Adult Census Income

Yi Zheng

2021-01-08

1 Introduction

This project uses demographic data to predicts whether a person's annual income exceeds **$50K**. The data was extracted from the 1994 Census Bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics), and the data has been published in the UCI Machine Learning repository. The topic was selected from a list of datasets curated by Kaggle, which is recommended by the capstone "Choose Your Own" project overview.

The file was downloaded from https://www.kaggle.com/uciml/adult-census-income/download/ and has been relocated at https://github.com/yi628/Adult-Census-Income/raw/main/adult.csv for easy access via automatic download.

adult.csv is the dataset file, containing **32,561** rows, each row represents a person's census record. There are **15** columns in the dataset. **age** is a continuous feature, ranging from **17** to **90** years old. **workclass** is a categorical feature that determines the job class or category for the person (such as a government or private company).

fnlwge is a continuous feature that weights the person's demographic characteristics on the Current Population Survey (CPS). **education** is a categorical feature that represents the highest education credential granted by the person. **education.num** is a continuous feature shows the person's years of education.

marital.status, **occupation**, **relationship**, **race**, **sex**, and **native.country** are categorical features that literally determine the demographic characteristics of the person. **capital.gain** and **capital.loss** are continuous features of a person's financial situation. **hours.per.week** is a continuous feature that indicates the weekly working hours. **income** is the class label used to determine whether the annual income exceeds **$50K**.

The goal of the project is to use the demographic information provided (such as marital status, occupation, etc) to successfully predict whether the person has a higher income and find the optimal prediction model with the highest accuracy.

In order to achieve this goal, multiple data analysis steps need to be performed: first, the original dataset will be prepared, cleaned, and explored; then, visuals will be drawn to help us better understand the data and gain helpful insights; finally, different prediction models will be implemented, and the best model with the highest accuracy will be determined and reported.

2 Analysis

2.1 Data Preparation

First, We check whether the necessary libraries are installed, and then load them all.

```r
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(devtools))   install.packages("devtools", repos = "http://cran.us.r-project.org")
if(!require(catboost))   devtools::install_github("catboost/catboost", subdir = "catboost/R-package")

library(tidyverse)
library(caret)
library(data.table)
library(catboost)
```

Some options are applied to customize the printout, for example, ignore minor warnings and avoid scientific notation.

```r
options(dplyr.summarise.inform = FALSE)
options(scipen = 999)
```

The file is downloaded and loaded into the memory. All categorical features will be converted into factor columns in the dataset.

```r
dl <- tempfile()
download.file("https://github.com/yi628/Adult-Census-Income/raw/main/adult.csv", dl)

adult <- read.csv(dl, stringsAsFactors = TRUE)
rm(dl)
```

2.2 Data Cleaning

The structure of the **adult** set can be examined as follows. Apparently, there are **32561** rows and **15** columns, including the class label. The categorical features **workclass**, **occupation**, and **native.country** contain "?" levels and need to be modified later. Most **capital.gain** are zeros. The class label **income** has two values "<=50K" and ">50K", which will be modified to pure characters to simplify the prediction models.

```r
str(adult)
```

```
## 'data.frame':    32561 obs. of  15 variables:
##  $ age           : int  90 82 66 54 41 34 38 74 68 41 ...
##  $ workclass     : Factor w/ 9 levels "?","Federal-gov",..: 1 5 1 5 5 5 5 8 2 5 ...
##  $ fnlwgt        : int  77053 132870 186061 140359 264663 216864 150601 88638 422013 70037 ...
##  $ education     : Factor w/ 16 levels "10th","11th",..: 12 12 16 6 16 12 1 11 12 16 ...
##  $ education.num : int  9 9 10 4 10 9 6 16 9 10 ...
##  $ marital.status: Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 7 7 7 1 6 1 6 5 1 5 ...
##  $ occupation    : Factor w/ 15 levels "?","Adm-clerical",..: 1 5 1 8 11 9 2 11 11 4 ...
##  $ relationship  : Factor w/ 6 levels "Husband","Not-in-family",..: 2 2 5 5 4 5 5 3 2 5 ...
##  $ race          : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 3 5 5 5 5 5 5 5 ...
##  $ sex           : Factor w/ 2 levels "Female","Male": 1 1 1 1 1 1 2 1 1 2 ...
##  $ capital.gain  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ capital.loss  : int  4356 4356 4356 3900 3900 3770 3770 3683 3683 3004 ...
##  $ hours.per.week: int  40 18 40 40 40 45 40 20 40 60 ...
##  $ native.country: Factor w/ 42 levels "?","Cambodia",..: 40 40 40 40 40 40 40 40 40 1 ...
##  $ income        : Factor w/ 2 levels "<=50K",">50K": 1 1 1 1 1 1 1 2 1 2 ...
```

Check if the **adult** set contains any **NULL** values.

After calling the sum of the **colSums** function to calculate the number of **NULL** values, we can determine that the result is **0**.

```
sum(colSums(is.na(adult)))
```

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

Check how many "?" are in the **adult** set. Thus, from the result below, we can find that the dataset contains **4262** "?" values.

```
length(which(adult == "?"))
```

```
## [1] 4262
```

There are **1836** records in the **workclass** column are missing. We determine the factor levels from the working classes, and change the "?" to "Unknown".

Hence, the result is shown by calling the **table** function.

```
level_list <- levels(adult$workclass)
level_list[1] <- "Unknown"
levels(adult$workclass) <- level_list
table(adult$workclass)
```

```
##
##          Unknown      Federal-gov        Local-gov     Never-worked
##             1836              960             2093                7
##          Private     Self-emp-inc Self-emp-not-inc        State-gov
##            22696             1116             2541             1298
##      Without-pay
##               14
```

There are **1843** missing records in the **occupation** column. We determine the factor levels from the occupations, and then change the "?" to "Unknown".

Thus, the result can be displayed by calling the **table** function.

```
level_list <- levels(adult$occupation)
level_list[1] <- "Unknown"
levels(adult$occupation) <- level_list
table(adult$occupation)
```

```
##
##           Unknown      Adm-clerical      Armed-Forces      Craft-repair
##              1843              3770                 9              4099
##   Exec-managerial   Farming-fishing Handlers-cleaners Machine-op-inspct
##              4066               994              1370              2002
##     Other-service   Priv-house-serv    Prof-specialty   Protective-serv
##              3295               149              4140               649
##             Sales      Tech-support  Transport-moving
##              3650               928              1597
```

There are **583** records in the **native.country** column are missing. We determine the factor levels from the native countries, and then change the "?" to "Unknown".

Therefore, the result is displayed by calling the **table** function. The first 28 native countries including unknown records are shown as follows.

```
level_list <- levels(adult$native.country)
level_list[1] <- "Unknown"
levels(adult$native.country) <- level_list
table(adult$native.country)[1:28]
```

```
##
##            Unknown           Cambodia             Canada              China
##                583                 19                121                 75
##           Columbia               Cuba Dominican-Republic            Ecuador
##                 59                 95                 70                 28
##        El-Salvador            England             France            Germany
##                106                 90                 29                137
##             Greece          Guatemala              Haiti Holand-Netherlands
##                 29                 64                 44                  1
##           Honduras               Hong            Hungary              India
##                 13                 20                 13                100
##               Iran            Ireland              Italy            Jamaica
##                 43                 24                 73                 81
##              Japan               Laos             Mexico          Nicaragua
##                 62                 18                643                 34
```

After completing the above modifications, check the number of "?" again in the **adult** set.

Hence, from the result below, we can find that the number of "?" has been reduced to **0**, such that there is no "?" in the **adult** set.

```
length(which(adult == "?"))
```

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

To simplify the prediction models, we change the class label ">50K" to "Yes" and "<=50K" to "No". Calculat the modified labels and number as shown below. Therefore, there are **7841** "Yes" and **24720** "No".

```
levels(adult$income) <- list("Yes" = ">50K", "No" = "<=50K")
table(adult$income)
```

```
##
##   Yes    No
##  7841 24720
```

```
rm(level_list)
```

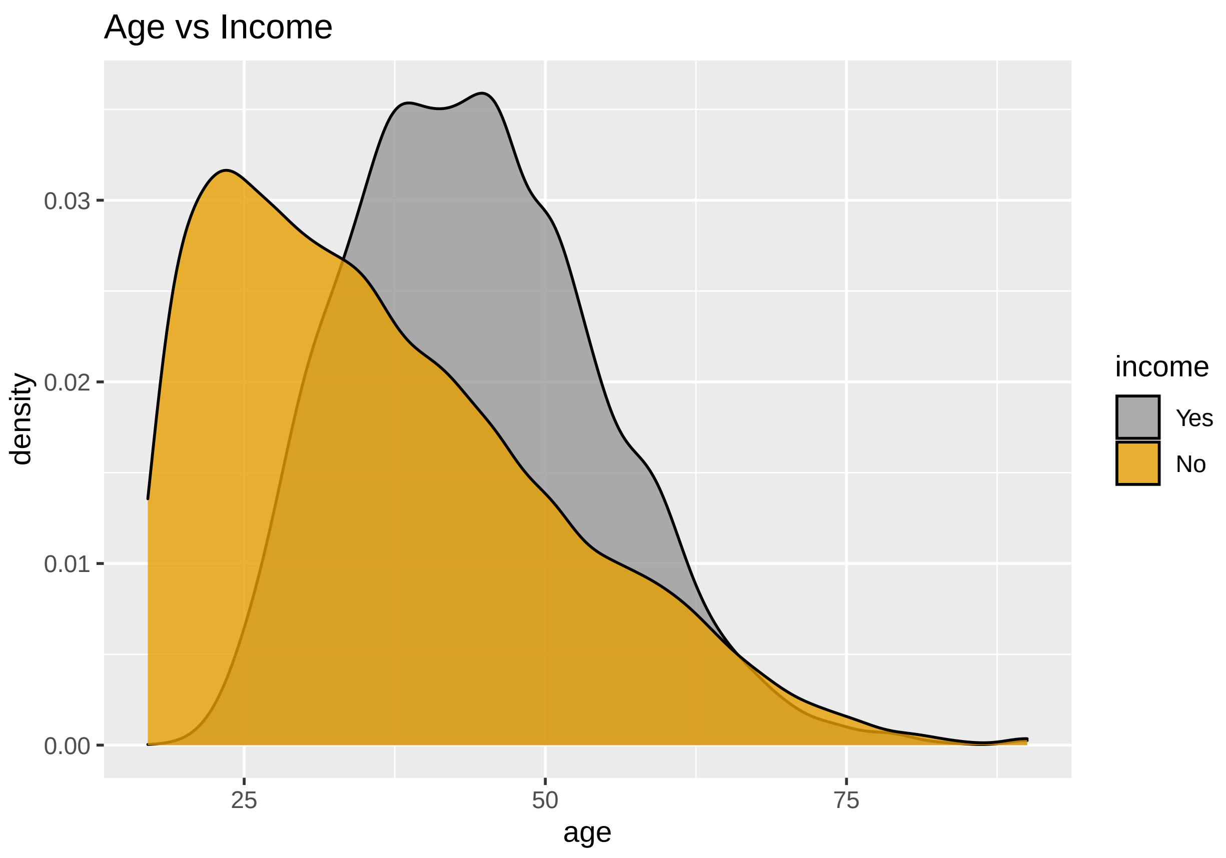
2.3 Data Exploration

2.3.1 Age vs Income

age is a continuous feature, ranging from **17** to **90** years old. The following shows the age distribution of different income groups. Orange represents people whose annual income is less than **$50K**, and gray illustrates people whose income exceeds **$50K**.

The age distribution with annual income below **$50K** is skewed to the right, and most people are around **25**. This is absolutely in line with the fact that young people have low incomes. In addition, the age distribution of incomes over **$50K** seems to be bell-shaped. In this way, it is normally distributed, which is another fact that applies to everyone: making big money is not a miracle.

```
adult %>%
  ggplot(aes(x = age, fill = income)) +
  geom_density(alpha = 0.8) +
  scale_fill_manual(values = c("#999999", "#E69F00")) +
  ggtitle("Age vs Income")
```

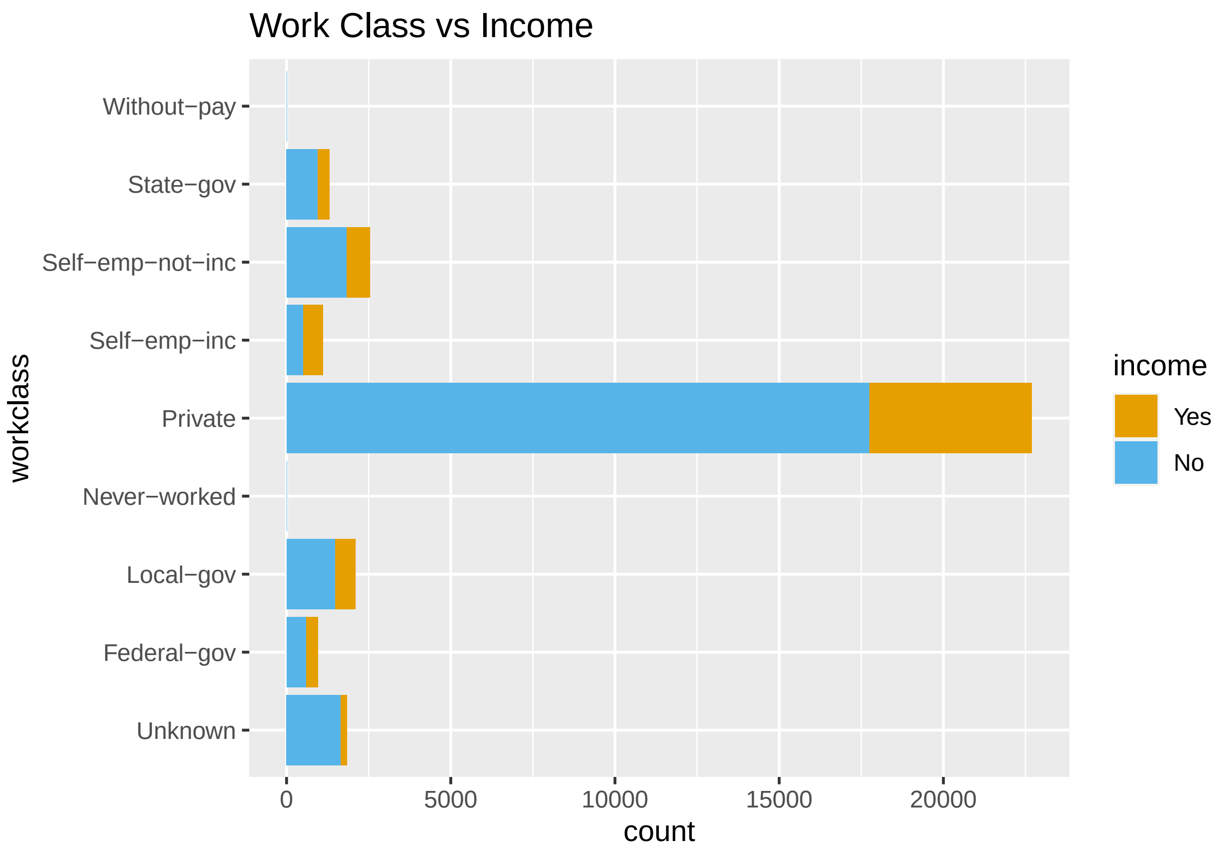


2.3.2 Work Class vs Income

workclass is a categorical feature that determines the job class or category for the person (such as a government or private company). The stacked bar chart for each job category is shown below. The number of people whose income exceeds **$50K** is stacked in orange, while the number of people whose income is lower than **$50K** is colored in blue.

Obviously, most people work in private companies in the **adult** set. However, the proportion of high-income earners does not seem to have any regularity. Hence, the working class will not affect people's income.

```
adult %>%
  group_by(workclass, income) %>%
  summarize(count = n()) %>%
  ggplot(aes(y = workclass, x = count, fill = income)) +
  geom_bar( position = "stack", stat = "identity") +
  scale_fill_manual(values = c("#E69F00", "#56B4E9")) +
  ggtitle("Work Class vs Income")
```

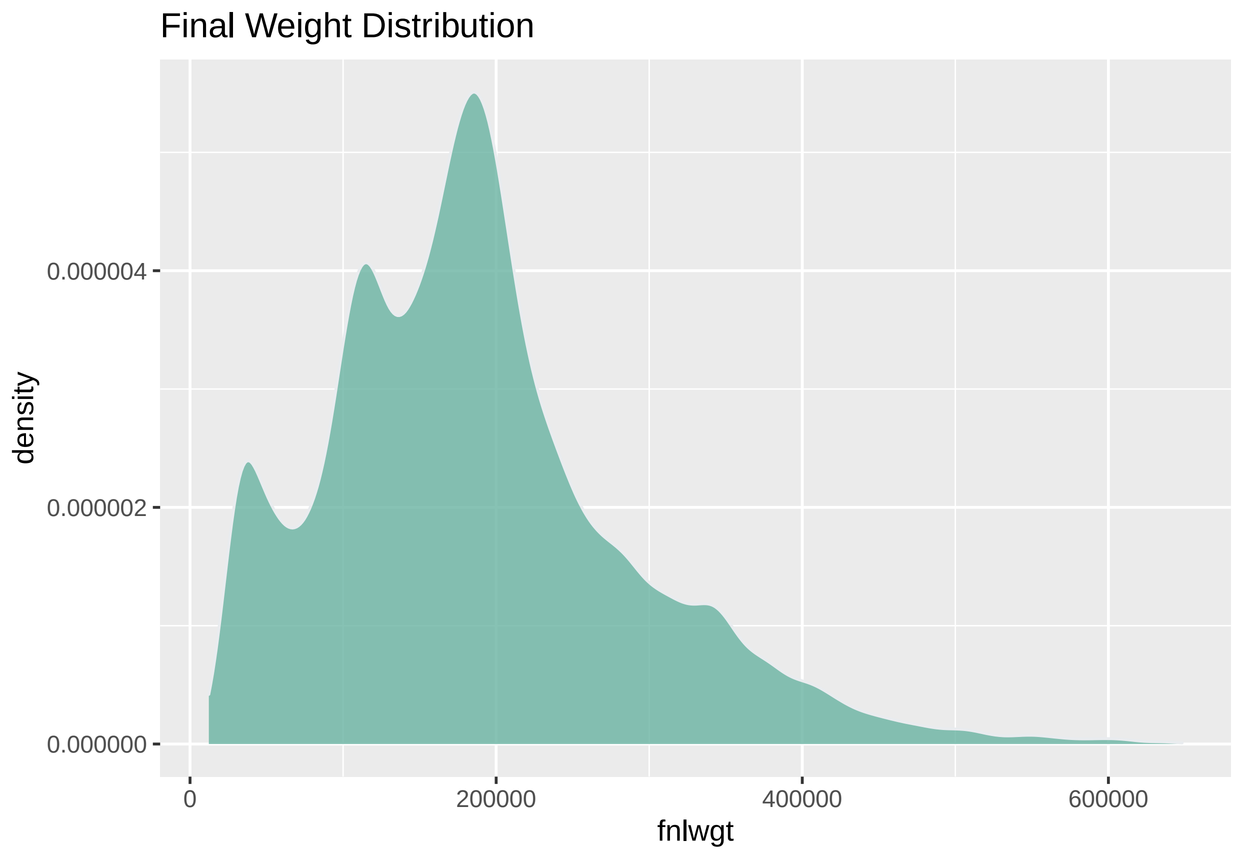


2.3.3 Final Weight Distribution

fnlwge is a continuous feature that weights the demographic characteristics of the person on the Current Population Survey (CPS). The final weights distribution is shown below, and the distribution is skewed to the right. Also, most weights are around **200,000**.

We limit the number to less than **650,000** to omit some outliers and make the graph easy to view.

```
adult %>%
  filter(fnlwgt < 650000) %>%
  ggplot(aes(x = fnlwgt)) +
  geom_density(fill = "#69B3A2", color = "#E9ECEF", alpha = 0.8) +
  ggtitle("Final Weight Distribution")
```

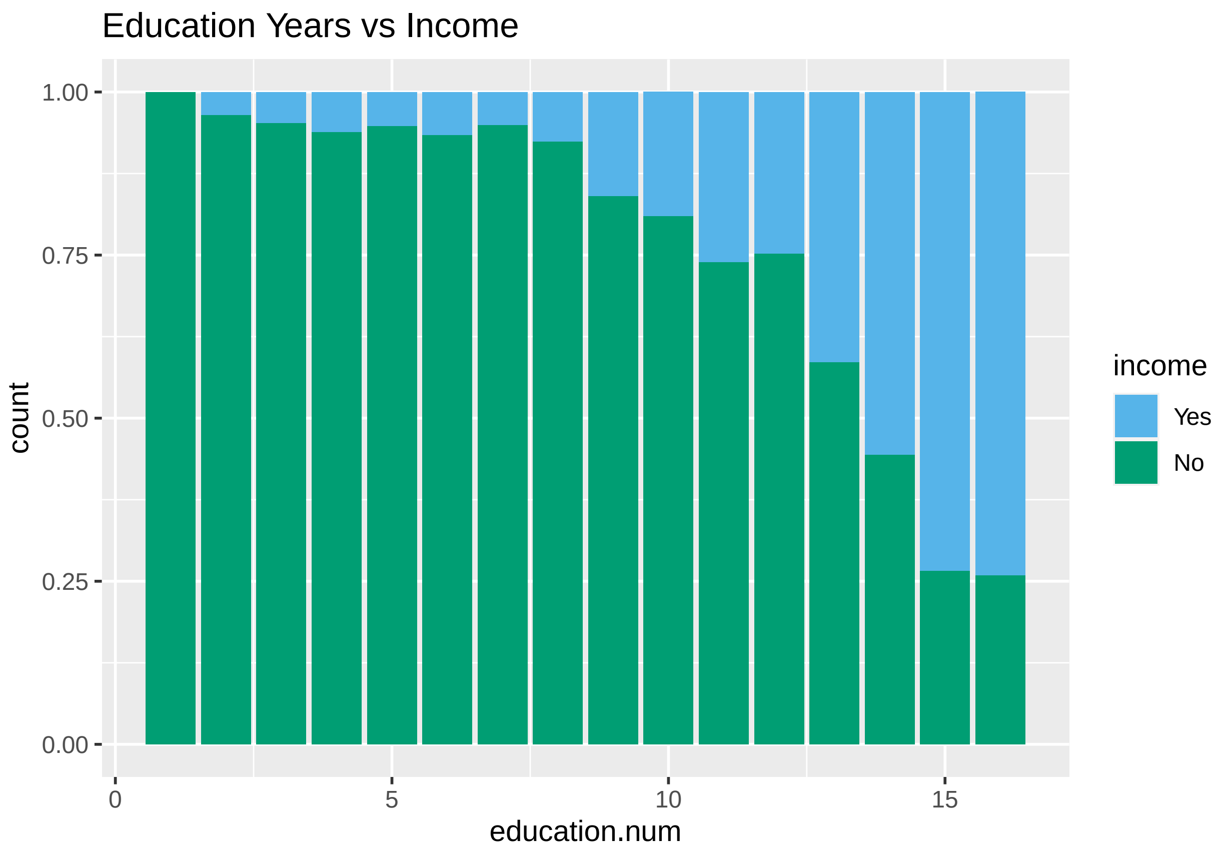


2.3.4 Education Years vs Income

education.num is a continuous feature that shows the person's years of education. The figure below shows the distribution of the proportion of people in different years of education whether their income exceeds \50K. Green illustrates people whose annual income is less than **$50K**, and blue represents people who make over **$50K** per year.

Apparently, this high-income pattern is easy to determine. As people's years of education increase, the proportion of high-income groups will also increase.

```r
adult %>%
  group_by(education.num, income) %>%
  summarize(count = n()) %>%
  ggplot(aes(x = education.num, y = count, fill = income)) +
  geom_bar(position = "fill", stat = "identity") +
  scale_fill_manual(values = c("#56B4E9", "#009E73")) +
  ggtitle("Education Years vs Income")
```



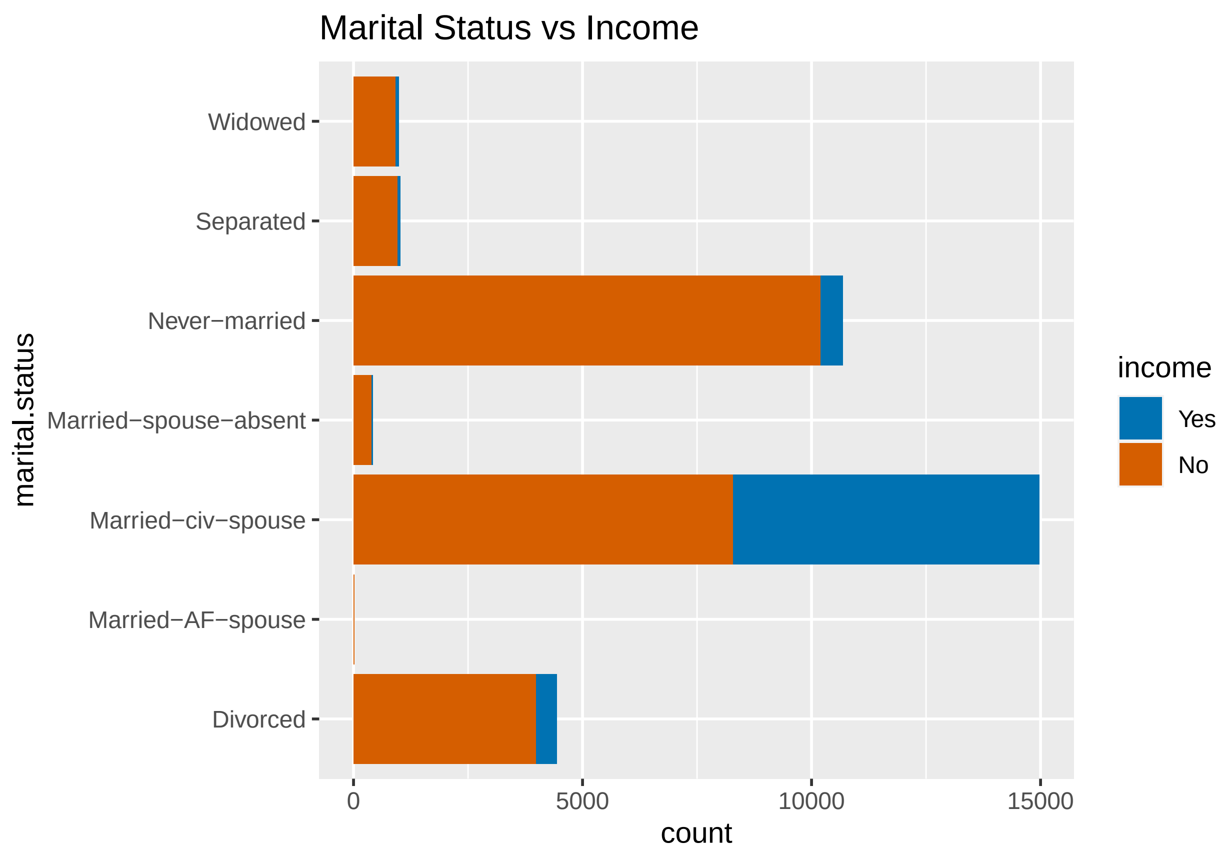
2.3.5 Marital Status vs Income

marital.status is a categorical feature that literally determines the marital status of the person. The stacked bar chart for each marital status is shown below. **Married-civ-spouse** corresponds to married to a civilian, and **Married-AF-spouse** means a spouse in the armed forces.

The number of people whose income exceeds **$50K** is stacked in blue, and the number of people whose income does not exceed **$50K** is colored in orange.

As can be seen from the chart below, most people marry civilian spouses in the **adult** set. The proportion of high-income earners is higher than people with other marital status. Therefore, marital status does affect people's income.

```r
adult %>%
  group_by(marital.status, income) %>%
  summarize(count = n()) %>%
  ggplot(aes(y = marital.status, x = count, fill = income)) +
  geom_bar( position = "stack", stat = "identity") +
  scale_fill_manual(values = c("#0072B2", "#D55E00")) +
  ggtitle("Marital Status vs Income")
```

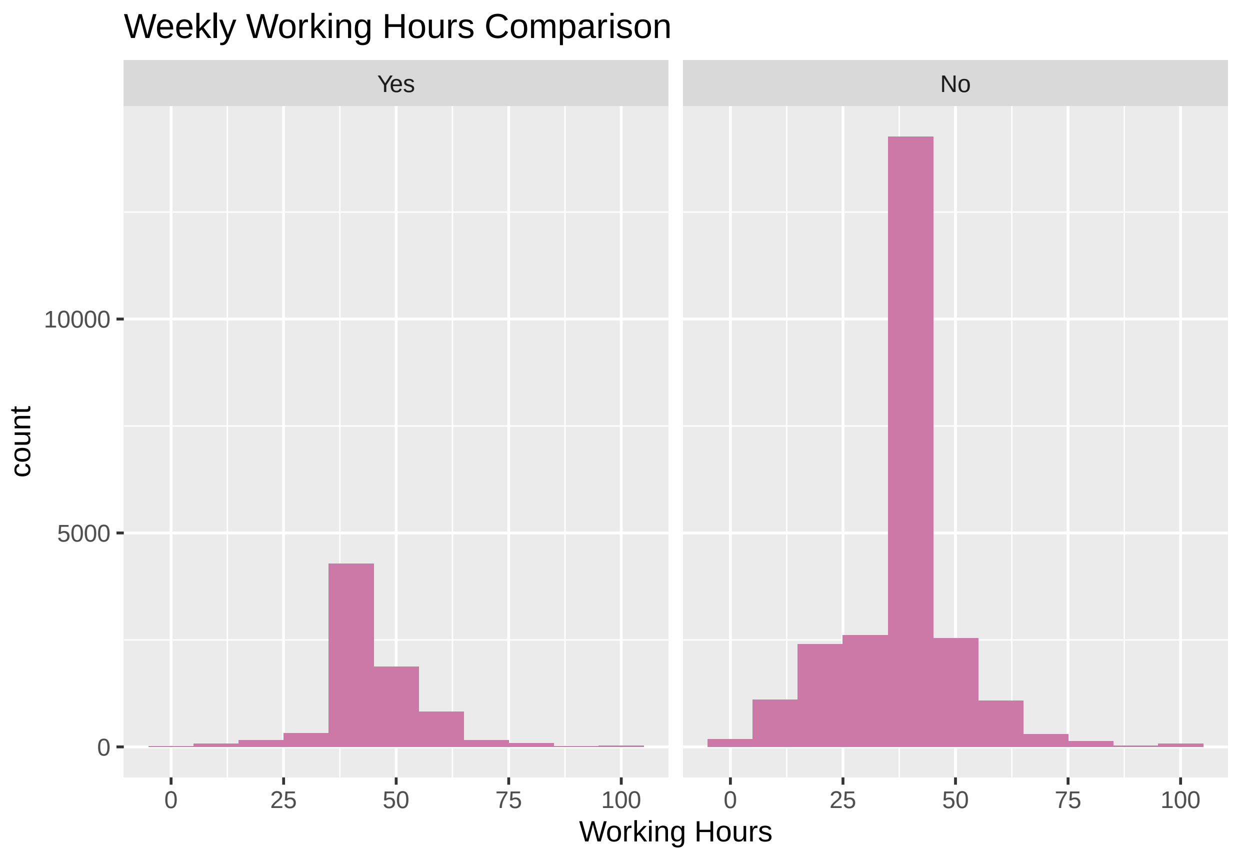


2.3.6 Weekly Working Hours Comparison

hours.per.week is a continuous feature that indicates the weekly working hours. Below is two histograms of working hours per week under different income levels. **binwidth** has been set to **10**, so the working time has been grouped into **0** to **10**, **10** to **20**, etc.

The distribution seems to be the same, but there are some subtle differences, for example, there are more people who work **40** to **50** hours a week but earn less than **$50K** per year.

```r
adult %>%
  ggplot(aes(x = hours.per.week)) +
  geom_histogram(fill = "#CC79A7", binwidth = 10) +
  facet_wrap(~income) +
  xlab("Working Hours") +
  ggtitle("Weekly Working Hours Comparison")
```



2.4 Insights gained

In the **adult** set, the **age** feature is a good indicator of class label, because the distribution is consistent with common sense. The working class does not affect people's income, so the **workclass** feature is less important for income prediction. People's years of education and are highly correlated with income, so **education.num** is another good predictor of class label. The distribution of **hours.per.week** is slightly different and can be used as a feature of class label.

2.5 Modeling Approach

In order to evaluate the performance of the project, the **adult** set is divided, **90%** of the data is included in the training set, and **10%** of the data is included in the testing set. Thus, **train_set** contains **29304** rows, and **test_set** contains **3257** records.

3 different models will be implemented for analysis and comparison. Each model will be built on the training set and predict whether the income in the testing set will exceed **$50K** per year .

```
set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(adult$income,
                                  p = 0.1,
                                  list = FALSE)
train_set  <- adult[-test_index,]
test_set   <- adult[test_index,]

rm(test_index)
```

Prepare the training scheme, use 10-fold Cross-Validation as the resampling method, and keep the default 75% as the percentage of the training set.

```
control <- trainControl(method = "cv")
```

2.5.1 Random Forest

We implement the Random Forest algorithm firstly. We set **ntree** to **20**, so that **20** branches will grow after each time split. Accuracy has been defined as the comparison metric for the best model.

We take advantage of the **caret** package and tune the parameter **mtry**, which is to randomly collect the number of features to be sampled at each split time. We tuned **mtry** from **1** to **14** because there are **14** features in the **adult** set.

After the training process is completed, the best **mtry** = **14** is reported. We keep constant value **14** for **mtry** in the tune grid, and rerun the code to shorten the report generation time.

The results of the Random Forest model are shown below.

```
#grid <- expand.grid(mtry = c(1:14))
grid <- expand.grid(mtry = 14)

model_rf <- train(income~.,
                  data = train_set,
                  method = "rf",
                  ntree = 20,
                  metric = "Accuracy",
                  trControl = control,
                  tuneGrid = grid)

print(model_rf)
```

```
## Random Forest
##
## 29304 samples
##    14 predictor
##     2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 26373, 26374, 26374, 26374, 26375, 26373, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8621689  0.6038019
##
## Tuning parameter 'mtry' was held constant at a value of 14
```

2.5.2 CatBoost

Secondly, we perform the CatBoost algorithm. We set the number of trees **iterations** to **100**, and the logging period **verbose** to **0** so that logging is disabled. Accuracy has been defined as the comparison metric for the optimal model.

We tune the learning rate **learning_rate** from **0.03** to **0.99**, the tree depth **depth** from **6** to **10**, the L2 regularization coefficient **l2_leaf_reg** from **0.04** to **1.00**, the percentage of features to use at each iteration **rsm** from **0.05** to **1.00**, and the number of splits for numerical features **border_count** from **10** to **100**.

After the training process is completed, the best **learning_rate** = **0.75**, **depth** = **9**, **l2_leaf_reg** = **0.88**, **rsm** = **0.95**, and **border_count** = **40** are reported. We keep the optimal values in the tune grid, and rerun the code to shorten the report generation time. The original tuning process took several hours to complete.

The results of the CatBoost model are shown below.

```
#grid <- expand.grid(iterations    = 100,
#                     learning_rate = seq(from = 0.03, to = 0.99, by = 0.03),
#                     depth         = seq(from = 6,    to = 10,   by = 1   ),
#                     l2_leaf_reg   = seq(from = 0.04, to = 1.00, by = 0.04),
#                     rsm           = seq(from = 0.05, to = 1.00, by = 0.05),
#                     border_count  = seq(from = 10,   to = 100,  by = 10  ))
grid <- expand.grid(iterations = 100,
                    learning_rate = 0.75,
                    depth = 9,
                    l2_leaf_reg = 0.88,
                    rsm = 0.95,
                    border_count = 40)

model_ctb <- train(x = train_set[, -ncol(train_set)],
                   y = train_set$income,
                   method = catboost.caret,
                   eval_metric = "Accuracy",
                   tuneGrid = grid,
```

```r
                   verbose = 0)

# Review the tuning result
print(model_ctb)
```

```
## Catboost
##
## 29304 samples
##    14 predictor
##     2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 29304, 29304, 29304, 29304, 29304, 29304, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8551992  0.5886049
##
## Tuning parameter 'depth' was held constant at a value of 9
## Tuning
##
## Tuning parameter 'rsm' was held constant at a value of 0.95
## Tuning
##  parameter 'border_count' was held constant at a value of 40
```

2.5.3 XGBoost

Finally, we execute the XGBoost algorithm. We set the boosting iterations **nrounds** to **100**. Accuracy has been defined as the comparison metric for the best model.

We tune the parameter max tree depth (model complexity) **max_depth** from **3** to **10**, the learning rate **eta** from **0.1** to **0.3**, the minimum loss reduction **gamma** with **0** and **1**, the subsample ration of columns **colsample_bytree** from **0.5** to **1.0**, the minimum sum of instance weight **min_child_weight** with **1** and **2**, and the ratio of the training instances **subsample** from **0.5** to **1.0**.

After the training process is completed, the best **max_depth** = **6**, **eta** = **0.1**, **gamma** = **0**, **colsample_bytree** = **0.5**, **min_child_weight** = **1**, and **subsample** = **1** are reported. We keep the optimal values in the tune grid, and then rerun the code to shorten the report generation time. The original tuning process took days to complete.

The results of the XGBoost model are shown below.

```r
#grid <- expand.grid(nrounds          = 100,
#                    max_depth        = seq(from = 3,   to = 10,  by = 1  ),
#                    eta              = seq(from = 0.1, to = 0.3, by = 0.1),
#                    gamma            = seq(from = 0,   to = 1,   by = 1  ),
#                    colsample_bytree = seq(from = 0.5, to = 1.0, by = 0.1),
#                    min_child_weight = seq(from = 1,   to = 2,   by = 1  ),
#                    subsample        = seq(from = 0.5, to = 1.0, by = 0.1))
grid <- expand.grid(nrounds = 100,
                    max_depth = 6,
                    eta = 0.1,
                    gamma = 0,
                     colsample_bytree = 0.5,
                     min_child_weight = 1,
                     subsample = 1)

model_xgb <- train(income~.,
                   data = train_set,
                   method = "xgbTree",
                   metric = "Accuracy",
                   trControl = control,
                   tuneGrid = grid)

print(model_xgb)
```

```
## eXtreme Gradient Boosting
##
## 29304 samples
##    14 predictor
##     2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 26374, 26373, 26373, 26373, 26374, 26374, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8727475  0.6285957
##
## Tuning parameter 'nrounds' was held constant at a value of 100
## Tuning
##  held constant at a value of 1
## Tuning parameter 'subsample' was held
##  constant at a value of 1
```

3 Results

Using the models created in the previous section, we can use the **predict** function in the **caret** package to predict the class labels in the **test__set**. Then, we extract the accuracy rates from the results of the function **confusionMatrix** and report them below.

```
predicted <- predict(model_rf, test_set[, -ncol(test_set)], type = "raw")
accuracy_rf <- confusionMatrix(predicted, test_set$income)$overall["Accuracy"]
names(accuracy_rf) <- "Random Forest Accuracy"

predicted <- predict(model_ctb, test_set[, -ncol(test_set)], type = "raw")
accuracy_ctb <- confusionMatrix(predicted, test_set$income)$overall["Accuracy"]
names(accuracy_ctb) <- "CatBoost Accuracy"

predicted <- predict(model_xgb, test_set[, -ncol(test_set)], type = "raw")
accuracy_xgb <- confusionMatrix(predicted, test_set$income)$overall["Accuracy"]
names(accuracy_xgb) <- "XGBoost Accuracy"

c(accuracy_rf, accuracy_ctb, accuracy_xgb)
```

```
## Random Forest Accuracy      CatBoost Accuracy       XGBoost Accuracy
##              0.8606079              0.8633712              0.8722751
```

Obviously, the accuracy rate gradually increased from **0.8606** to **0.8634**, and the highest accuracy rate produced by the third model with XGBoost was **0.8723**. Furthermore, during the tuning and training process, the XGBoost model runs faster than the other two models. We found the best model with the highest accuracy and achieved the goal of the project.

4 Conclusion

4.1 Project Summary

This project illustrates the complete process of data science analysis and achieved the set goals. We acquired and preprocessed adult income data for analysis; explored and visualized them to better understand the census data, and obtained useful insights that can be used in future model construction.

Then, we tried **3** different models and gradually improved the accuracy. In the end, the accuracy reached **0.8723**, and we determined that the XGBoost model was the best.

4.2 Limitations

The original data file was downloaded from the existing data repository, and the data had been cleaned and preprocessed. Therefore, this project is only applicable to this version of the 15-column adult income data and does not support any other version of the census data. If the census data contains any other features, this project will not produce correct results.

4.3 Future Work

The raw adult income data from the Census Bureau database contains more demographic data for each person. With more features, we can train the model and predict better accuracy.

5 Reference

HarvardX PH125.8x Data Science: Machine Learning

https://courses.edx.org/courses/course-v1:HarvardX+PH125.8x+2T2020/course/

UCI Machine Learning Repository: Adult Data Set

https://archive.ics.uci.edu/ml/datasets/adult

Kaggle Adult Census Income

https://www.kaggle.com/uciml/adult-census-income