HarvardX PH125.9x - Capstone Project - CYO
Classification of Income Using Machine Learning (with KNN and RandomForest)

Kwaku Owusu-Tieku (kowusu01@gmail.com)

2/20/2022

Introduction

Income disparity is common in every society and it's probably been around since man has been able to measure it. There are various reasons why income disparities exist. These include age, gender, race, level of education, etc.

For instance, with respect to age, younger people who have just entered the workforce may earn less than their counterparts who have been in the industry for many years. The level of education may also explain income disparities since some high paying jobs often require skilled and highly educated individuals.

Even different geographic regions or cities may offer different levels of income since some high paying jobs are sometimes more concentrated in some regions or cities than others. However, it is still difficult to measure how much these factors determine one's income.

In this project, I will I apply kNN and randomForest to a dataset containing various attributes (predictors) and individual's income level (above or below $50K), and predict the individual's income level.

The kNN and randomForest algorithms were chosen because they are very easy to train, and they are able handle a wide range data including a mix of categorical and numeric variables and offer good performance.

For details on reproducing the analysis presented in this document, refer to the Technical Details section.

The original data

Source
The data used in this project is part of the UCI machine learning datasets provided to the public. It can be found at: https://archive.ics.uci.edu/ml/machine-learning-databases/adult/.

Three main files are provided as part of the dataset.

- adult.names
- adult.data
- adult.test

The *adult.names* file, provides very useful information about the data. The *adult.data* and *adult.test* files are the training and test sets. The data had already been split into training and test sets by the contributer(s). The training data contains *32561* records. The test set contains *16281* records, about 50% of the training set. This provides a good amount of data for training while leaving enough for validation. Each record in the datasets represents an observation.

The documentation indicates that the original data was retrieved from the 1994 census bureau database (http://www.census.gov/ftp/pub/DES/www/welcome.html), so the samples were taken from real dataset. The U.S. population and workfoce have changed dramatically since 1994, so the patterns in this data may be different from when the samples were originally taken.

Reading the data
A quick peek at the data reveals that there are no column names. However, descriptions in the *adult.names* file provides useful information in naming the columns.
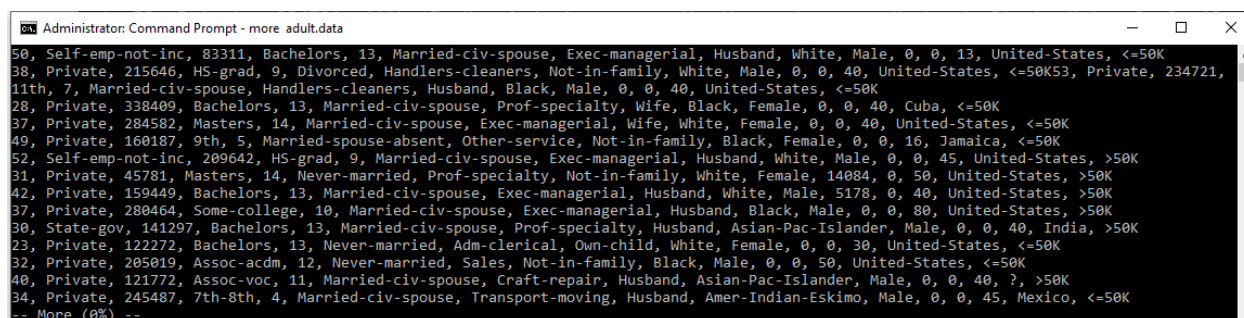

Figure 1: original data - raw

Code to read the data

```r
# use wrapr::qc() function to create a list of quoted string for the column headers
COL_HEADERS <- qc(age, workclass, final_weight, education, education_num,
                  marital_status, occupation, relationship, race, sex,
                  capital_gain, capital_loss, hours_per_week, native_country, Income)

# read training and test datasets into memory
adult_income_train <- read.csv(TRAINING_DATA_PATH, col.names = COL_HEADERS)
adult_income_validation  <- read.csv(VALIDATION_DATA_PATH, col.names = COL_HEADERS)

# trim all character columns
adult_income_train <- adult_income_train %>% mutate(across (where(is.character), str_trim))
adult_income_validation  <- adult_income_validation %>%
  mutate(across (where(is.character), str_trim))
```

Structure of the dataframe

The data contains 15 columns; the *Income* column is the response variable with the other fourteen columns being the independent variables. The structure of the dataframes looks as below:

```
glimpse(adult_income_train)
```

```
Rows: 32,560
Columns: 15
$ age            <int> 50, 38, 53, 28, 37, 49, 52, 31, 42, 37, 30, 23, 32, 40,~
$ workclass      <chr> "Self-emp-not-inc", "Private", "Private", "Private", "P~
$ final_weight   <int> 83311, 215646, 234721, 338409, 284582, 160187, 209642, ~
$ education      <chr> "Bachelors", "HS-grad", "11th", "Bachelors", "Masters",~
$ education_num  <int> 13, 9, 7, 13, 14, 5, 9, 14, 13, 10, 13, 13, 12, 11, 4, ~
$ marital_status <chr> "Married-civ-spouse", "Divorced", "Married-civ-spouse",~
$ occupation     <chr> "Exec-managerial", "Handlers-cleaners", "Handlers-clean~
$ relationship   <chr> "Husband", "Not-in-family", "Husband", "Wife", "Wife", ~
$ race           <chr> "White", "White", "Black", "Black", "White", "Black", "~
$ sex            <chr> "Male", "Male", "Male", "Female", "Female", "Female", "~
$ capital_gain   <int> 0, 0, 0, 0, 0, 0, 0, 14084, 5178, 0, 0, 0, 0, 0, 0, 0, ~
$ capital_loss   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
$ hours_per_week <int> 13, 40, 40, 40, 40, 16, 45, 50, 40, 80, 40, 30, 50, 40,~
$ native_country <chr> "United-States", "United-States", "United-States", "Cub~
$ Income         <chr> "<=50K", "<=50K", "<=50K", "<=50K", "<=50K", "<=50K", "~
```

```
glimpse(adult_income_validation)
```

```
Rows: 16,280
Columns: 15
$ age            <int> 38, 28, 44, 18, 34, 29, 63, 24, 55, 65, 36, 26, 58, 48,~
$ workclass      <chr> "Private", "Local-gov", "Private", "?", "Private", "?",~
$ final_weight   <int> 89814, 336951, 160323, 103497, 198693, 227026, 104626, ~
$ education      <chr> "HS-grad", "Assoc-acdm", "Some-college", "Some-college"~
$ education_num  <int> 9, 12, 10, 10, 6, 9, 15, 10, 4, 9, 13, 9, 9, 9, 14, 10,~
$ marital_status <chr> "Married-civ-spouse", "Married-civ-spouse", "Married-ci~
$ occupation     <chr> "Farming-fishing", "Protective-serv", "Machine-op-inspc~
$ relationship   <chr> "Husband", "Husband", "Husband", "Own-child", "Not-in-f~
$ race           <chr> "White", "White", "Black", "White", "White", "Black", "~
$ sex            <chr> "Male", "Male", "Male", "Female", "Male", "Male", "Male~
$ capital_gain   <int> 0, 0, 7688, 0, 0, 0, 3103, 0, 0, 6418, 0, 0, 0, 3103, 0~
$ capital_loss   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
$ hours_per_week <int> 50, 40, 40, 30, 30, 40, 32, 40, 10, 40, 40, 39, 35, 48,~
$ native_country <chr> "United-States", "United-States", "United-States", "Uni~
$ Income         <chr> "<=50K.", ">50K.", ">50K.", "<=50K.", "<=50K.", "<=50K.~
```

Data quality checks

Although the data for this analysis has previously been cleaned by the team that made it available to UCI, a few things had to be checked to make sure the data is ready to be transformed and used in an analysis.

1. Verify correct number of records read. Based on the documentation in the *adult.names*, there are 32561 records in the training set.

2. Check for invalid/missing data (NAs, invalid zeros for age, etc). Verify the presence of NA in all columns. No NAs (null values) are found in both train and test sets. However, some of the categorical columns have questions mark (?) as values. Again, according to the documentation, these were missing values; the NAs (null values) had been replaced with questions marks.

3. Numeric variables. Check the range of values to make sure they do not include invalid values, e.g. age should be greater than zero, and not excessively high number.

4. Categorical values. Check the number of levels.

The following code was used for the basic sanity checks described above.

Basic data quality checks - data read

```
########################################################################
## DATA QUALITY CHECKS
########################################################################

# 1. check that correct number of rows was read
TRAIN_NUM_OF_ROWS_EXPECTED <- 32561
VALIDATION_NUM_OF_ROWS_EXPECTED  <- 16281

verify_stats <- data.frame("Item" = c("Number of row (training set)",
                                       "Number of row (test set)"))
verify_stats$Expected <- c(TRAIN_NUM_OF_ROWS_EXPECTED, VALIDATION_NUM_OF_ROWS_EXPECTED)
verify_stats$Actual   <- c(nrow(adult_income_train), nrow(adult_income_validation))

#2. check for NA in all columns
verify_stats$HasNA   <- c(any(is.na(adult_income_train)), any(is.na(adult_income_validation)))

colnames(verify_stats) <- c("", "Expected", "Actual", "Has Nulls")
kbl(verify_stats, booktabs = T, caption = "Basic data quality checks") %>%
kable_styling(latex_options = c("striped", "hold_position")) %>%
  column_spec(3, color = "red") %>%
  column_spec(4, color = "blue")
```

Table 1: Basic data quality checks

	Expected	Actual	Has Nulls
Number of row (training set)	32561	32560	FALSE
Number of row (test set)	16281	16280	FALSE

From the table, it appears that each dataset is off by one record. Since this is not a large number, I will ignore it. We also see that there are no NAs in each dataset.

Basic data quality check - min/max for numeric data

```
#3. numeric columns: check range of values, min/max

# function to check min/max
fnFindMinMax <- function(myFeature){
  data.frame("min" = min(myFeature), "max" = max(myFeature))
}

sapply( adult_income_train %>% select(where(is.numeric)), fnFindMinMax) %>%
  kbl(booktabs=T, caption="Numeric Variables min/max") %>%
  kable_styling(latex_options = c("striped", "hold_position")) %>%
  column_spec(0, bold=T) %>%
  row_spec(0, bold=T)
```

Table 2: Numeric Variables min/max

	age	final_weight	education_num	capital_gain	capital_loss	hours_per_week
min	17	12285	1	0	0	1
max	90	1484705	16	99999	4356	99

Basic data quality check - categorical data unique values

```
#4. for categorical values, inspect the unique values
fnShowUniqueValues <- function(myFeature){
  unique(myFeature)
}

adult_income_train %>% select(where(is.character)) %>% sapply(fnShowUniqueValues)
```

```
$workclass
[1] "Self-emp-not-inc" "Private"          "State-gov"        "Federal-gov"
[5] "Local-gov"        "?"                "Self-emp-inc"     "Without-pay"
[9] "Never-worked"

$education
 [1] "Bachelors"    "HS-grad"     "11th"        "Masters"     "9th"
 [6] "Some-college" "Assoc-acdm"  "Assoc-voc"   "7th-8th"     "Doctorate"
[11] "Prof-school"  "5th-6th"     "10th"        "1st-4th"     "Preschool"
[16] "12th"

$marital_status
[1] "Married-civ-spouse"    "Divorced"              "Married-spouse-absent"
[4] "Never-married"         "Separated"             "Married-AF-spouse"
[7] "Widowed"

$occupation
 [1] "Exec-managerial"   "Handlers-cleaners" "Prof-specialty"
```

```
 [4] "Other-service"    "Adm-clerical"     "Sales"
 [7] "Craft-repair"     "Transport-moving" "Farming-fishing"
[10] "Machine-op-inspct" "Tech-support"     "?"
[13] "Protective-serv"  "Armed-Forces"     "Priv-house-serv"


$relationship
[1] "Husband"        "Not-in-family" "Wife"          "Own-child"
[5] "Unmarried"      "Other-relative"


$race
[1] "White"             "Black"            "Asian-Pac-Islander"
[4] "Amer-Indian-Eskimo" "Other"


$sex
[1] "Male"   "Female"


$native_country
 [1] "United-States"            "Cuba"
 [3] "Jamaica"                  "India"
 [5] "?"                        "Mexico"
 [7] "South"                    "Puerto-Rico"
 [9] "Honduras"                 "England"
[11] "Canada"                   "Germany"
[13] "Iran"                     "Philippines"
[15] "Italy"                    "Poland"
[17] "Columbia"                 "Cambodia"
[19] "Thailand"                 "Ecuador"
[21] "Laos"                     "Taiwan"
[23] "Haiti"                    "Portugal"
[25] "Dominican-Republic"       "El-Salvador"
[27] "France"                   "Guatemala"
[29] "China"                    "Japan"
[31] "Yugoslavia"               "Peru"
[33] "Outlying-US(Guam-USVI-etc)" "Scotland"
[35] "Trinadad&Tobago"          "Greece"
[37] "Nicaragua"                "Vietnam"
[39] "Hong"                     "Ireland"
[41] "Hungary"                  "Holand-Netherlands"


$Income
[1] "<=50K" ">50K"
```

From the listings, there are no unusual values for either the numeric or the categorical data. As mentioned earlier, three variables (workclass, occupation, and native_country) do have the "?" as one of the values, meaning these three variables originally had null values which have been replaced. All the data, including the "?", will be used in the analysis.

Data Visualization and Exploration

Equal access is not equal gain

Even though one can argue that the U.S. population has equal access to education and employment, we know from experience that there are many factors or barriers that force people into certain income groups. These factors includes levels of education, gender, race, geographic areas people live in, to mention a few.

In this section, I explore the data to find general properties and patterns. The dataset contains many categorical features, so I am very interested in learning how income differs across different categories. Since the response variable is categorical, it's not possible to average incomes across different categories, however, the proportion of people with income over 50K across different categories can be compared.

Since there are many variables to explores, I focus on a few. The following questions were used as a guide for the data exploration.

Questions

1. Age

- What age group dominates the workforce? Which age group has the highest proportion of over 50K earners?
- Age can often determine how long a person has been in the workforce, and hence a higher income since as people stay longer in the workfoece, they advance in their careers, aften incressing their income. Do people in different age groups have different income? Do older workers earn more than younger workers?

2. Sex

- The dataset is distributed between males and females workers; what proportion of each sex is represented in the data?

- Do the proportions represent the gender composition in the the general U.S. population?

- Which sex tends to have higher income?

3. Level of education

- Level of education can also be a strong determinant of one's income. Does the data suggest that people with higher levels of education have higher income?

4. Capital Gain/Capital Loss

- Can the amount of capital gain or loss indicate an individual's income?

Data Distribution

Prevalance of people earning over 50K. Out of the 32560 records in the training set, only 24% (7841) has income over 50K. Therefore, this sample has a much lower percentage of people earning over 50K and does not exactly reflect the current US income distribution which has about 45% of people earning over 50K[1].

---

[1]https://www.cs.mcgill.ca/~rwest/wikispeedia/wpcd/wp/h/Household_income_in_the_United_States.htm

Numeric Variables Distribution   I start by visualizing the distribution of the numeric features.

```
adult_income %>% select(where(is.numeric)) %>% summary()
```

```
      age          final_weight     education_num    capital_gain
 Min.   :17.00   Min.   :  12285   Min.   : 1.00   Min.   :    0
 1st Qu.:28.00   1st Qu.: 117832   1st Qu.: 9.00   1st Qu.:    0
 Median :37.00   Median : 178363   Median :10.00   Median :    0
 Mean   :38.58   Mean   : 189782   Mean   :10.08   Mean   : 1078
 3rd Qu.:48.00   3rd Qu.: 237055   3rd Qu.:12.00   3rd Qu.:    0
 Max.   :90.00   Max.   :1484705   Max.   :16.00   Max.   :99999
  capital_loss     hours_per_week   capital_gain_loss
 Min.   :   0.00   Min.   : 1.00   Min.   :-4356.0
 1st Qu.:   0.00   1st Qu.:40.00   1st Qu.:    0.0
 Median :   0.00   Median :40.00   Median :    0.0
 Mean   :  87.31   Mean   :40.44   Mean   :  990.3
 3rd Qu.:   0.00   3rd Qu.:45.00   3rd Qu.:    0.0
 Max.   :4356.00   Max.   :99.00   Max.   :99999.0
```
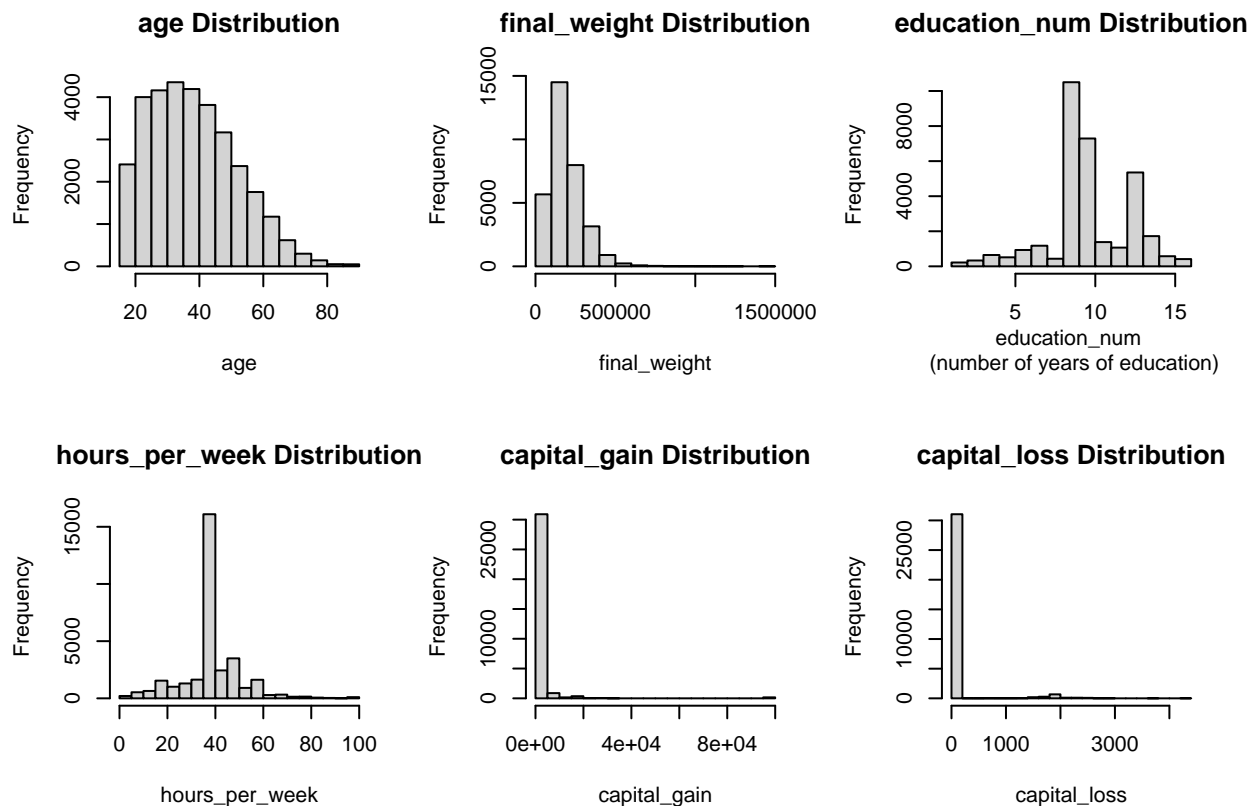

In general there is a wide range of values for the different variables.

Among the six numeric variables, variable *final_weight*, has the widest range (`12285 - 1484705`). This is followed by the *capital_gain* and *capital_loss*; all three variables are also very highly right-skewed.

Variables *capital_gain* and *capital_loss* both have a mean and median of zero and a very small averages, reflecting on the large number of records having a value of zero (no capital gain or loss). In fact for *capital_gain*, only 8% reporting gains and 92% reported a value of zero.

Variable *education_num* is the more normally distributed than the rest, followed by *age*, and *hours_per_week*, even though *age* is still right-skewed.

The *hours_per_week* variable is a little bit of bell curve with a mean and median being same 40, meaning the majority of people are working fourty hours a week. However, one surprising thing is the presence of near 100hrs per week for some individuals. Although not a significant proportion, it is important to note that 789 individuals in this dataset work over seventy hours or more per work.

The *age* summary statistics and the histogram show a right-skewed distribution with a mean of 39 and median of 37 years. This indicates, as expected, that the bulk of the workforce is concentrated in the younger age groups[2].
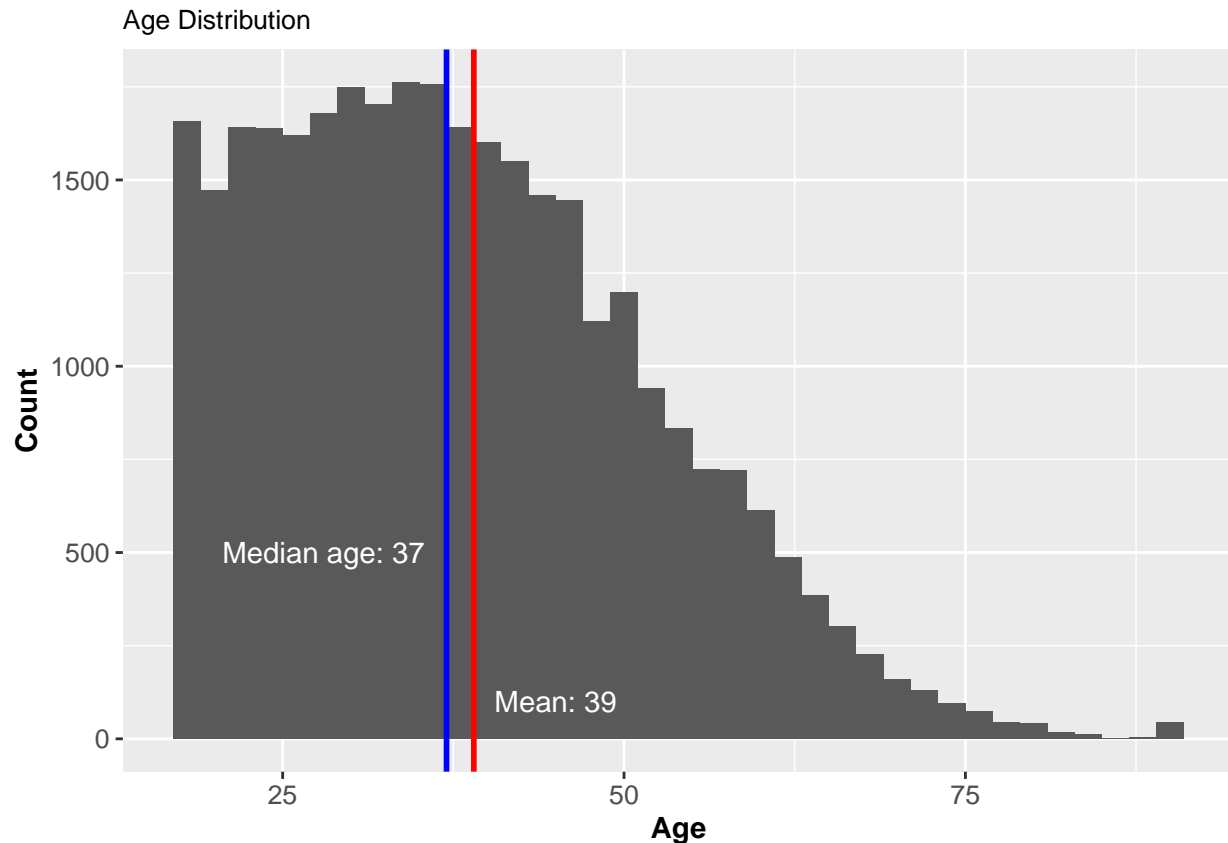
It can be seen from the summary statistics that while the minimim age in the workforce is 17 we see some ninety year olds still in the workforce; hopefully these seniors are not drawn to work by the need for income but rather by the need to stay active.

---

[2]https://www.bls.gov/careeroutlook/2017/article/older-workers.htm

Categorical Variable Distributions **Race and gender**. With race, the largest proportion is White 85%. The rest is divided among Black (10%), and the others.

When it comes to gender, the sample is predominantly *male* which makes up 67% of the sample; the rest is *female*.

Education. Education is one of the categorical variables with enourmous amount of space (16 levels). The categories range from Preschool to Doctorate degree. High school diploma is the largest category (32%), followed by *Some-college* (22%), and *Masters* (16%).

In addition to *education* categorical variable, the *education_num* has the number of years of education for each individual. In essence these two vaiables may be reporting the same information.

When the categories are collapsed into more broader categories with fewer levels, it becomes even more clear that the vast majority of the individuals have no college degree. This may explain why the majority of the individuals earn less than 50K.

Education Distribution – Original Categories

Education Distribution using more general categories
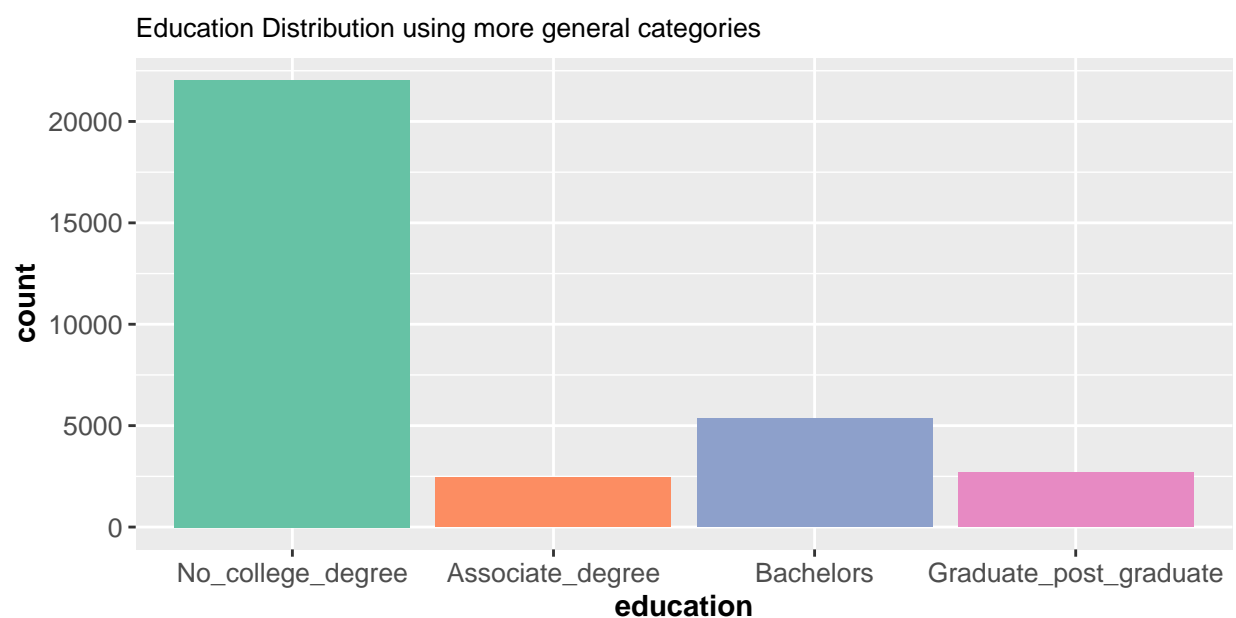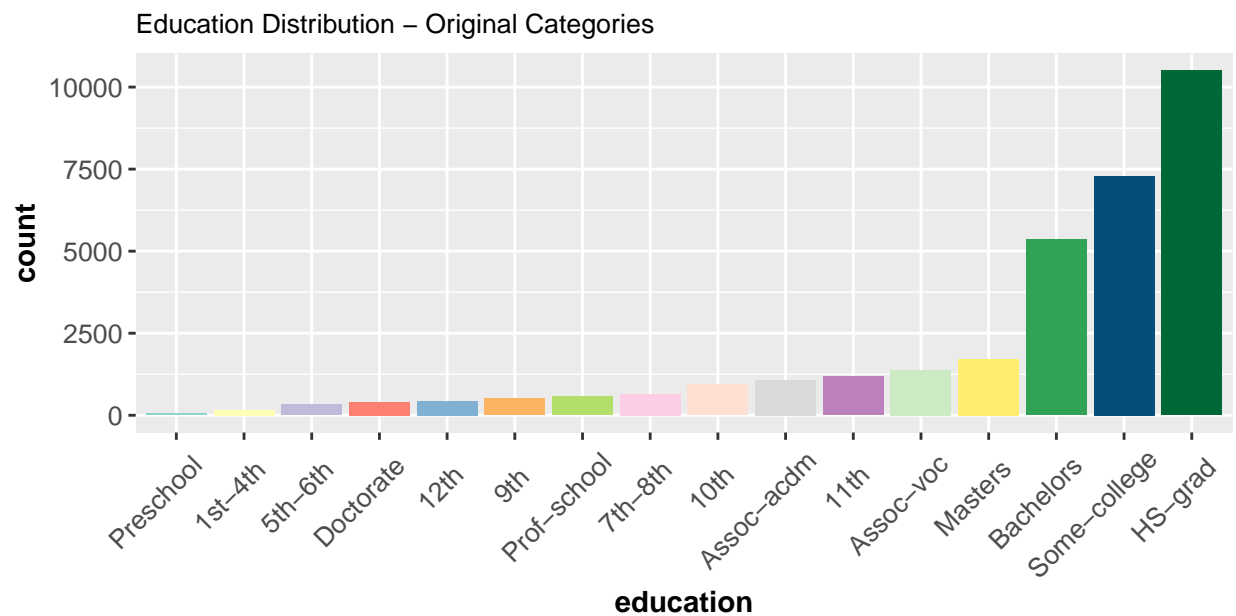
Workclass. The *workclass* variable is more like job sectors, has eight categories and one missing or not specified (?) category. The largest category is Private (`70%`), the rest is split among the other categories. There are some outliers (Never-worked, and Without-pay) with a counts of `7` and `14` repectively.

Occupation. Another import variable is *occupation*, a categorical variable with 15 levels, one missing (?) category.

Table 3: Occupation by Category - Orginal Categories

| Category	Count
?	1843
Adm-clerical	3769
Armed-Forces	9
Craft-repair	4099
Exec-managerial	4066
Farming-fishing	994
Handlers-cleaners	1370
Machine-op-inspct	2002
Other-service	3295
Priv-house-serv	149
Prof-specialty	4140
Protective-serv	649
Sales	3650
Tech-support	928
Transport-moving	1597

Occupation Distribution
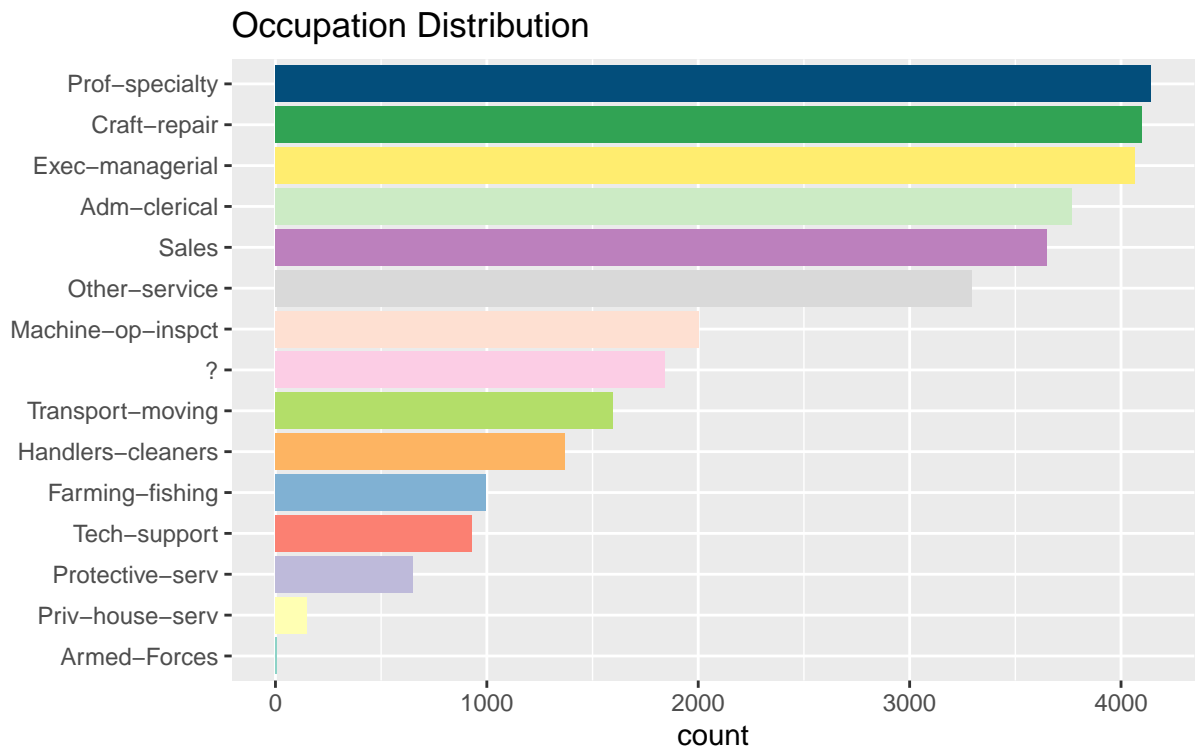
For visualization purposes, I merged the *occupation* categories into new categories based on the occupation classifications defined by US Bureau of Labor Statistics (BLS). These categories (classifications) are known as the Standard Occupation Classifications (SOC) [3].

The SOC defines many categories but the following are the ones used:

- Management_business_science_and_arts
- Service
- Sales_and_Office
- Natural_resources_construction_and_maintenance
- Production_transportation_and_material_moving

Table 4: Occupation by Category Based on SOC Classifications

| Category	Count
Unspecified	1843
Sales_and_Office	8347
Armed-Forces	9
Natural_resources_construction_and_maintenance	7095
Management_business_science_and_arts	8206
Service	5463
Production_transportation_and_material_moving	1597

Occupation by Category
Based on SOC Classifications
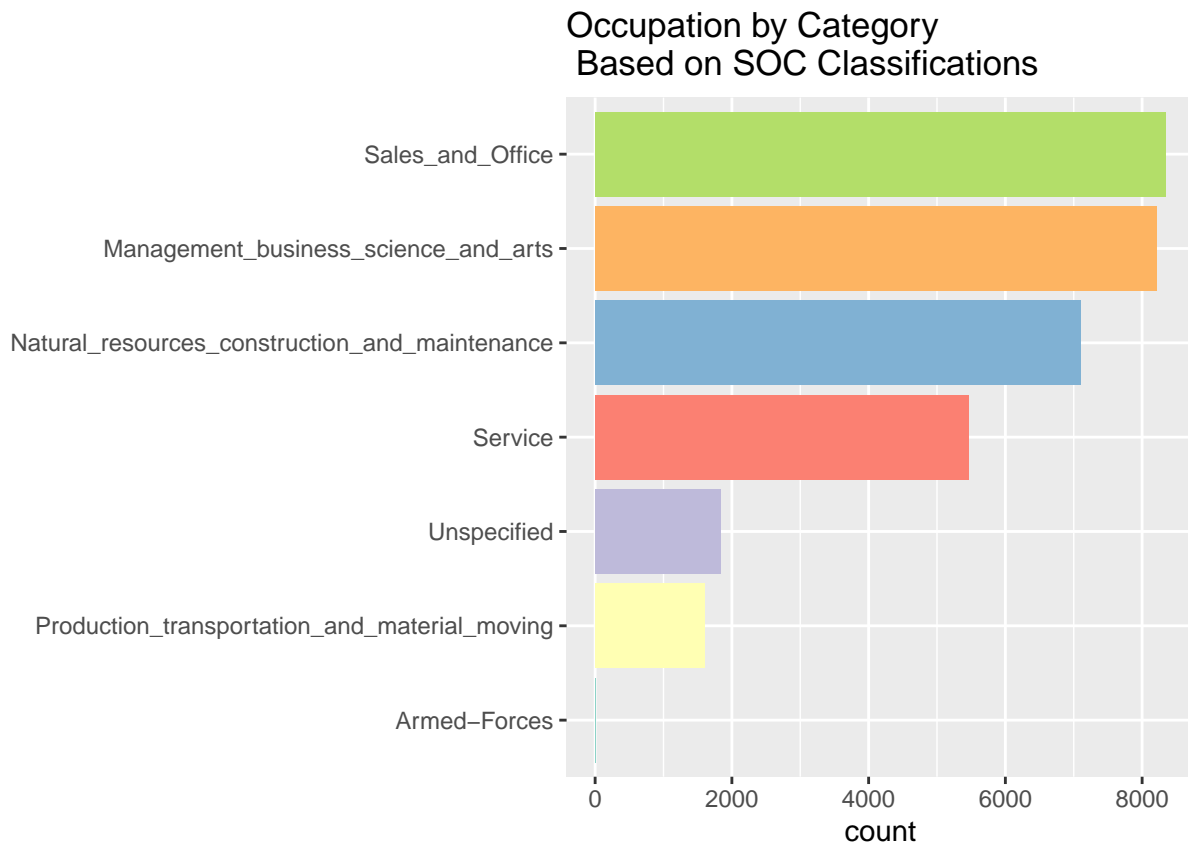
Trends

In this section I will explore the data in an attempt to find some trends.

Income and Age. One of the questions asked earlier was whether age can indicate an individual's income level. The graph below shows that indeed, overall, there is a positive relationship. The proportion of individuals earning over 50K increases with age and peaks around the fifties, before starting to decline.


Proportion of >50K earners at different ages
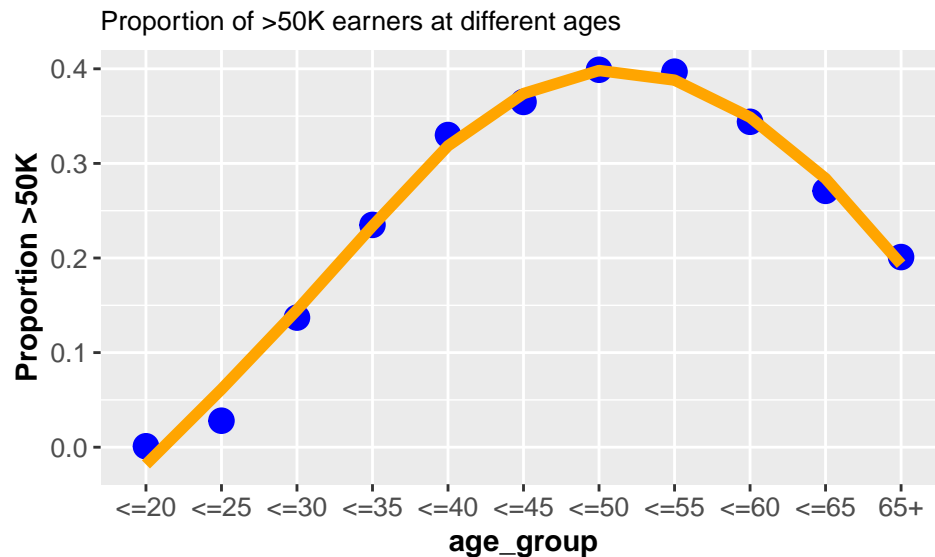
Income and education. It is generally accepted that income level is associated with educational attainment; the higher the educational level, the higher the income. Does the data support this belief? Based on the data, at every educational levels (except graduate and post graduate), the number of individuals earning >50K is lower than those earning >50K.

Number of people earning >50K exceeds earning <=50K only with graduate degree and above
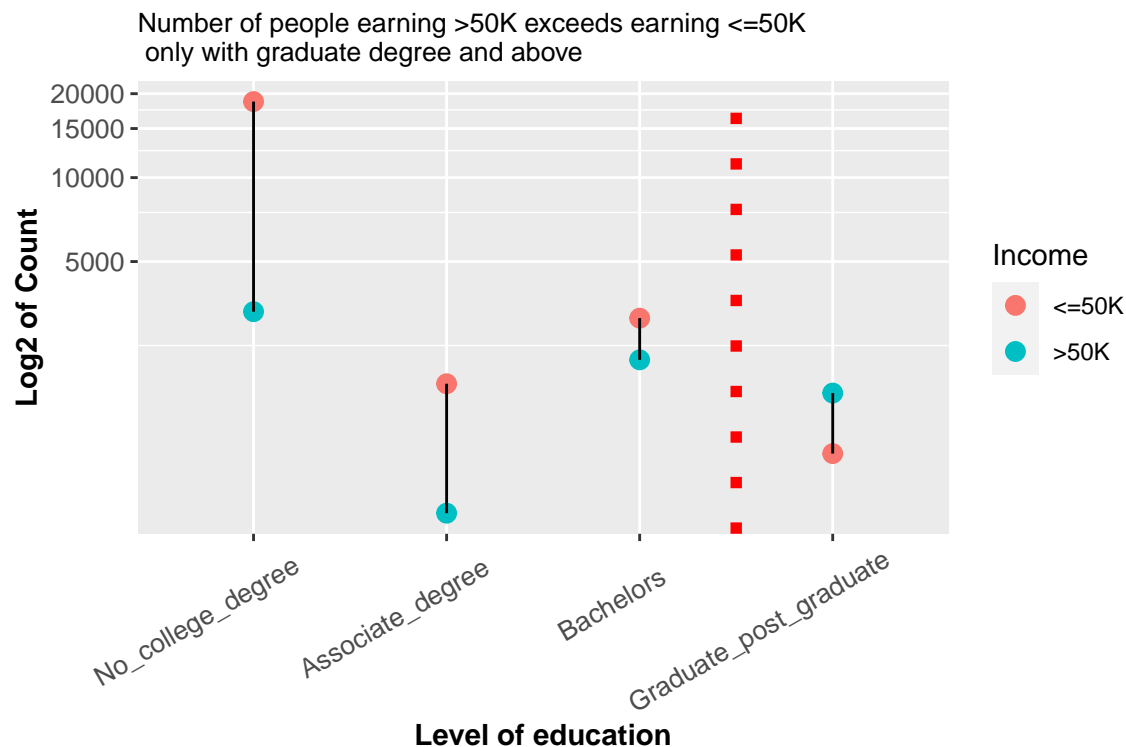
However, as the gragh below shows, the proportion of individuals earning >50K is actually increasing and the rate gets steeper after an individual attains some college level education. Therefore, education seems to have a positive effect on income.

Proportion of >50K earners is increasing
as educational level increases
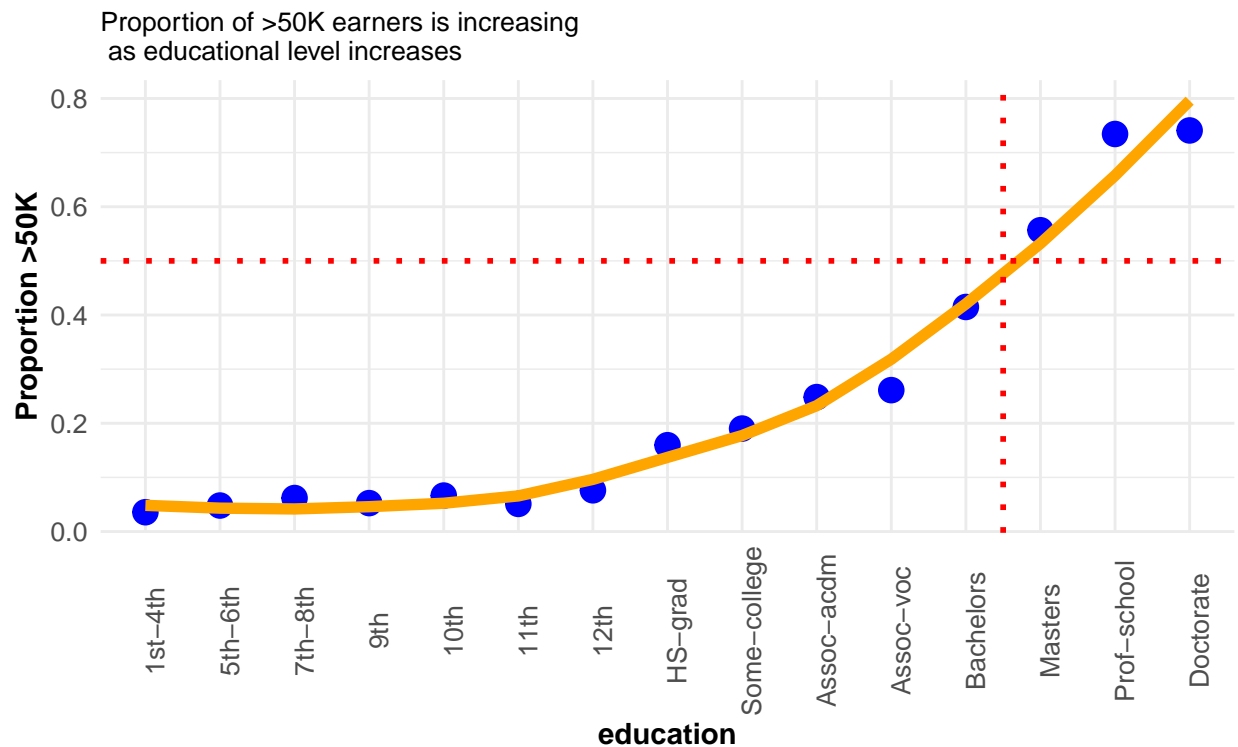
It is important to note that the graph above can be generated using *education_num* instead of the categorical variable *education* (see chart below). This is because both *education_num* and *education* have the same information. Since these two variables may be correlated, *education_num* will be excluded in final analysis.

Proportion of >50K earners at different ages
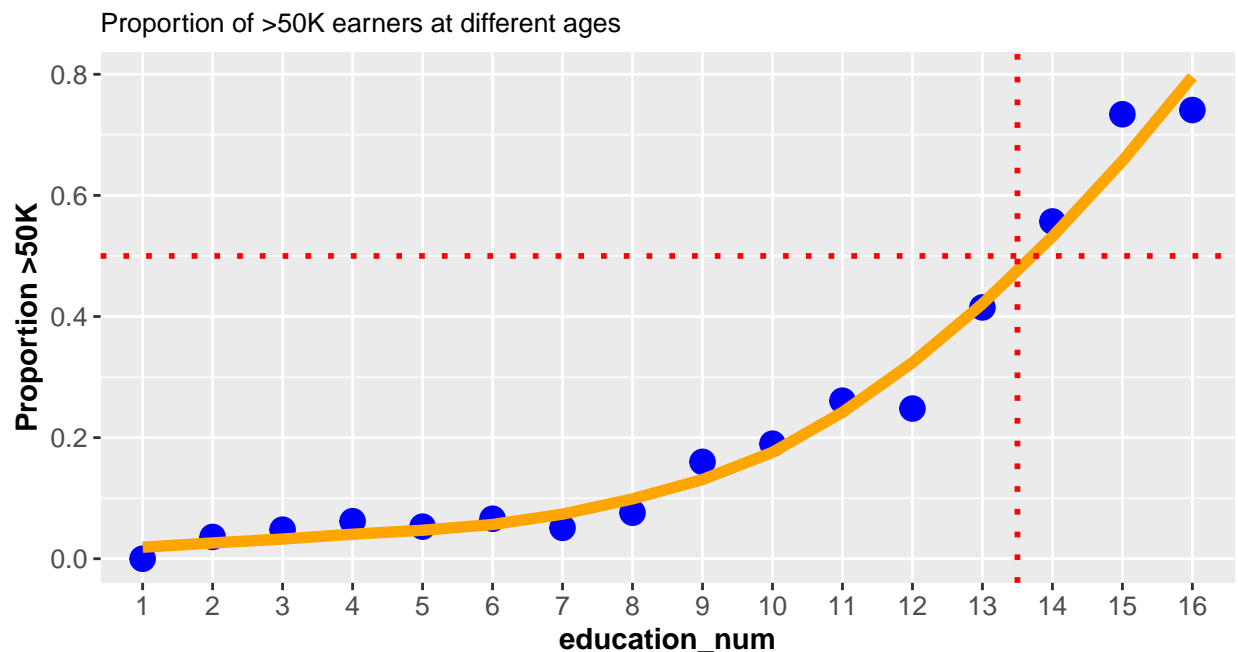
Income by gender . Gender and income inequality has always been a topic of concern in almost every society. This section shows a simple analysis of how income differs by gender in the dataset. The table below shows that in general, a higher proportion of males earn higher income than females.

Table 5: Income and Gender

Gender	Total	Total over50K	Proportion >50K
Female	10771	1179	0.11
Male	21789	6662	0.31

Education wise, the average number of years spent in education is the same for both males and females (see boxplot below right). But the males have a higher proportion of people with education above the average than the females, which could translate into higher incomes for males.

Table 6: Gender, Avg Age, and Education

Gender	Avg Age	Avg Education Yrs
Female	37	10
Male	39	10

Income by gender
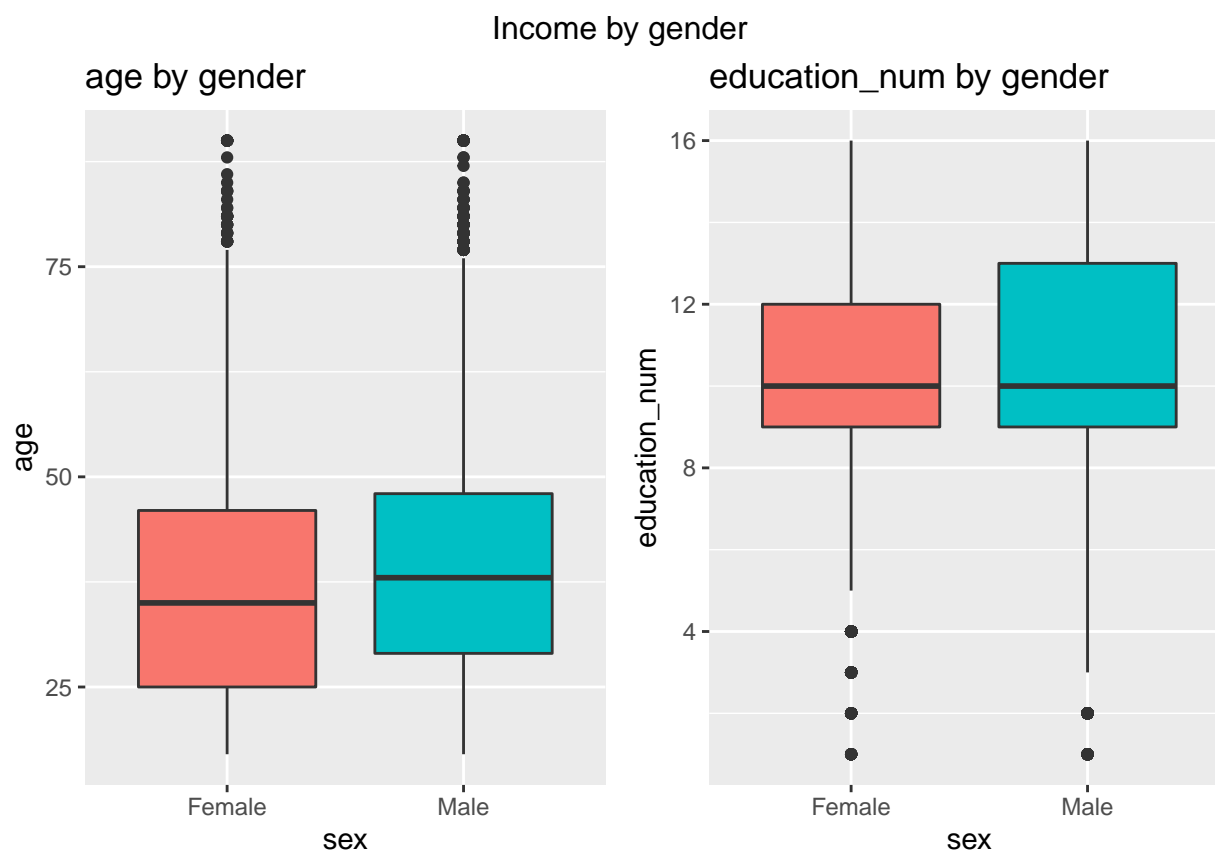
Income and capital gain/capital loss. Another set of variables I wanted to explore is the *capital_gain* and *capital_loss*. In general a vast majority of individuals did not report a gain or loss. And even with those that reported a gain, there is a wide range for the values reported. Do people who reported capital gain have higher income or vice versa?

The following chart and tables explore if capital gain or loss can provide an indication of one's income level.
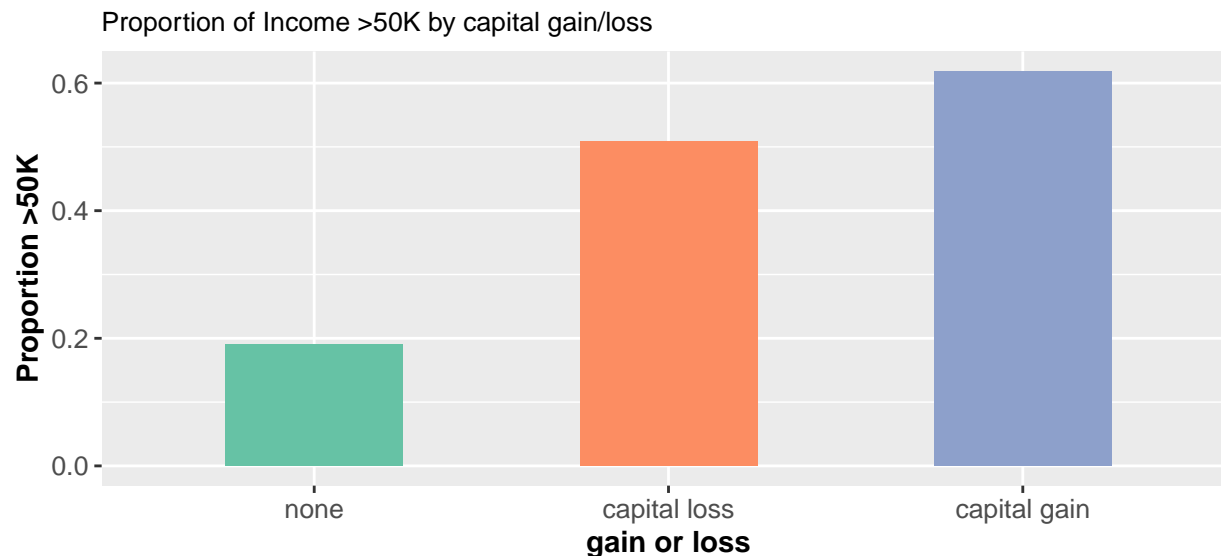
Proportion of Income >50K by capital gain/loss

Table 7: Proportion of Income over50K by Capital Gain

capital_gain_level	total	over50K	proportion
none	29849	6164	0.21
<=10K	1941	921	0.47
<=20K	517	511	0.99
>20K	253	245	0.97

Table 8: Proportion of Income over50K by Capital Loss

capital_loss_level	total	over50K	proportion
none	31041	7068	0.23
<=1K	36	2	0.06
<=2K	1158	629	0.54
<=3K	314	139	0.44
>3K	11	3	0.27

It can be noted from the chart above that individuals who reported no capital gain or loss have a higher probability of earning <=50K. Individuals reporting capital gains have a higher probability of earning >50K. From the tables, it can be seen that Individuals with a gain of 20K and above are almost guaranteed to earn >50K. In fact, the maximum capital gain reported, 99999, is reported by 159 individuals; out of those indivduals, not a single one reported an income <=50K.

```
adult_income %>% filter(capital_gain==max(capital_gain)  & Income=="<=50K" ) %>% nrow()
```

```
[1] 0
```

Analysis

Approach

This section presents the analysis, approach and results. The proposed approach is outlined below:

0. Data partitioning (train/test)
1. Data Engineering, Transformation & Preprocessing
2. Training, Optimization & Model Selection
3. Final model and final prediction on validation set

Data Partitioning (train/test)

The training data (adult_income_train dataset) is split into train/test (90/10 partitions).

Data Engineering, Transformation & Preprocessing

The datasets used in this project had been cleaned by the original contributors, however, a few tasks were performed to prepare the data for the final analysis.

1. replace question marks in categorical data. The question mark "?" values were replaced with the value *Unspecified* in the three variables that had this value.

2. remove special characters. Remove special charactes like the ampersand (&) and hyphen (-) from categorical values.

3. remove period from the values in the response variable Income. The validation dataset had period "." in the values of the response variable, Income.

```
unique(adult_income_validation$Income)
```

```
[1] "<=50K." ">50K."
```

4. Convert the values in the response variable to Y,N. Before the analysis, the values in the response (Income) was converted to Y/N: (Y equals >50K, N: equals <=50K) and with Y being set as the positive.

5. convert all categorical variables into dummy variables. All categorical variables were converted into numeric (0,1) dummy variables using one-hot encoding from the caret package.

6. log transformations. Some variables (*age* and *final_weight*) were log2 transformed before scaling.

7. new column for capital gain/loss. The *capital_gain* and *capital_loss* variables were combined into a single column called *capital_gain_loss*. The values then range from positive (capital gain) to negative (capital loss). In addition, values were all adjusted to remove zeros and then log2 transformed.

6. scale all numeric variables. After all the transformation, the variables were numeric; the final step was to scale them before the analysis.

The tasks for perorming the transformation are all placed in R functions so they can be reused.

```r
################################################################################
#
# PREPROCESSING PIPELINE
# TAKE ALL DATASETS THROUGH THE SAME TRANSFORMATION
#
################################################################################
# 1. rename some categorical values
training_set_final <- fnManualTransformations(train_set)
test_set_final     <- fnManualTransformations(test_set)

# 2. create caret preprocessor for dummy variables
caret_dummy_vars_preprocessor <- fnCreateCaretDummyVarsPreproc(training_set_final)
training_set_final <- fnApplyCaretDummyVarsPreproc(training_set_final,
                                                   caret_dummy_vars_preprocessor)

test_set_final     <- fnApplyCaretDummyVarsPreproc(test_set_final,
                                                   caret_dummy_vars_preprocessor)

# 3. cosmetic stuff
# convert response var to Y,N, and set the positive class to Y (over50K)
training_set_final <- fnEncodeResponseVar(training_set_final)
test_set_final     <- fnEncodeResponseVar(test_set_final)

# 4. perform final caret preprocess - center and scale
caret_scale_preprocessor <- fnCreateCaretScalePreproc(training_set_final)
training_set_final <-  fnApplyCaretScalePreproc(training_set_final,
                                                caret_scale_preprocessor)

test_set_final     <-  fnApplyCaretScalePreproc(test_set_final,
                                                caret_scale_preprocessor)
```

After all the transformations are performed on the datasets, the training data looks are below:


Figure 2: training data after transformation and scaling

The summary, and scatterplots, and bpxplots below show the distribution of the new scaled variables.

```
training_set_final %>%
  select(c(age, final_weight, education_num, hours_per_week, capital_gain_loss)) %>%
  summary()
```

```
      age              final_weight       education_num       hours_per_week
 Min.   :-2.10114   Min.   :-4.0592    Min.   :-3.52142    Min.   :-3.20698
 1st Qu.:-0.71563   1st Qu.:-0.4825    1st Qu.:-0.42088    1st Qu.:-0.03757
 Median : 0.05826   Median : 0.1732    Median :-0.03332    Median :-0.03757
 Mean   : 0.00000   Mean   : 0.0000    Mean   : 0.00000    Mean   : 0.00000
 3rd Qu.: 0.78097   3rd Qu.: 0.6229    3rd Qu.: 0.74182    3rd Qu.: 0.36876
 Max.   : 2.52638   Max.   : 3.5273    Max.   : 2.29208    Max.   : 4.75718
 capital_gain_loss
 Min.   :-21.9503
 1st Qu.: -0.1536
 Median : -0.1536
 Mean   :  0.0000
 3rd Qu.: -0.1536
 Max.   :  8.1079
```


age Distribution (scaled)
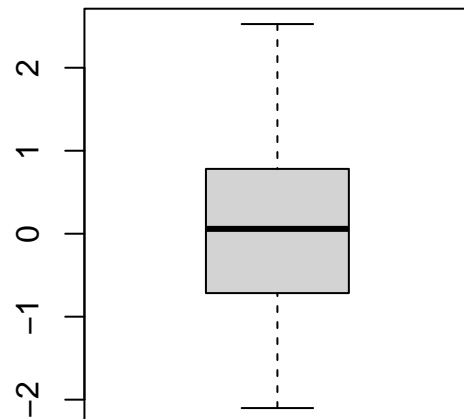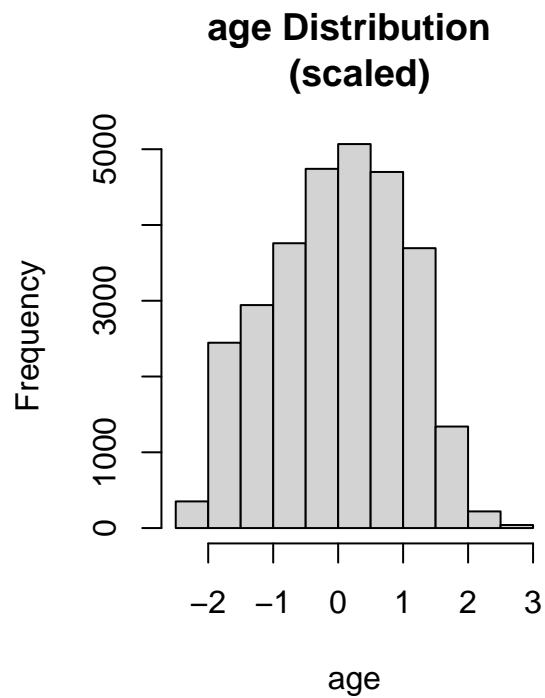
education_num Distribution
(scaled)

education_num
(number of years of education)
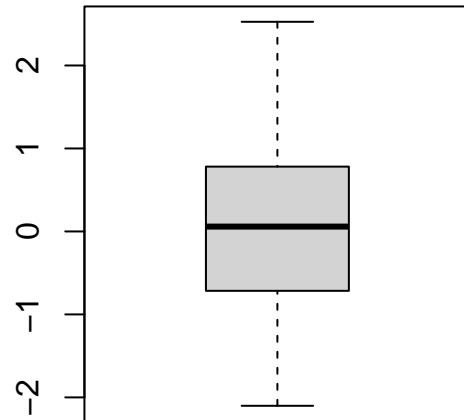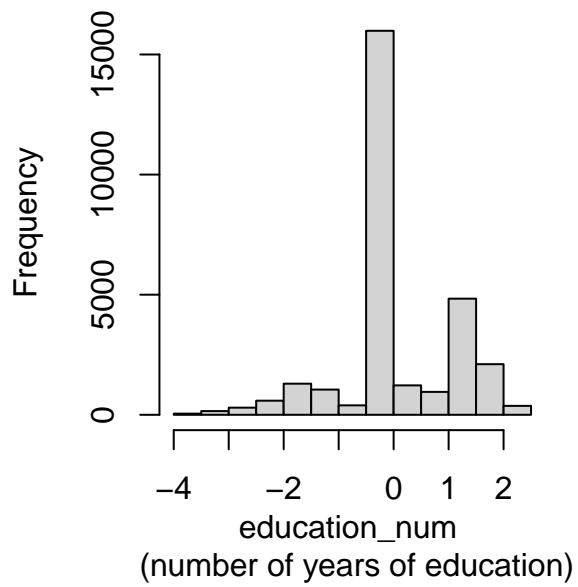
final_weight Distribution
(scaled)

final_weight

hours_per_week Distribution (scaled)

capital gain/loss Distribution (scaled)

Training, Optimization, Final Model, and Results

Datasets The following table lists the datasets as used in the final part of the project.

Table 9: Final datasets for analysis and validation

data	numrows
training_set_final	29303
test_set_final	3257
validation	16280

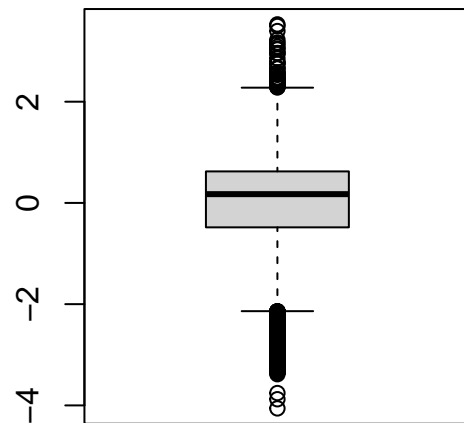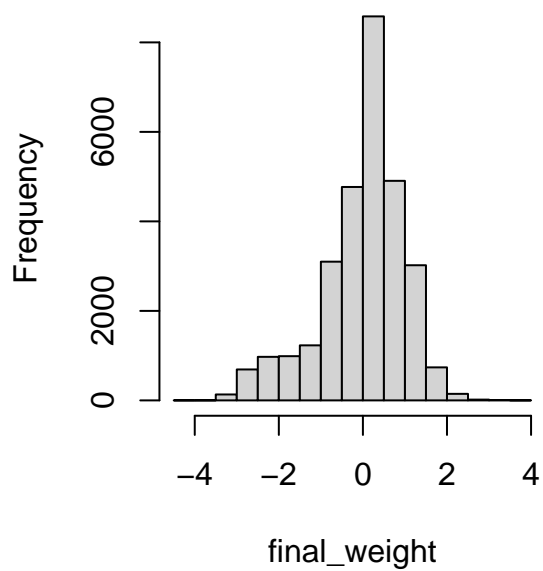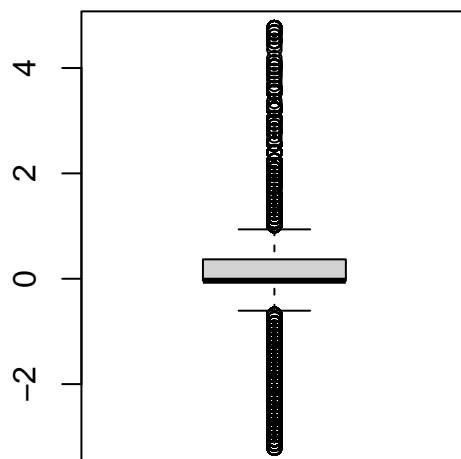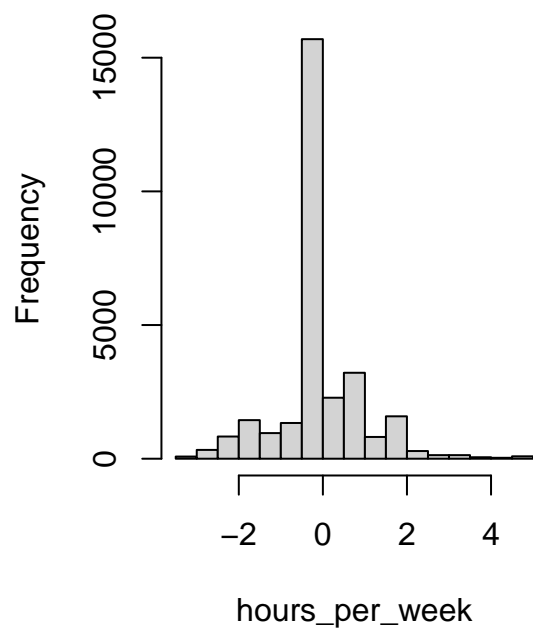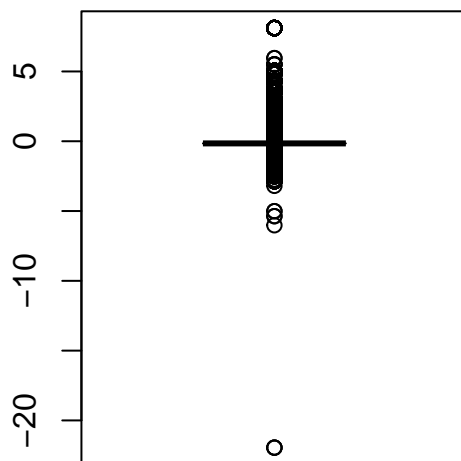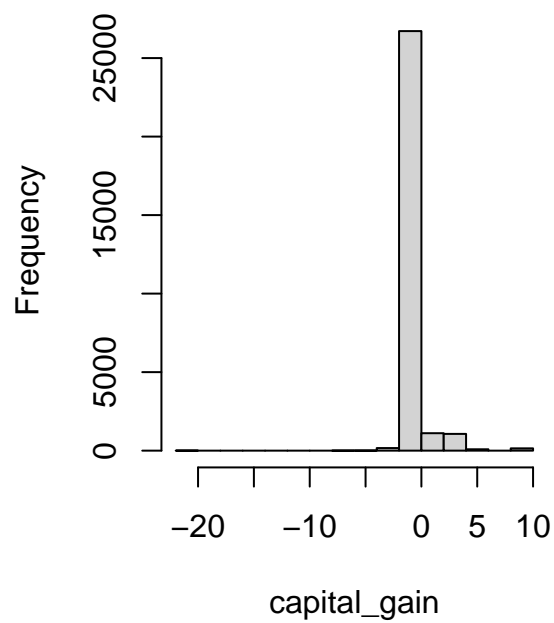**Cross Validation and Sampling.** To reduce the chances of overfitting, a 5-fold cross validation was used while training the models using a sample size of 5000 from the training set.

The following parameters are used to define a common training control for all models:

```
cv_control <- trainControl(method = "cv",
                           number = 5,
                           summaryFunction=twoClassSummary,
                           classProbs=TRUE,
                           savePredictions = T)
```

The *summaryFunction* is set to *twoClassSummary*, *classProbs=TRUE*, both are required when using ROC as the performance metric. *savePrediction* is set to *TRUE* so caret will retain the predictions from the final model, which can be used later.

Performance metric. In this projet, I use *AUC* (Area Under the Curve) as the metric for selecting best models during cross validation. In th caret training function, the *metric* will be set to **ROC**, to indicate to caret to select best model using *AUC* rather than Accuracy.

When a model is fitted on a test, I use Kappa instead of accuracy when selecting best model. Kappa is often preferred to accuracy when dealing with imbalanced dataset which is the case in this project.

In the analysis, I have intentionally set the positive class to "Y" (>50K). Even though in terms of income, both classes (>50K and <=50K) both have equal importance; mistakingly predicting that someone earns <50K does not pose a risk to the individual. Howvever, I want to place emphasis on the positive class and pay attention to how the models are able to distinguish between the classes.

Due to the high imbalance of the data, (the positive class, over50K (Y) only occurs 24% in the dataset), any algorithm can achieve 76% accuracy by picking (guessing) the class that occurs most. The overall accuracy, therefore, is not the best performance measure in this case. using Kappa is recommended as an alternative to using the overall accuracy[4].

Kappa uses the overall *predictive accuracy* which is pretty much a ratio of correct predictions to all predictions. The overall accuracy is then adjusted by factoring into the equation, the *possibility of predicting the correct class by chance*. The details of how to compute Kappa is beyond the score of the project.

Kappa values range from 0 to 1 with values above 0.5 being considered moderate to very good. Values below 0.5 are considered fair to poor performance[Nwagana, 324]

---

[4]Nwanganga & Chapple, (2020), pp. 323-325

KNN Models

First, I trained two KNN models and picked the best performing one. My approach was, first, I trained a model using caret defaults (knn_model_1). In kNN, there is only one parameter to tune k. In the default model, I used a tuneLength of 10 to allow caret to select 10 values for k. This model provided a *best* value for k from the cross validation. I fit that model to the entire training set, then use the fit to make predictions on the test.

I take the *best k* from the first model, and expanded around it to provide more tuning values for k and used it to train another model (knn_model_2). I fitted the new model on the entire training set, and made another predictions on test. I then compared the two models.

```
knn_model_1 <- train(
  over50K ~ .,
  method = "knn",
  metric="ROC",
  data = train_sample,
  trControl = cv_control,
  tuneLength=10
)
```

```
knn_k_tuning_grid   <- data.frame(k = c(seq(1, 39, 2), 50, 100, 150, 200, 250, 350))
knn_model_2 <- train(
  over50K ~ .,
  method = "knn",
  metric="ROC",
  data = train_sample,
  trControl = cv_control,
  tuneGrid = knn_k_tuning_grid
)
```

The following shows the two models, the cross validation results, and confusion matrix.

Table 10: knn model 1 - cross validation results

k	ROC	Sens	Spec	ROCSD	SensSD	SpecSD
5	0.8298756	0.5268029	0.9035800	0.0219731	0.0208404	0.0212336
7	0.8423998	0.5309799	0.9156656	0.0217475	0.0245849	0.0174451
9	0.8488515	0.5284835	0.9193426	0.0206718	0.0210215	0.0154721
11	0.8520937	0.5167540	0.9198685	0.0203978	0.0202878	0.0137352
13	0.8534034	0.5025034	0.9182917	0.0240775	0.0166372	0.0162282
15	0.8566805	0.4974825	0.9253869	0.0232774	0.0105069	0.0155962
17	0.8587579	0.5092121	0.9264378	0.0262555	0.0165783	0.0154192
19	0.8601051	0.4974860	0.9256483	0.0239880	0.0264794	0.0149933
21	0.8609908	0.4999719	0.9266989	0.0238443	0.0371619	0.0160217
23	**0.8620224**	**0.4949650**	**0.9285375**	**0.0251038**	**0.0387958**	**0.0161928**

Table 11: knn model 2 - cross validation results

k	ROC	Sens	Spec	ROCSD	SensSD	SpecSD
1	0.7065225	0.5528251	0.8602198	0.0211425	0.0431591	0.0210851
3	0.7927070	0.5100875	0.8893868	0.0140070	0.0274657	0.0156270
5	0.8264031	0.5251644	0.8980575	0.0119190	0.0372817	0.0082596
7	0.8394166	0.5243205	0.9056763	0.0118339	0.0299509	0.0113835
9	0.8447791	0.5125804	0.9098793	0.0098401	0.0225648	0.0093181
11	0.8473865	0.5042087	0.9143446	0.0092301	0.0186361	0.0107425
13	0.8518287	0.4924721	0.9177602	0.0108106	0.0250892	0.0111068
15	0.8541247	0.5025562	0.9211767	0.0124063	0.0329162	0.0138158
17	0.8569977	0.4966984	0.9203883	0.0131958	0.0393789	0.0127310
19	0.8570346	0.5000492	0.9227526	0.0116748	0.0381563	0.0106190
21	0.8593500	0.4899828	0.9245933	0.0120677	0.0362205	0.0095881
23	0.8592190	0.4883091	0.9251186	0.0119774	0.0338225	0.0090530
25	0.8585884	0.4849689	0.9248557	0.0110762	0.0447794	0.0113170
27	0.8606909	0.4908266	0.9240680	0.0102617	0.0402766	0.0113888
29	0.8633550	0.4908126	0.9256438	0.0092480	0.0412141	0.0117743
31	0.8639854	0.4958405	0.9243301	0.0094042	0.0419179	0.0105359
33	0.8643207	0.4883091	0.9238055	0.0082074	0.0455172	0.0110241
35	0.8647594	0.4899863	0.9264330	0.0082529	0.0414680	0.0092811
37	0.8641838	0.4925073	0.9264312	0.0082479	0.0411547	0.0087730
39	0.8645409	0.4908372	0.9293204	0.0078083	0.0443906	0.0088008
50	0.8639559	0.4841426	0.9282682	0.0084840	0.0402629	0.0109596
100	0.8661886	0.4497943	0.9353662	0.0117490	0.0339988	0.0133107
150	0.8690080	0.4841285	0.9311639	0.0120177	0.0301425	0.0083111
200	0.8744032	0.4791006	0.9337924	0.0113771	0.0352903	0.0078709
250	0.8785073	0.4606835	0.9403617	0.0095762	0.0454341	0.0097738
350	**0.8794772**	**0.4405612**	**0.9479811**	**0.0079969**	**0.0338225**	**0.0093730**

The following show the graphs of the cross validation results.

cross validation results – knn models 1 & 2
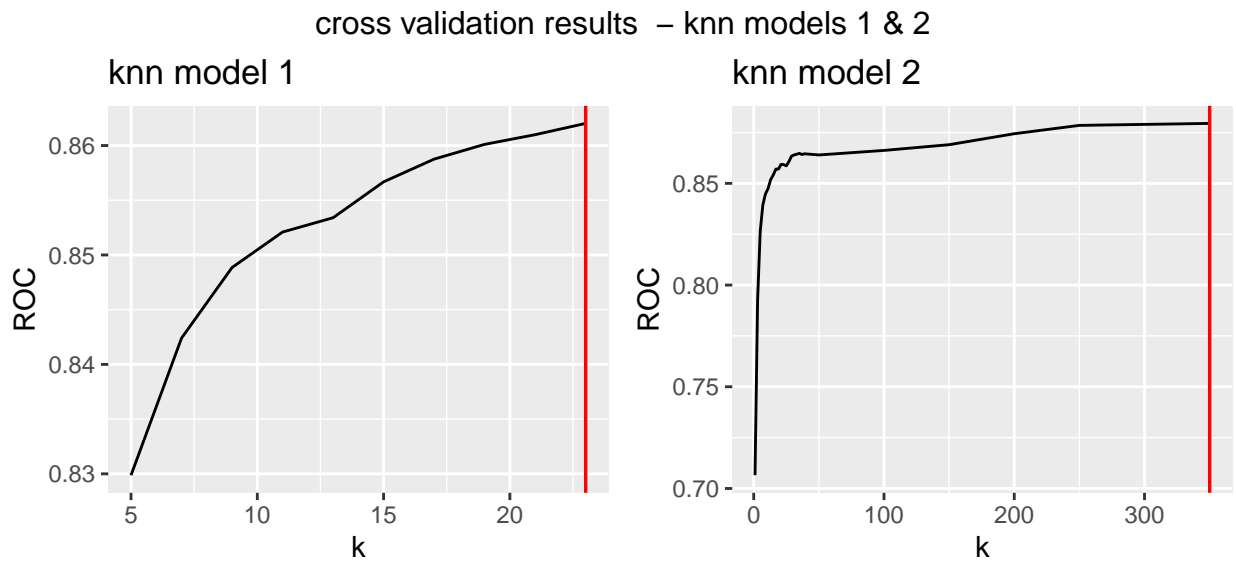
From the cross validation results in first model, it appear k is still increasing and might even yield a better result if we increase k. But from the second model, it's clear that increasing k does not yield better results. In fact the first model using caret default is better. The following details from the confusion matrix show how both models perform on the test set.

ConfusionMatrix on test set – knn models 1 & 2

	term	estimate		term	estimate
1	accuracy	0.8317470	1	accuracy	0.8289837
2	kappa	0.4987134	2	kappa	0.4737755
3	sensitivity	0.5299363	3	sensitivity	0.4815287
4	specificity	0.9275890	4	specificity	0.9393204
5	pos_pred_value	0.6991597	5	pos_pred_value	0.7159091
6	neg_pred_value	0.8613824	6	neg_pred_value	0.8508611
7	precision	0.6991597	7	precision	0.7159091
8	recall	0.5299363	8	recall	0.4815287
9	f1	0.6028986	9	f1	0.5757807
10	prevalence	0.2410193	10	prevalence	0.2410193
11	detection_rate	0.1277249	11	detection_rate	0.1160577
12	detection_prevalence	0.1826835	12	detection_prevalence	0.1621124
13	balanced_accuracy	0.7287627	13	balanced_accuracy	0.7104245
14	accuracy.conf.low	0.8184496	14	accuracy.conf.low	0.8156054
15	accuracy.conf.high	0.8444457	15	accuracy.conf.high	0.8417684
16	no_information_rate	0.7589807	16	no_information_rate	0.7589807

We can see from the details above that in model 1 (left table), accuracy, kappa, and sensitivity are higher than those in model 2 (right table). Therefore, I selected knn_model_1 as best model from knn. Next I will train other models using randomForest and select the best model again from there. I will then pick from two final models, the best performing model and use as my final model for validation.

RandomForest Models

RandomForest Model 1. Using randomForest defaults for training. In this section, I used the same approach as used when training the kNN models. I used a sample (5K observations) to train the models. There are two main parameters of interest to me; *mtry* (number of random features to use in each split), and nodesize (minimum number of nodes in the terminal nodes or leaves). Number of tree (ntree) can also be set for randomForest, but in this case, I let caret choose values for ntree.

First, I train a model using caret defaults (*rf_model_1*), then take the best mtry from that model, expand on it and use it to train another model (*rf_model_2*) to see if there is an *mtry* that yields a better performance. I then use the new mtry value and train another model (rf_model_3), this time searching for an optimized *nodesize.* At the end, I select the best model among these three, and compare that to the one from kNN.

```r
# use caret defaults to train rf model using cross validation
# use ROC as a metric to for selecting best model
fnTrainRandomforestWithDefaults <- function(train_sample){
  train(
    over50K ~ .,
    method = "rf",
    metric="ROC",
    data = train_sample,
    trControl = cv_control,
    tuneLength=10,
    allowParallel=TRUE
  )
}
```

Again I set *tuneLength* to 10 to allow caret to generate 10 random values for mtry (by default caret will pick three values for mtry).

The training model was then used to fit a model on the entire training set, and used it to make predictions on the test set. The results below show the results from the cross validation. The cross validation results show that the ROC ranges from 88% to 90%. The best cross validation model achieves ROC of 90% with mtry of 21, at 61% sensitivity, and 93% specificity.

Table 12: Cross validation results - randomForest model 1

mtry	ROC	Sens	Spec	ROCSD	SensSD	SpecSD
2	0.8851525	0.0108998	1.0000000	0.0100475	0.0117177	0.0000000
11	0.9039384	0.6139095	0.9353700	0.0131581	0.0183875	0.0113521
21	**0.9055073**	**0.6356809**	**0.9238083**	**0.0125443**	**0.0087221**	**0.0123971**
31	0.9044757	0.6281425	0.9217061	0.0121216	0.0122555	0.0139578
41	0.9029177	0.6348441	0.9235441	0.0114891	0.0093646	0.0120730
51	0.9024135	0.6306494	0.9243332	0.0112254	0.0181792	0.0130011
61	0.9011316	0.6247846	0.9219676	0.0117500	0.0180436	0.0100948
71	0.9002778	0.6306494	0.9211798	0.0124986	0.0191180	0.0145822
81	0.8993970	0.6122218	0.9222314	0.0128678	0.0180488	0.0118454
91	0.8982945	0.6097078	0.9232816	0.0119274	0.0187544	0.0113019

Confusion matrix for randomForest model 1 on the test set.

```
$positive
[1] "Y"

$table
         Reference
Prediction    Y    N
        Y   454  152
        N   331 2320

$overall
       Accuracy          Kappa  AccuracyLower  AccuracyUpper    AccuracyNull
   8.517040e-01   5.604634e-01   8.390299e-01   8.637430e-01    7.589807e-01
AccuracyPValue  McnemarPValue
   4.486038e-39   5.528521e-16

$byClass
          Sensitivity          Specificity          Pos Pred Value
            0.5783439            0.9385113               0.7491749
       Neg Pred Value            Precision                  Recall
            0.8751415            0.7491749               0.5783439
                   F1           Prevalence          Detection Rate
            0.6527678            0.2410193               0.1393921
 Detection Prevalence    Balanced Accuracy
            0.1860608            0.7584276

$mode
[1] "sens_spec"

$dots
list()

attr(,"class")
[1] "confusionMatrix"
```

From the confusion matrix, we see that the accuracy is 0.852. Sensitivity (the True Positive prediction rate) is 0.578, much lower than the Specificity (True Negative prediction rate) of 0.939. This is due to the low prevalence of the positive class, as already mentioned. Our main performance statistic, Kappa is 0.56.

Overall, without any tuning, the model performance is pretty good; Sensitivity above 58% indicates that the model is doing fairly good in predicting the positive class. The Kappa value also indicates that the model is doing fairly good in distinguising between the positive and negative classes. The next models attempt to improve upon the first model by tuning some of the parameters.

RandomForest Model 2. Attempt to tune mtry paramaters. With this model, the same sample (5K observations from the training set) with the same cross validation were used in an attempt to tune mtry. From model 1, the best mtry was 21 . The idea was to expand the around the mtry obtained from model 1 and see if there is a better mtry. With this approach, the best cross validation ROC for model 2 is achieved with mtry = 20.

```
# model 2 - train an rf model with cross validation and with different values
# for best mtry
fnTrainRandomforest_Tune_mtry <- function(train_sample){
  train(
    over50K ~ .,
    method = "rf",
    data = train_sample,
    metric="ROC",
    trControl = cv_control,
    tuneGrid  = data.frame(mtry = c(5, 10, 15, 20, 30)),
    allowParallel=TRUE
  )
}
```

Table 13: Cross validation results - randomForest model 2 (finding best mtry)

mtry	ROC	Sens	Spec	ROCSD	SensSD	SpecSD
5	0.8983858	0.4790127	0.9537547	0.0113837	0.0407408	0.0077555
10	0.9037128	0.5970782	0.9356262	0.0131728	0.0623222	0.0098900
15	0.9060860	0.6280511	0.9293198	0.0137753	0.0676973	0.0096035
20	**0.9069139**	**0.6406139**	**0.9266913**	**0.0140655**	**0.0628479**	**0.0145438**
30	0.9051630	0.6297177	0.9235393	0.0132992	0.0672162	0.0132676

The following shows the cross validation results for model 2. Even though the best mtry from model 2 is different from the default one selected by caret in model 1, model 2 did not result in any improvement over model 1; ROC is still at 90% with both sensitivity and specificity remaining the same. In fact, the ROC curves for both models look identical.
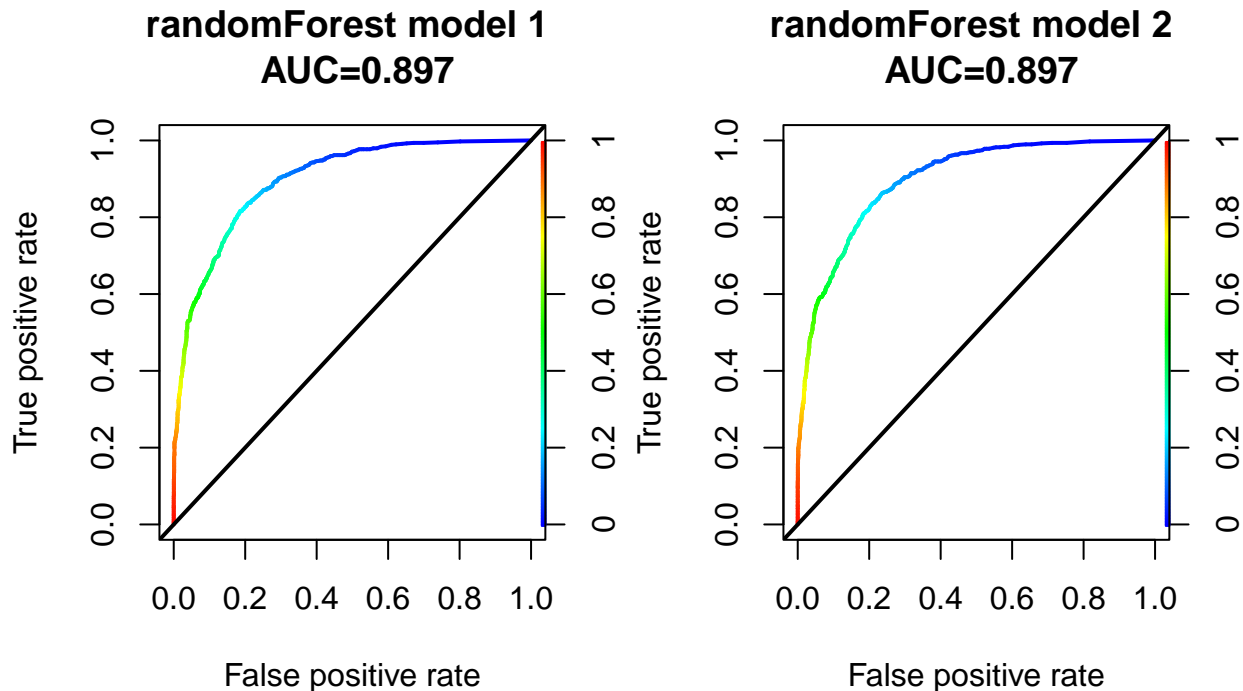
RandomForest Model 3. Attempt to tune randomForest nodesize parameter. The final two models involved attempting to tune randomforest mtry parameter. In model 1, caret selected the mtry; model 2 tuned the mtry from model 1.

In model 3, mtry from model 2 is used and held it constant to find the best nodesize. Different models were fitted across different values of nodesize and the one with best Kappa was selected.

```r
# given a fixed mtry value, train an rf model and with different values for nodesize
fnTrainRandomforest_Tune_nodesize <- function(train_sample, selected_mtry){
  nodesize <- c(seq(10, 100, 5),150,200)
  map_df(nodesize, function(n){
    rf_train <-
      train(
        over50K ~ .,
        method = "rf",
        data = train_sample,
        tuneGrid  = data.frame(mtry = c(selected_mtry)),
        allowParallel=TRUE, nodesize = n)
        list(nodesize=n,accuracy=rf_train$results$Accuracy, kappa=rf_train$results$Kappa)
  } )
}
```

The plot below shows the values of Kappa across different values for nodesize in model 3.
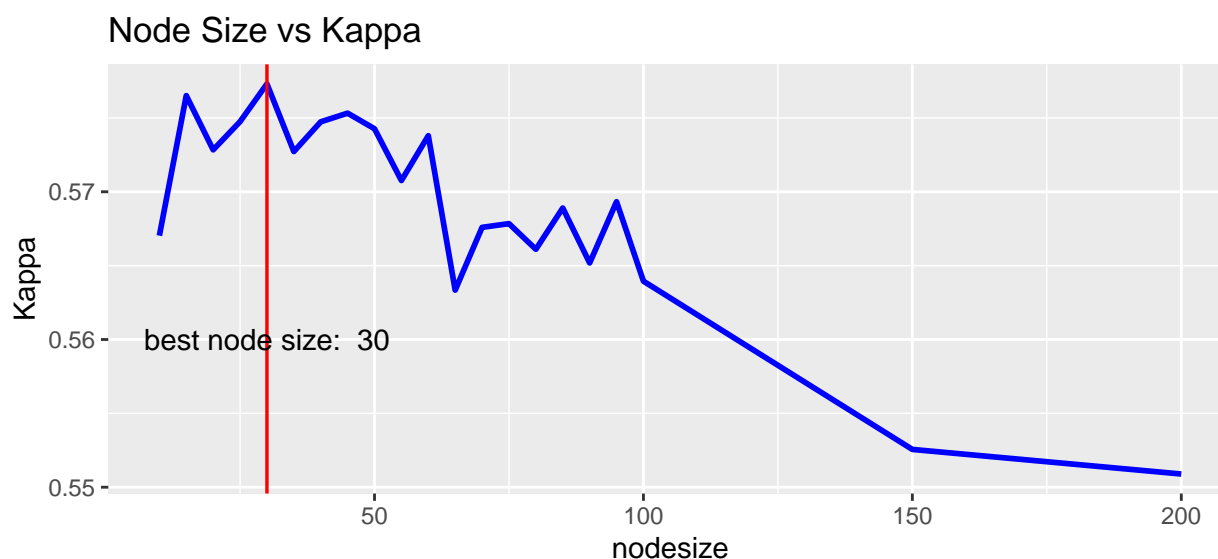
Models 3 suggests the number `30` as an ideal value for node size. The following table shows the performance statistics for all three models fitted on the test set, including the best model from knn.

Table 14: Model performance on Test Set

	test.accuracy	test.kappa	test.sensitivity	test.specificity
knn_best_model	0.8517040	0.5604634	0.5783439	0.9385113
rf_model.1 (using caret defaults)	0.8317470	0.4987134	0.5299363	0.9275890
rf_model.2 (find best mtry)	0.8532392	0.5656277	0.5834395	0.9389159
rf_model.3 (find best nodesize)	**0.8541603**	**0.5648721**	**0.5745223**	**0.9429612**

Final model and final prediction on validation set. From table above, randomForest model 3 (*rf_model*) is the best performing model. Even though the performance between kNN and the last two randomForest models are very close in terms of accuracy, it it clear that randomForest model 3 is a better model using Kappa as a metric.

However, the execution time for randomForest model 3 (rf_model_3) was over 2hours (more than the time it took for two kNN models plus the first two randomForest models). Yet, rf_model_3 only offered a small improvement over rf_model_2, Therefore, for the sake of simplicity, I selected randomForest model 2 (**rf_model_2**) as the final model to apply on the validation set. In the code, this corresponds to **fit_2** (the fitted model based on rf_model_2). The selected model used mtry of **15**.

```
predict_final <- predict(fit_2, adult_income_validation, type="class")
confusionMatrix_final  <- confusionMatrix(predict_final,
                                          adult_income_validation[, "over50K"])
```

```
$positive
[1] "Y"

$table
          Reference
Prediction    Y      N
         Y  2319    851
         N  1527  11583

$overall
      Accuracy            Kappa  AccuracyLower  AccuracyUpper    AccuracyNull
  8.539312e-01     5.690650e-01   8.484122e-01   8.593235e-01    7.637592e-01
AccuracyPValue  McnemarPValue
 1.087726e-179    1.422574e-43

$byClass
         Sensitivity           Specificity          Pos Pred Value
           0.6029641             0.9315586               0.7315457
        Neg Pred Value             Precision                  Recall
           0.8835240             0.7315457               0.6029641
                    F1            Prevalence          Detection Rate
           0.6610604             0.2362408               0.1424447
  Detection Prevalence     Balanced Accuracy
             0.1947174             0.7672614

$mode
[1] "sens_spec"

$dots
list()

attr(,"class")
[1] "confusionMatrix"
```

The final confusion matrix shows an Accuracy of almost **86%**, Kappa of `0.57`, and Sensitivity of `0.6`.

Important Variables. The randomForest model ranks the features (variables) used in the classification. The chart below shows the top 20 most important features in classifying whether a given income is >50K or not. From the chart, it can be seen that the capital_gain/loss variable is far the most important variable in the model. This is followed by education_num (number of years of education), and hour_per_week.

**Top 20 features in the final model
in the classification of individual's income**
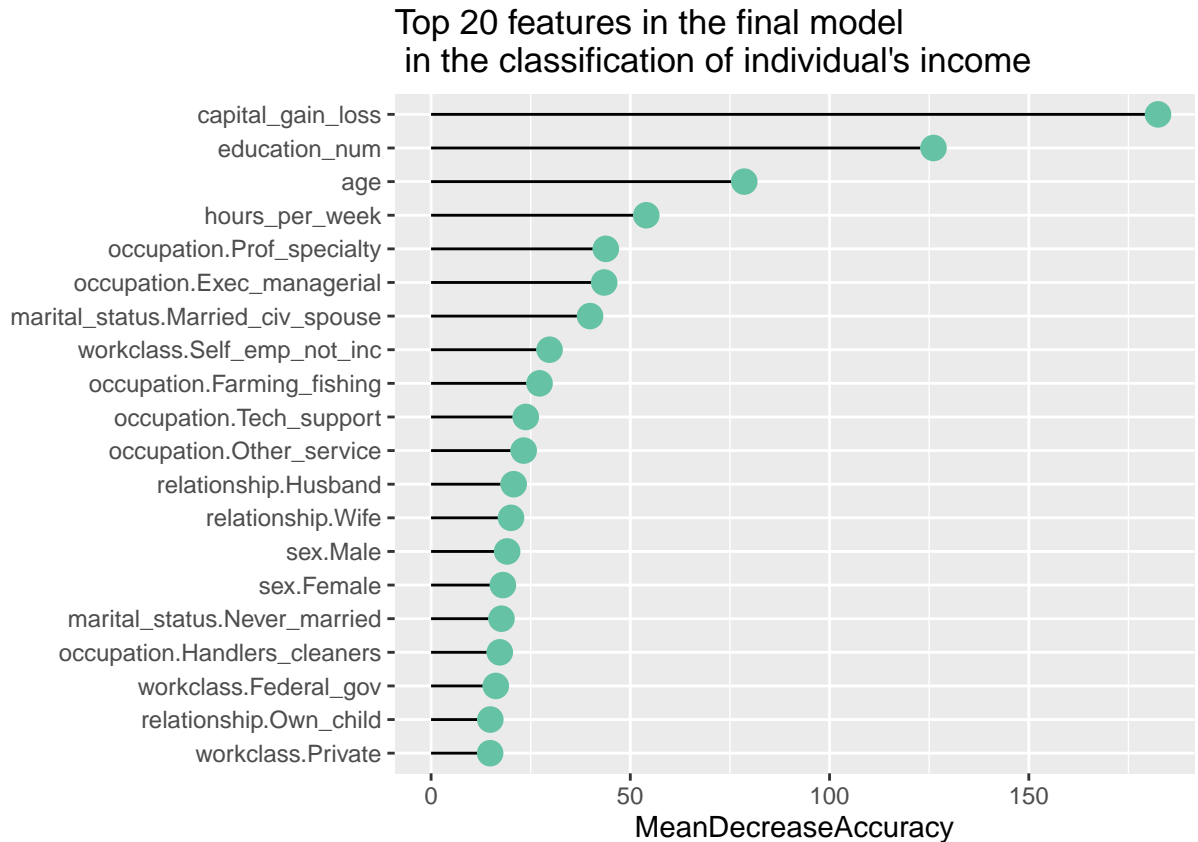
Conclusion

In this project, kNN and randomForest were used to predict whether an individual's income will be <=50K or >50K. Both algorithms performed very competitively, with randomForest winning based on Kappa performance. The final model on the validation set yielded an overall accuracy of 86%. Even with the low prevalence of the positive class, the model is able to correctly predict the positive class 60% of the time, which is pretty good. The model also showed that capital gain or loss is the most important variable in the classification of whether an individual's income is <=50K or >50K. The top four variables in the model for predicting the outcome are *capital_gain_loss*, *education_num*, *age*, and *hour_per_week*.

The project, for the sake of time, focused on two algorithms (kNN and randomForest). Although the final model's performance is fairly good, an improvement may have been achieved by considering more models and using ensemble methods. However, the skills acquired through this project is by far the most valuable to me.

Technical Notes

Source Code & Github Repository

The source code for the entire project can be found on github at:
https://github.com/kowusu01/KOwusu.Tieku.HarvardX.Capstone.CYO

Development Environment Used

OS: Windows 10 x64 (build 19041), 16GB RAM
Machine: Dell Latitude, i5 Dual Core @ 2.4GHz and 2.5GHz
R: version 4.1.2 (2021-11-01)

Executing the Scripts outside RStudio

Note:

- This project was developed and tested in Windows. It has not been tested under any other operating system.

- RStudio is not required to execute the script.

To execute the script without RStudio:

- create a root folder and name it e.g. *base*
- create a subfolder called *R* under the base folder and copy the main script (**adults_income.R**) to that folder
- create folders *data* and *rda* under the base folder
- open a command window with Administrator privileges (right-click Windows Command Prompt icon and select *Run As Administrator*)
  make sure you are pointing to the correct version of R (min version 4.1.2). Type *R –version* at the command prompt.
- change directory to the root folder you created
- issue the following command: **rscript R/adults_income.R**

REFERENCES

1. Irizarry, Rafael A., (2021), Introduction to Data Science Data Analysis and Prediction Algorithms with R, URL: https://rafalab.github.io/dsbook/

2. James, G., Witten, D., Hastie, T., Tibshirani, R., (2017), Introduction of Statistical Learning with Applications in R, "Springer Text in Statistics".

3. Nwanganga, F & Chapple, Mike, (2020), Practical Machine Learning in R, "Wiley".

4. Zhu, Hao. (2020), Create Awesome LaTeX Table with knitr::kable and kableExtra,
   URL: https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_pdf.pdf

5. U.S. Bureau of Labor Statistics, Standard Occupational Classification and Coding Structure, https://www.bls.gov/soc/2018/soc_2018_class_and_coding_structure.pdf

6. U.S. Bureau of Labor Statistics, Standard Occupational Classification Manual, https://www.bls.gov/soc/2018/soc_2018_manual.pdf

7. Wickham, H & Grolemund, G; R for Data Science - Visualize, Model, Transform, Tidy, and Import Data, https://r4ds.had.co.nz/

8. Zhu, Hao, (2021), Create Awesome HTML Table with knitr::kable and kableExtra, URL: https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_html.html