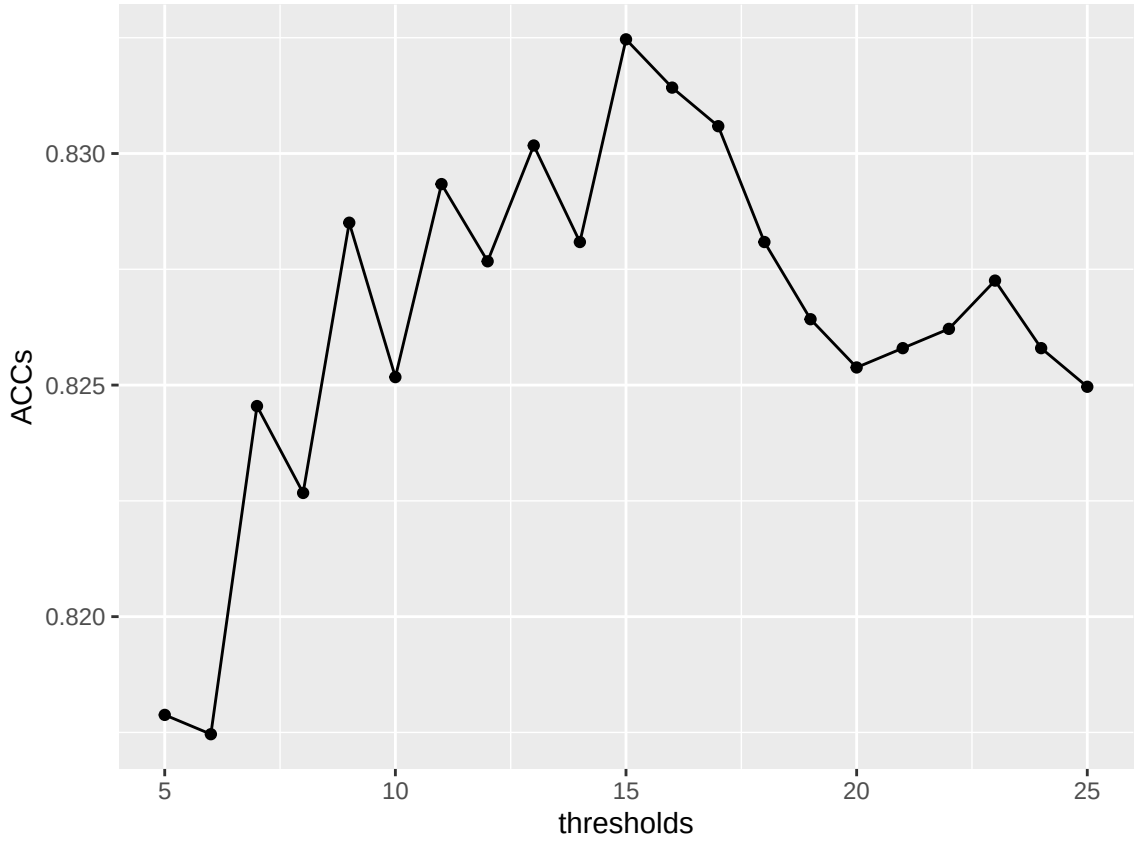
| Model | Dataset | Accuracy |
|---|---|---|
| Logistic Regression Model | cd_test | 83.87 |
| Decision Tree Model | cd_test | 79.14 |
| Ramdom Forest Model | cd_test | 82.16 |
| **Support Vector Machine** | **cd_test** | **83.54** |

### 5.6 Model 5: k Nearest Neighbors

```r
# to create KNN model
#to loop through various threshold to find the optimal threshold
thresholds <- seq(5, 25, 1)

ACCs <- sapply(thresholds, function(threshold) {
  knn_model <- knn(cd_train, cd_test,
                   cl = cd_train_labels, k = threshold)
  # return the calculated accuracy
  return(mean(knn_model == cd_test_labels))
})

# to plot threshold point vs accuracy
qplot(thresholds, ACCs, geom = c("point", "line"))
```




```
    # to find the optimal threshhold for which accuracy is maximum
    threshold <- thresholds[which.max(ACCs)]

    threshold
```

```
## [1] 15
```

```
    # accuracy for the optimal threshhold value
    knn_acc <- max(ACCs)

    # append results to the table
    model_results <- bind_rows(model_results,
                               tibble(Model = "k Nearest Neighbor",
                                      Dataset = "cd_test",
                                      Accuracy = round(knn_acc * 100,
                                                       digits = 2)))
    model_results %>%
          kable("latex", booktabs = T)  %>%
          kable_styling(latex_options = c("striped", "hover", "condensed")) %>%
          row_spec(0, bold = T) %>%
          # to highlight the last row
          row_spec(5:5, bold = T, color = "white", background = "#D7261E")
```

| Model | Dataset | Accuracy |
|---|---|---|
| Logistic Regression Model | cd_test | 83.87 |
| Decision Tree Model | cd_test | 79.14 |
| Ramdom Forest Model | cd_test | 82.16 |
| Support Vector Machine | cd_test | 83.54 |
| **k Nearest Neighbor** | **cd_test** | **83.25** |

### 5.7 Evaluating the selected model on validation dataset

The Logistic Regression Model generated the highest accuracy of **83.87%** with an improvement over other models. Next, let's check the effectiveness of the model on the validation.

```
    ##############################################################################
      # Predict using Logistic Regression Model on validation set
    ##############################################################################

    #extract 'income' column of dataset to labels datasets
    cd_train_vlabels <- cd_sub$income
    cd_test_vlabels <- cd_validation$income

    # then remove income label from the cd_sub and validation dataset
    cd_sub <- cd_sub %>%
      select(-income)
    cd_validation  <- cd_validation  %>%
      select(-income)

    # to scale features between 0 and 1.
    preprocessor <-
      preProcess(cd_sub, method = 'range', rangeBounds = c(0, 1))
    cd_sub <- predict(preprocessor, cd_sub)
    cd_validation <- predict(preprocessor, cd_validation)

    # to perform PCA on cd_sub and validation datasets with threshold of 95% variance
    preprocessor <- preProcess(cd_sub, method = 'pca', threshold = 0.95)
    cd_sub <- predict(preprocessor, cd_sub)
    cd_validation <- predict(preprocessor, cd_validation)

    # optimal threshold selected in Log Regression Model
    threshold_log
```

```
## [1] 0.5
```

```
    # to create logistic regression model using validation set
    log_model_val <- glm(cd_train_vlabels ~ ., data = cd_sub, family = binomial(link='logit'))

    Prediction <- predict(log_model_val, newdata = cd_validation, type = "response")
        pred_table <- table(cd_test_vlabels, Prediction >= threshold_log)

    # accuracy for the optimal threshhold value
    log_val_acc <- (sum(diag(pred_table)) / sum(pred_table))

    # append results to the table
    model_results <- bind_rows(model_results,
                               tibble(Model = "Logistic Regression Model",
```

```
                                    Dataset = "cd_validation",
                                    Accuracy = round(log_val_acc * 100,
                                                     digits = 2)))
model_results %>%
  # to apply theme to the table
  kable("latex", booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hover", "condensed")) %>%
  row_spec(0, bold = T) %>%
  # to highlight the last row
  row_spec(6:6,bold = T, color = "white", background = "#D7261E")
```

| Model | Dataset | Accuracy |
|---|---|---|
| Logistic Regression Model | cd_test | 83.87 |
| Decision Tree Model | cd_test | 79.14 |
| Ramdom Forest Model | cd_test | 82.16 |
| Support Vector Machine | cd_test | 83.54 |
| k Nearest Neighbor | cd_test | 83.25 |
| **Logistic Regression Model** | **cd_validation** | **84.15** |

This model generated the accuracy of **84.15%** on validation set. Next, we list and compare all the results in the next section.

# 6. Results

### 6.1 Modeling result comparison and performance

The results of the accuracy of various models are listed below. The **Logistic Regression Model** achieved the highest accuracy and the ultimately proposed model (highlighted in green) of all models explored in this project. It is closely followed by Support Vector Machine, k Nearest Neighbor; whereas, Random Forest Model and Decision Tree Model ranked at the least two positions with respect to the performance.

The poor performance by Random Forest as compared with other models is attributed to not excluding the less important features (feature importance). kNN model is known to work better for classification problems; however, for the same feature importance reason it doesn't yield better performance. Support Vector Machine produced the similar result as compared with kNN, the SVM performance could be improved by introducing gamma. Logistic Regression Model is a simpler model with less time to train.

```
  # to display result tables with prediction model results
model_results %>%
      kable("latex", booktabs = T)  %>%
      kable_styling(latex_options = c("striped", "hover", "condensed")) %>%
      row_spec(0, bold = T) %>%
      # to highlight the last row
      row_spec(6:6, bold = T, color = "white", background = "#3DDB48")
```

| Model | Dataset | Accuracy |
|---|---|---|
| Logistic Regression Model | cd_test | 83.87 |
| Decision Tree Model | cd_test | 79.14 |
| Ramdom Forest Model | cd_test | 82.16 |
| Support Vector Machine | cd_test | 83.54 |
| k Nearest Neighbor | cd_test | 83.25 |
| **Logistic Regression Model** | **cd_validation** | **84.15** |

# 7. Conclusion

### 7.1 Brief summary

The main aim of the project is to develop a classifier system to predict if a person earns an annual income of greater than 50k. The used Adult Census Income dataset is a real-world challenge that seasoned data scientists would have handled. From data setup, data wrangling as well as exploratory analysis and ggplot based visualization graphs to various modeling approaches, the concepts learned throughout the series of these courses are implemented.

Initially, Decision Tree, and Support Vector Machine models are developed, later more complex classification models are attempted. An accuracy of **84.15%** is attained with the **Logistic Regression Model**.

From the analysis, it can be summarized that 'age', 'education', 'hours_per_week', 'sex', and 'occupation' are the most important features.

### 7.2 Future work

In this project, the popluar analysis and modeling approaches are used to keep it limited to the scope of Capstone project. Nonetheless, more advanced modeling methods such as NNET, Deep Neural Network, etc. as well as modeling with features based on their importance could be used as a potential extension of this project to further improve the performance. In addition, other significant comparison quality features including Sensitivity, Specificity, Prevalence, F1 Score, AUC, ROC curve can be explored further.

### 7.3 Limitations

Initially, Deep Neural Network was also planned; however, due to computing resource constraint it was not successful. Also, due to the constraint, more sophisticated models with higher performance didn't succeed and thus not included in this report. Those would require powerful computing machines.

# 8. References


- Data Science textbook by Rafael Irizarry
  - https://rafalab.github.io/dsbook/large-datasets.html#dimension-reduction
  - https://rafalab.github.io/dsbook/machine-learning-in-practice.html#k-nearest-neighbor-and-random-forest
- UCI
  - Adult Census Income Dataset: http://archive.ics.uci.edu/ml/machine-learning-databases/adult
- adult: "Census Income" Dataset: https://rdrr.io/cran/datadr/man/adult.html
- Sisay Menji Bekena:"Using decision tree classifier to predict income levels", Munich Personal RePEc Archive 30th July, 2017
- http://cseweb.ucsd.edu/classes/sp15/cse190-c/reports/sp15/048.pdf
- https://www.researchgate.net/publication/230766603_How_Many_Trees_in_a_Random_Forest

- https://scholar.smu.edu/cgi/viewcontent.cgi?article=1041&context=datasciencereview

# 9. Github Repo

- https://github.com/jha-r/Harvardx-PH125.9x-Capstone-Census_Income_CYO