Census Pay Predictions

Rajesh Haridas

03 December, 2021

Preface

This document and the associated code is based on the Capstone HarvardX Professional Certificate in Data Science (PH125.9x) course work, and additional reading material provided in the course.

The following files (3 types) are included in the upload:

- **CensusPay.rmd -** Markdown for the main summary report file
- **CensusPay.R -** Main R code for the project
- **DatasetProcessingCode.R -** The R code for downloading, cleaning and converting the dataset into tidy form to be used in the project
- **CensusPaySummaryReport.pdf -** Main summary report containing analysis
- **CensusPayExecutionReport.pdf -** Main execution report containing output of the runs

This document does not describe various machine learning terminologies in detail. The appendix has more information on how to get to that information and more.

The code in this project is memory and CPU intensive. The minimum requirements are at least 8 CPU core, 64 GB RAM, and 1 TB of disk space. Anything lower will take excessively long time.

Contents

List of Figures

Introduction to Census income level predictions

The Current Population Survey (CPS), sponsored jointly by the U.S. Census Bureau and the U.S. Bureau of Labor Statistics (BLS), is the primary source of labor force statistics for the population of the United States. The adult census income data was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics).

This machine learning project uses the adult census income data to predict annual money incomes for adults, given a set of 13 employment and demographic attributes. Census money income is defined as income received on a regular basis before payments for personal income taxes, social security, union dues, Medicare deductions, etc. The income levels were categorized into two classes – more than $50,000, and less than or equal to $50,000.

More information can be found at https://www.kaggle.com/uciml/adult-census-income.

In this project for the Capstone course of the HarvardX Professional Certificate in Data Science (PH125.9x), we will explore the adult census income data set. The objective will be to analyze and model based on the classification of incomes and develop a machine-learning model by creating, training and test sets to predict income levels on a validation set with accuracy of more than 80% and a reasonable sensitivity, and specificity

Here is a glimpse of adultpay dataset. The columns that are of interest are income, age, sex, education.num, education, race, marital.status, relationship, workclass, hours.per.week and occupation. There are 32,561 rows and 13 columns

```
library(tidyverse)
library(caret)
# inspect the out-of-box original dataset
glimpse(adultpay)
```

```
## Rows: 32,561
## Columns: 15
## $ age            <int> 90, 82, 66, 54, 41, 34, 38, 74, 68, 41, 45, 38, 52, 32,~
## $ workclass      <chr> "?", "Private", "?", "Private", "Private", "Private", "~
## $ fnlwgt         <int> 77053, 132870, 186061, 140359, 264663, 216864, 150601, ~
## $ education      <chr> "HS-grad", "HS-grad", "Some-college", "7th-8th", "Some-~
## $ education.num  <int> 9, 9, 10, 4, 10, 9, 6, 16, 9, 10, 16, 15, 13, 14, 16, 1~
## $ marital.status <chr> "Widowed", "Widowed", "Widowed", "Divorced", "Separated~
## $ occupation     <chr> "?", "Exec-managerial", "?", "Machine-op-inspct", "Prof~
## $ relationship   <chr> "Not-in-family", "Not-in-family", "Unmarried", "Unmarri~
## $ race           <chr> "White", "White", "Black", "White", "White", "White", "~
## $ sex            <chr> "Female", "Female", "Female", "Female", "Female", "Fema~
## $ capital.gain   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ capital.loss   <int> 4356, 4356, 4356, 3900, 3900, 3770, 3770, 3683, 3683, 3~
## $ hours.per.week <int> 40, 18, 40, 40, 40, 45, 40, 20, 40, 60, 35, 45, 20, 55,~
## $ native.country <chr> "United-States", "United-States", "United-States", "Uni~
## $ income         <chr> "<=50K", "<=50K", "<=50K", "<=50K", "<=50K", "<=50K", "~
```

We will start with data exploration and cleaning unwanted data and processing. Then we will mimic the ultimate evaluation process by splitting the data into two parts - training and validation and act as if we don't know the outcome for the validation set. We will split the dataset once we cleaned and processed the dataset. We want the test set to be large enough so that we obtain a stable prediction without fitting an impractical number of models. We will then progressively apply various algorithms on the training set and test the predictions on the test set and improve on the overall accuracy.

We will choose 13 attributes, income being the outcome and the rest 12 attributes being the predictors or features. After perusing the dimensions, we will randomly choose 10% of the dataset to be the test set and remaining 90% will be the training set. 90% is sizable enough to run various algorithms like lm, glm,nb, knn, rpart, and random forest with some tuning.

```
library(tidyverse)
library(gridExtra)
library(kableExtra)
# inspect dimensions for the cleaned, training and validation datasets
dim(adultpayclean)
```

```
## [1] 29170    13
```

```
dim(adultpayclean_train)
```

```
## [1] 26252    13
```

```
dim(adultpayclean_validation)
```

```
## [1] 2918   13
```

```
# check for NAs and verify there aren't any in the cleaned dataset
colSums(is.na(adultpayclean))
```

```
##           age        fnlwgt     education      eduyears maritalstatus
```

```
##            0            0            0            0            0
##   occupation relationship         race          sex hoursperweek
##            0            0            0            0            0
##       native       income        class
##            0            0            0
```

This is a classification problem and the outcome is binary (income above-50K or at-or-below-50K). On reviewing the summary of the dataset we have a choice of predictors. We will keep only USA specific data. Almost all of the predictors have data spread out. There are no NAs in the dataset. There is sufficient degree of variability.

```
# statistics for the clean dataset
summary(adultpayclean)
```

```
##       age            fnlwgt              education        eduyears
##  Min.   :17.00   Min.   :  12285   HSgrad     :9702   Min.   : 1.00
##  1st Qu.:28.00   1st Qu.: 115895   Somecollege:6740   1st Qu.: 9.00
##  Median :37.00   Median : 176730   Bachelors  :4766   Median :10.00
##  Mean   :38.66   Mean   : 187069   Masters    :1527   Mean   :10.17
##  3rd Qu.:48.00   3rd Qu.: 234139   Assocvoc   :1289   3rd Qu.:12.00
##  Max.   :90.00   Max.   :1484705   11th       :1067   Max.   :16.00
##                                    (Other)    :4079
##              maritalstatus            occupation         relationship
##  Divorced            : 4162   Execmanagerial:3735   Husband      :11861
##  MarriedAFspouse     :   23   Profspecialty :3693   Notinfamily  : 7528
##  Marriedcivspouse    :13368   Craftrepair   :3685   Otherrelative:  696
##  Marriedspouseabsent:  253   Admclerical   :3449   Ownchild     : 4691
##  Nevermarried        : 9579   Sales         :3364   Unmarried    : 3033
##  Separated           :  883   Otherservice  :2777   Wife         : 1361
##  Widowed             :  902   (Other)       :8467
##                  race           sex         hoursperweek      native
##  Amer-Indian-Eskimo:  296   Female: 9682   Min.   : 1.00   Length:29170
##  Asian-Pac-Islander:  292   Male  :19488   1st Qu.:40.00   Class :character
##  Black             : 2832                  Median :40.00   Mode  :character
##  Other             :  129                  Mean   :40.45
##  White             :25621                  3rd Qu.:45.00
##                                            Max.   :99.00
##
##         income                  class
##  Above50K  : 7171   Private      :21794
##  AtBelow50K:21999   Selfempnotinc: 2313
##                     Localgov     : 1956
##                     Stategov     : 1210
##                     Selfempinc   :  991
##                     Federalgov   :  886
##                     (Other)      :   20
```

We can also see the number or above-50K income is significantly less than (25%) the at-or-below-50K income. This implies that there is prevalence of at-or-below-50K group in the dataset.

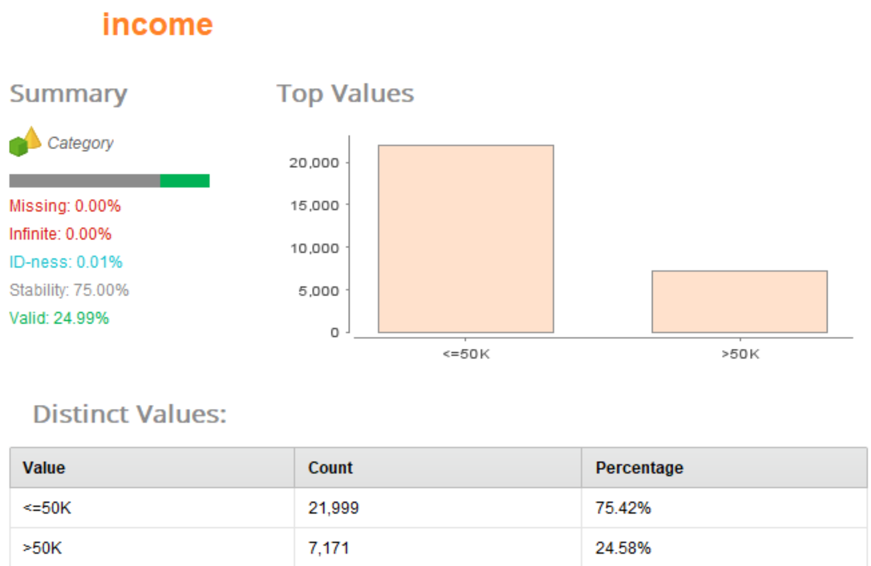


Figure 1: Income distribution

So we begin…

Analysis

Data Exploration, Processing and, Feature engineering

The adult pay dataset from census bureau is a ML ready database. For convenience, I have downloaded it from kaggle and stored in the same github repository as this code and markdown. The dataset is in zip format and is downloaded from https://github.com/rajeshharidas/havardxwork2/blob/main/adult.csv.zip. Then it is read into a data frame. The data is ML ready for the most part. However, there are some additional cleaning tasks needed before using the dataset. This processing is done in DatasetProcessingCode.R file. Code fragment is shown below. We perform the following cleaning tasks

1) Filter to keep only USA data
2) Remove all ? marks data from all the columns. Impute ? values to the modes in categorical columns like class, and occupation
3) Remove intermediate columns and columns that are not used in the project
4) Rename column names and column values to exclude non-alphanumeric characters
5) convert character labels to factors

```
# dataset wrangling and tidying

adultpayclean <- adultpay %>%
    filter(native.country == "United-States") %>%
    mutate(class = ifelse(workclass == "?", getmode(adultpay$workclass),
    ↪  str_replace_all(workclass,
       "-", ""))) %>%
    dplyr::select(-workclass, -capital.gain, -capital.loss) %>%
    rename(c(eduyears = education.num, maritalstatus = marital.status, hoursperweek =
    ↪  hours.per.week,
```

```
        native = native.country)) %>%
    mutate(maritalstatus = ifelse(maritalstatus == "?", getmode(adultpay$maritalstatus),
        str_replace_all(maritalstatus, "-", ""))) %>%
    mutate(occupation = ifelse(occupation == "?", getmode(adultpay$occupation),
    ↪   str_replace_all(occupation,
        "-", ""))) %>%
    mutate(education = ifelse(education == "?", getmode(adultpay$education),
    ↪   str_replace_all(education,
        "-", ""))) %>%
    mutate(relationship = ifelse(relationship == "?", getmode(adultpay$relationship),
        str_replace_all(relationship, "-", ""))) %>%
    mutate(native = ifelse(native == "?", "Unknown", str_replace_all(native, "-",
        ""))) %>%
    mutate(income = ifelse(income == "?", getmode(adultpay$income),
    ↪   str_replace_all(income,
        "<=50K", "AtBelow50K"))) %>%
    mutate(income = ifelse(income == "?", "Unknown", str_replace_all(income, ">50K",
        "Above50K")))
```

After performing this cleaning we see 29170 rows and 13 columns. We further analyze the makeup of the categorical columns.

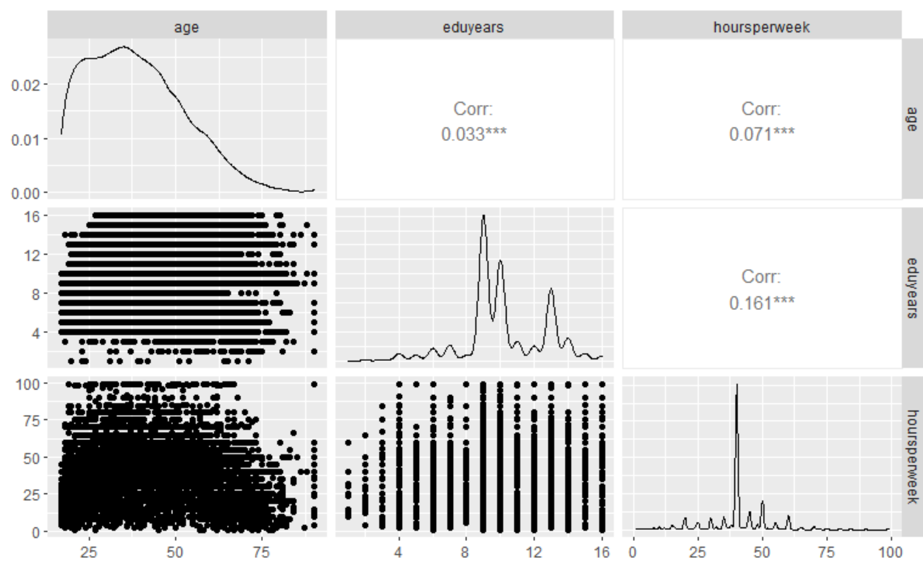


Figure 2: Comparison plots for numerical variables

Here are the comparison plots (generated by GGally package) for the numerical variables like age, eduyears and hoursperweek. There are very less people who have less than 4 years of education.Most people complete at least 9-10 years of education between 25-80 years age.Significant group of people stopped at elementary school. Significant group of people put in more than 40 hours per week past their retirement age. People who have 8-10 years of education put in more hours per week than others.

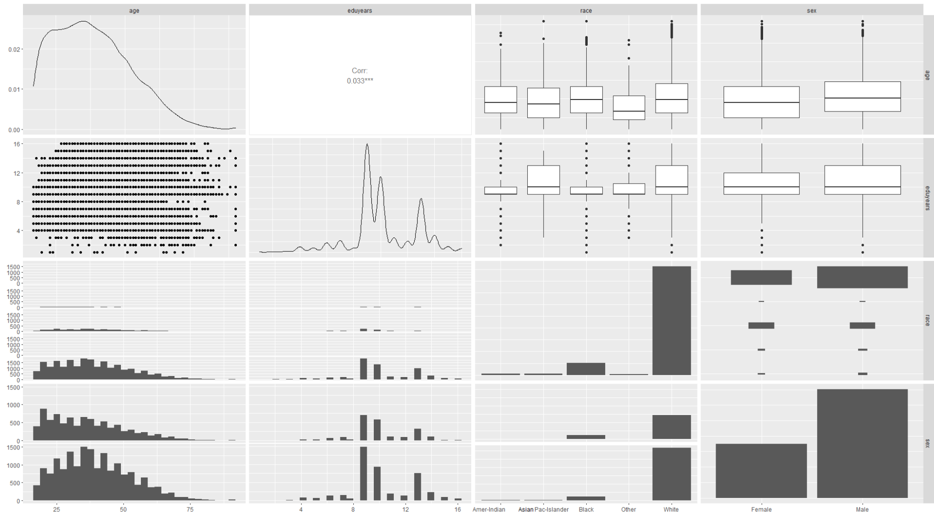
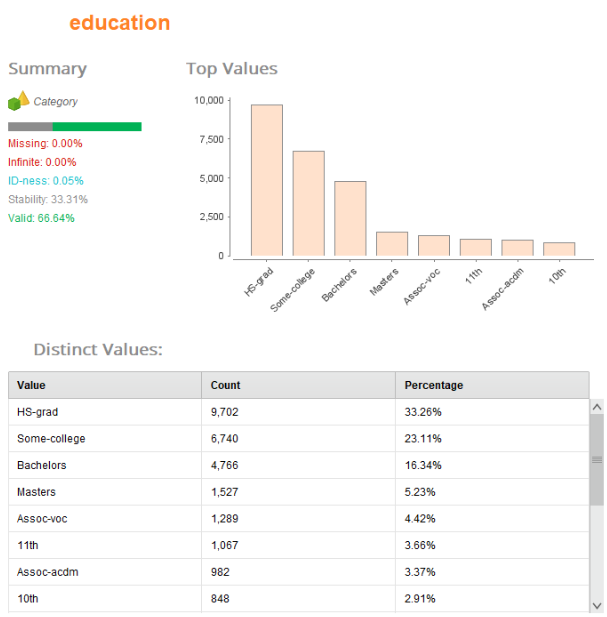


Figure 3: Comparison plots for categorical variables

Here are the comparison plots for the categorical variables. On an average white-Americans and black-Americans are gainfully employed at similar ages. More black and white people are employed past their retirement age when compared to other races. Similarly, more females are employed past their retirement age in comparison to male counterparts (outliers). More white males are employed than white females, however, the number of black males and females are about the same. There are more middle-aged men and younger females in the labor force.



Value	Count	Percentage
HS-grad	9,702	33.26%
Some-college	6,740	23.11%
Bachelors	4,766	16.34%
Masters	1,527	5.23%
Assoc-voc	1,289	4.42%
11th	1,067	3.66%
Assoc-acdm	982	3.37%
10th	848	2.91%

Figure 4: Education statistics

The number of high-school graduates are most frequent than other groups. There are 2 income distributions for education. One by the level of education and one by the number of years spent in education. Sure enough, most of the above-50K earners have at least high-school education. Bachelors and some college have more above-50K earners than other categories.

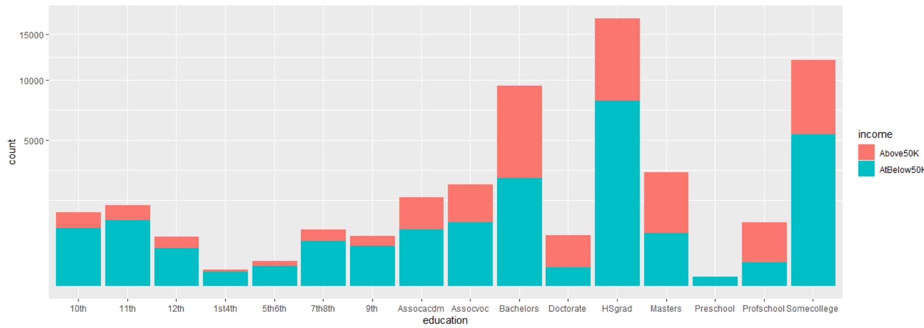


Figure 5: Education income distribution

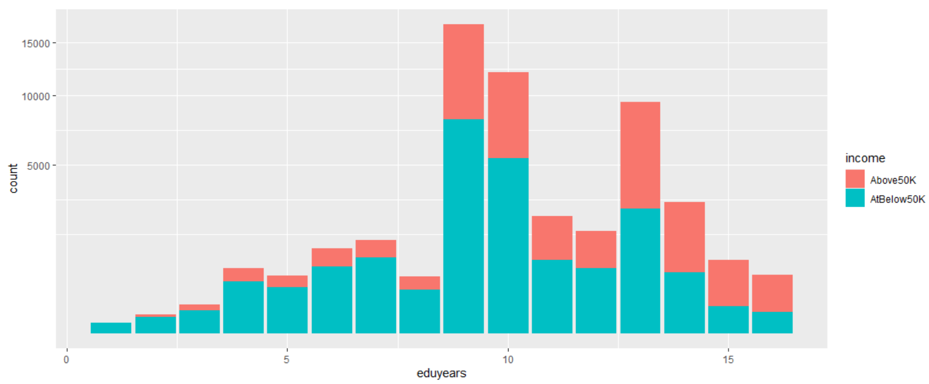


Figure 6: Education income distribution

Interestingly enough, in comparison to Bachelors and some college, group with masters have lesser above-50K. Additionally, the group that has at least 12 years of education have more above-50K than others. On an average the participants have 10 years of education with a standard deviation of 2.4. There are more at-or-below 50K earners in the high-school grad and less than 10 years of education categories.

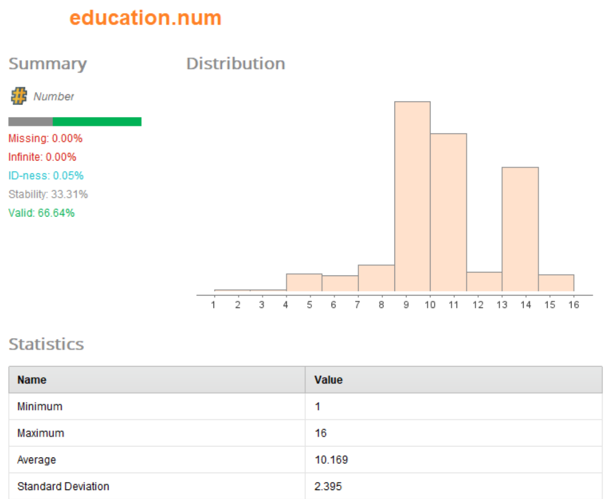


Figure 7: Years of education

The rest of the groups have sparse data. This could be because of missing or bad data that was removed during initial cleaning before it was loaded into kaggle.

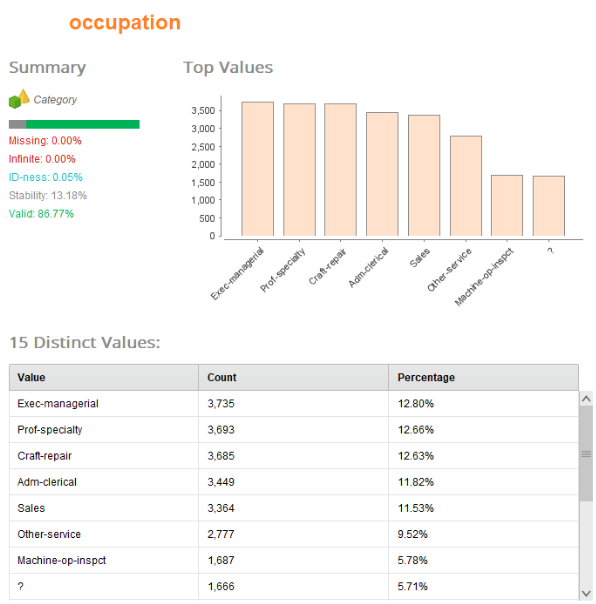


Figure 8: Occupation frequency

There are more executive and managerial occupations followed by professionals.

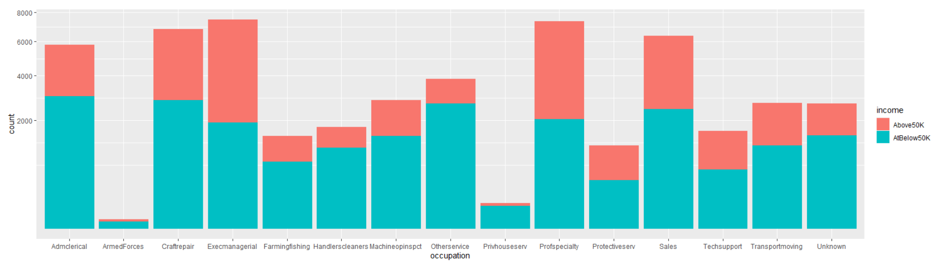
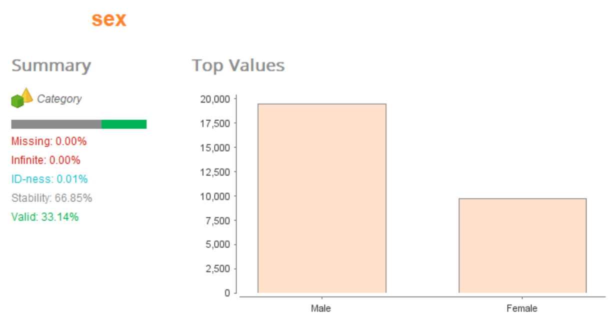


Figure 9: Occupation distribution

The same groups earn more and are in the above-50K category. There are fairly significant number of at-or-below-50K earners in the other categories. There are 1600 or so missing data (? or Unknown). This makes up 5-6% of the cleaned dataset.



Value	Count	Percentage
Male	19,488	66.81%
Female	9,682	33.19%

Figure 10: Sex feature characteristics

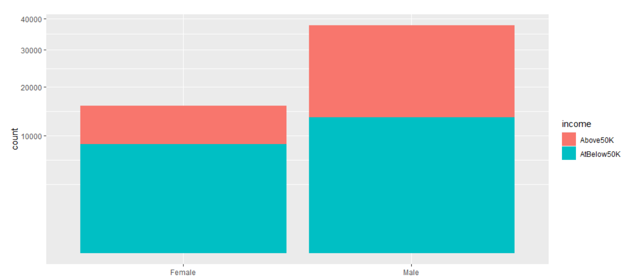
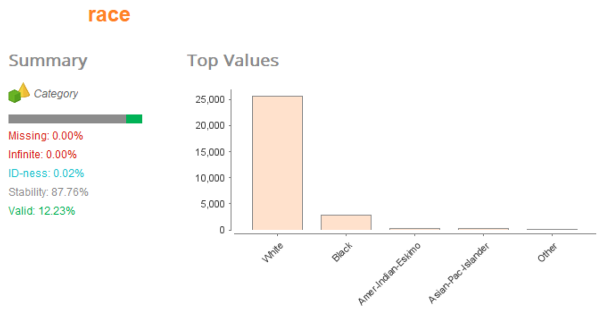


Figure 11: Sex feature characteristics

There are 67% more males in the workforce than females. The distribution also indicates that the number of males at above-50K are more than the number of females. Additionally, there are more males at-or-below-50K than females as well.



5 Distinct Values:

Value	Count	Percentage
White	25,621	87.83%
Black	2,832	9.71%
Amer-Indian-Eskimo	296	1.01%
Asian-Pac-Islander	292	1.00%
Other	129	0.44%

Figure 12: Race distribution

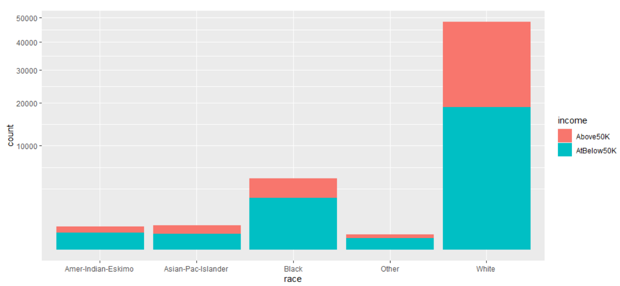
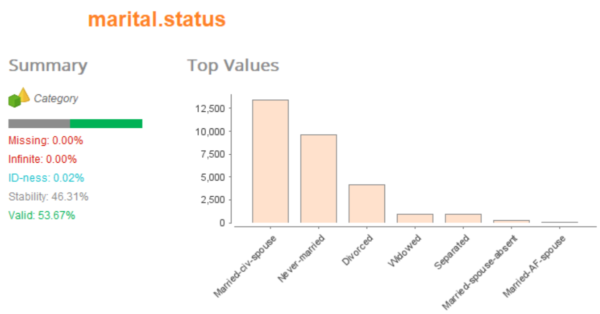


Figure 13: Race distribution

There are more white-Americans in the labor force in both the above-50K and at-or-below-50K categories. Black-Americans and other race follow next. The proportion of above-50K earners are lesser in other races when compared to white-Americans.



7 Distinct Values:

Value	Count	Percentage
Married-civ-spouse	13,368	45.83%
Never-married	9,579	32.84%
Divorced	4,162	14.27%
Widowed	902	3.09%
Separated	883	3.03%
Married-spouse-absent	253	0.87%
Married-AF-spouse	23	0.08%

Figure 14: Marital Status frequency

The number of married with civilian spouses are more than any other groups in labor force. The number of above-50K and at-or-below-50K earners are also more in this group when compared to others. Singles follow next.

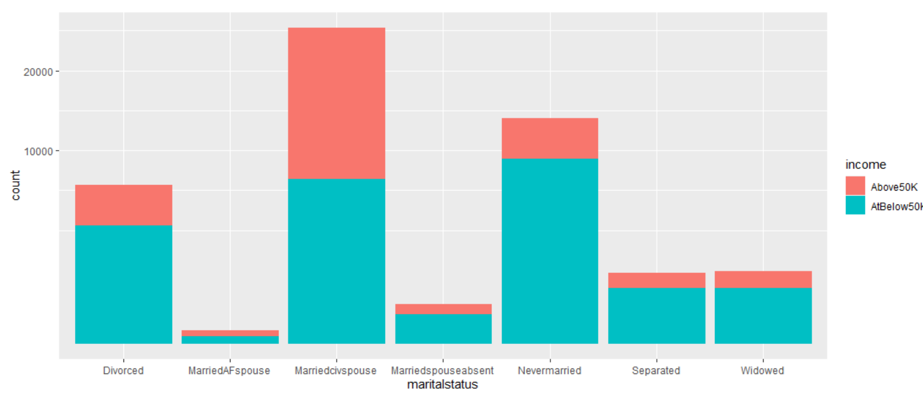
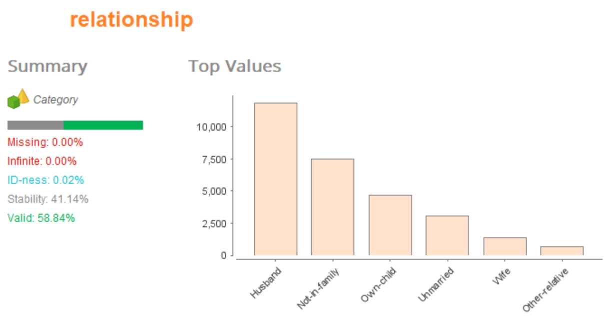


Figure 15: Marital Status distribution

The number of at-or-below-50K are more than the above-50K in the singles group and most of them are younger population. The plot also suggests most of this group is below 25 years age group. This can be correlated with the income by age group chart. The number of married with military spouses is less which may be due to lack of data.



6 Distinct Values:

Value	Count	Percentage
Husband	11,861	40.66%
Not-in-family	7,528	25.81%
Own-child	4,691	16.08%
Unmarried	3,033	10.40%
Wife	1,361	4.67%
Other-relative	696	2.39%

Figure 16: Relationship distribution

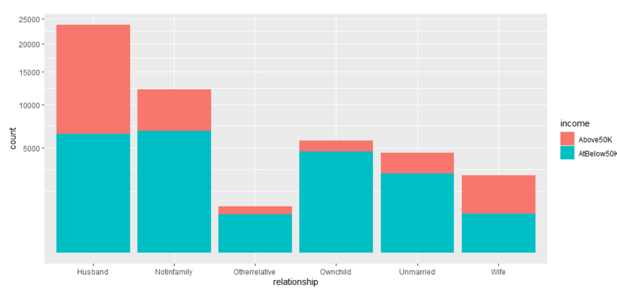


Figure 17: Relationship distribution

The number of husbands who earn more than 50K and at-or-below-50K are also more than the number of wives. Unmarried individuals also are more in the at-or-below-50K category.

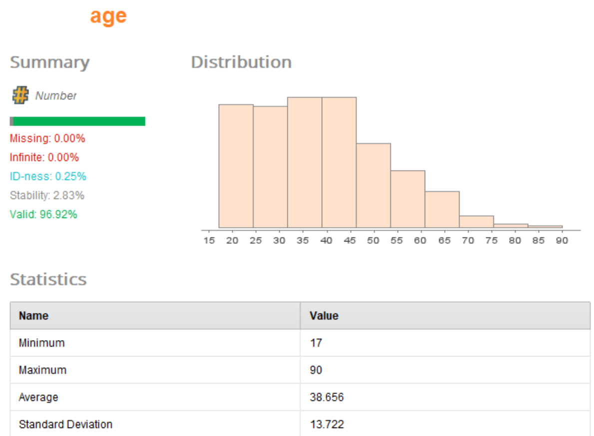


Figure 18: Age distribution

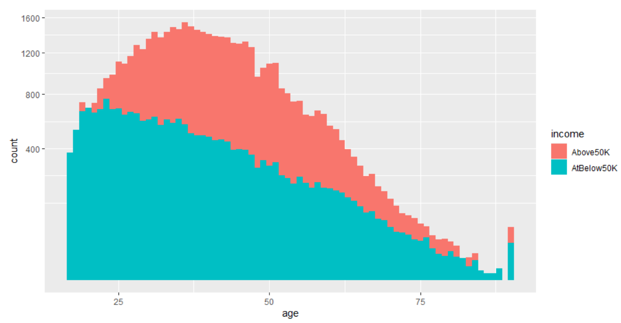
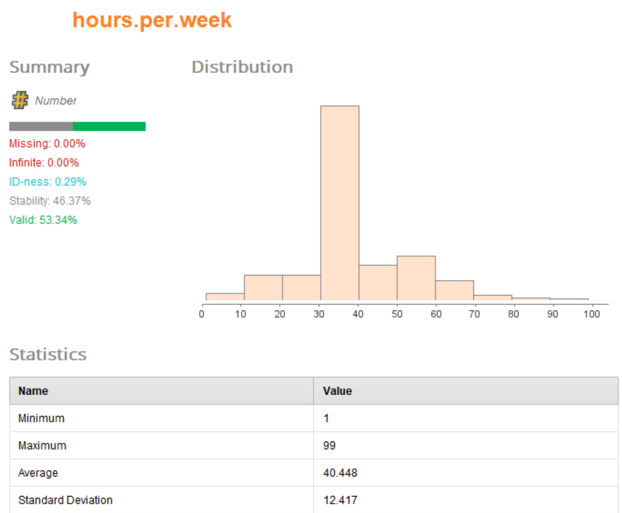


Figure 19: Age distribution

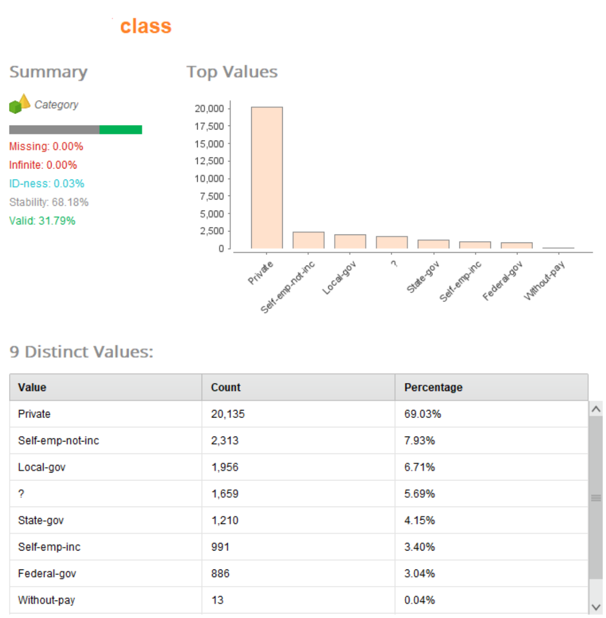
The number of people from age 30-45 are the most frequent in the distribution and the same group have more above-50K incomes. There are 12 outliers in the age group above 80 who earn Above-50K. There are similar outliers in age-sex and age-race comparisons. However, the numbers are not significant enough to skew the distributions.



Name	Value
Minimum	1
Maximum	99
Average	40.448
Standard Deviation	12.417

Figure 20: Hours worked

On a average most participants worked 40 hours a week followed by 50-60 hours.



Value	Count	Percentage
Private	20,135	69.03%
Self-emp-not-inc	2,313	7.93%
Local-gov	1,956	6.71%
?	1,659	5.69%
State-gov	1,210	4.15%
Self-emp-inc	991	3.40%
Federal-gov	886	3.04%
Without-pay	13	0.04%

Figure 21: Class distribution

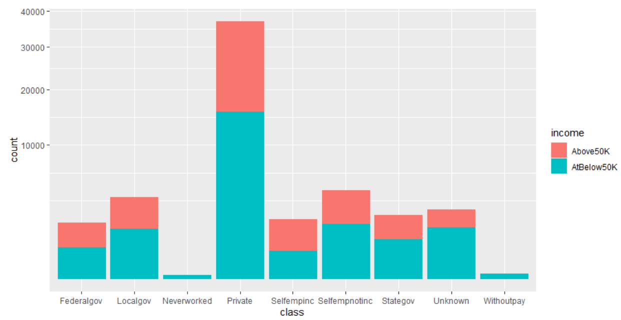


Figure 22: Class distribution

The number of private employer workers are more than other categories. They also earn more above-50K and at-or-below-50K than other categories.

As shown above the occupation and class categorical columns have more than 1600 Unknown data. As these are categorical data, we will impute them using mode. However, as this is 6% of the data, adding these to the predictor columns could skew the predictions and introduce errors. We will exclude this from our list of predictors. We will experiment with these two features in our final model. We also removed capital gains and loss columns because we do not have good realistic data for both of these columns. The final list of predictors/features used to predict the outcome income level are age, education, years of education (a.ka. eduyears), maritalstatus, relationship,race, sex, and hoursperweek. As analyzed in the above section, there is sufficient degree of variability and correlation between these features and how they impact the income levels based on socio-economic and cultural factors.

Process

The high-level process methodology used in this project is along the same lines as the CRISP-DM (Cross Industrial Standard Process for Data Mining) methodology. We got a business understanding of the problem we are trying to solve. We analyzed the raw data. Next we prepare the dataset from the raw data.

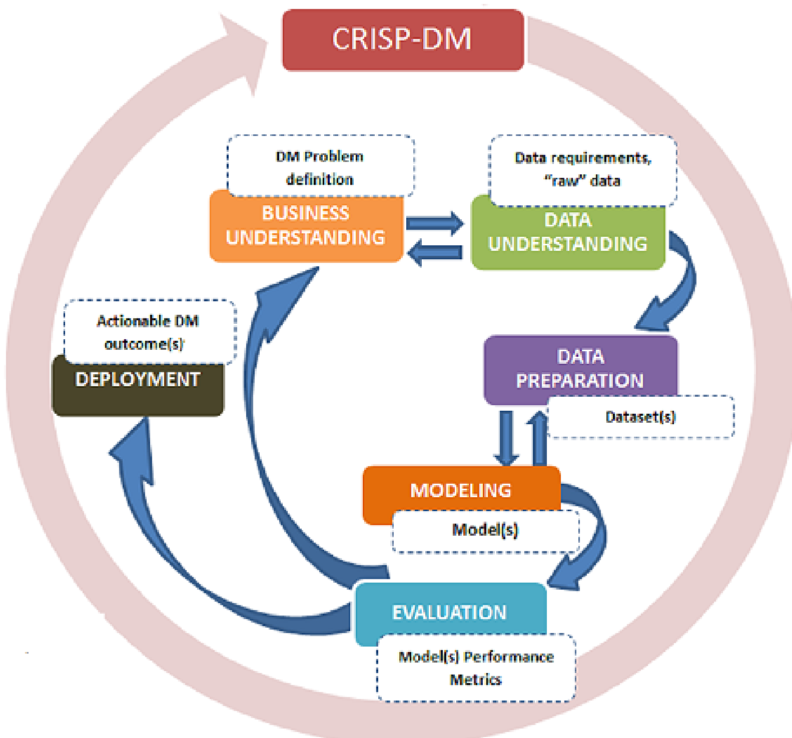


Figure 23: Process Methodology

The code DatasetProcessingCode.R downloads the dataset from a convenient location in github (originally downloaded from kaggle) and then unzips it and converts the dataset into a R data frame. It then does extensive cleaning, processing and munging of the data to make it tidy and meaningful for analysis. It then splits the original dataset into training and validation datasets. All analysis is done on this training dataset. Once the training is done the validation is done on the test dataset.

The caret package includes the function createDataPartition that helps us generates indexes for randomly splitting the data into training and test sets. The argument times is used to define how many random samples of indexes to return, the argument p is used to define what proportion of the data is represented by the index, and the argument list is used to decide if we want the indexes returned as a list or not. We can use the test_index from the createDataPartition function call to define the training and test sets.

```
# split dataset into training and validation sets

set.seed(1, sample.kind = "Rounding")

test_index <- createDataPartition(y = adultpayclean$income, times = 1, p = 0.1, list =
↪  FALSE)

adultpayclean_train <- adultpayclean[-test_index, ]

adultpayclean_validation <- adultpayclean[test_index, ]
```

```
dim(adultpayclean)
# [1] 29170 13

dim(adultpayclean_train)
# [1] 26252 13

dim(adultpayclean_validation)
# [1] 2918 13
```

We then develop an algorithm using only the training set. Once we are done developing the algorithm, we then freeze it and evaluate it using the test set.

The simplest way to evaluate the algorithm for the adultpay dataset is by simply reporting the proportion of cases that were correctly predicted in the test set. This metric is referred to as overall accuracy. Some times accuracy is skewed due to prevalence or bias of one class or the other, therefore we will weigh in on sensitivity and specificity (check true/false positives and true/false negatives) when choosing the final model.

We will also use F1 measure to validate the accuracy of the sensitivity and specificity and see if we have a good precision. F1 score is a combination of two important error metrics: Precision and Recall. Thus, it can be considered as the Harmonic mean of Precision and Recall error metrics for an imbalanced dataset with respect to binary classification of data. As income level has only 30% data for above-50K class this score is useful. The highest possible value of an F-score is 1.0, indicating perfect precision and recall, and the lowest possible value is 0. Higher the precision the better.

We will also use a widely used plot receiver operating characteristic (ROC) curve. Receiver Operating Characteristic curves are a popular way to visualize the trade-offs between sensitivity and specificity in a binary classifier. The ROC curve plots sensitivity (TPR) versus 1 - specificity or the false positive rate (FPR). This also gives us Area under the curve (a.k.a. AUC). The AUC summarizes ROC in a single value. The probabilistic interpretation is that if you randomly choose a positive case and a negative case, the probability that the positive case outranks the negative case according to the classifier is given by the AUC. Higher the AUC score the better. This indicates that the model has higher chances for getting correct predictions.

Model Creation

Simplest model using random sampling

We are now going to evaluate various algorithms progressively

We first start with a simple model. We sample randomly for the desired outcome. We then compare it with the actual outcomes and take a mean of the results. The result is 50% which is expected for guessing a binary outcome. Its akin to tossing a coin and getting head or tail. The chances are 50/50.

```
# Plain old guessing

seat_of_the_pants <- sample(c("Above50K", "AtBelow50K"), length(test_index), replace =
↪  TRUE) %>%
    factor(levels = levels(adultpayclean_validation$income))

mean(seat_of_the_pants == adultpayclean_validation$income)

# [1] 0.5010281
```

```r
# build a confusion matrix for this simple model

cm <- confusionMatrix(data = seat_of_the_pants, reference =
↪  adultpayclean_validation$income)

# cm
```

```
Confusion Matrix and Statistics

            Reference
Prediction   Above50K AtBelow50K
  Above50K        347       1087
  AtBelow50K      371       1113

               Accuracy : 0.5003
                 95% CI : (0.482, 0.5186)

    No Information Rate : 0.7539
    P-Value [Acc > NIR] : 1
                  Kappa : -0.0081
 Mcnemar's Test P-Value : <2e-16
            Sensitivity : 0.4833
            Specificity : 0.5059
         Pos Pred Value : 0.2420
         Neg Pred Value : 0.7500
             Prevalence : 0.2461
         Detection Rate : 0.1189
   Detection Prevalence : 0.4914
      Balanced Accuracy : 0.4946
       'Positive' Class : Above50K
```

```r
p <- 0.1

n <- length(test_index)

y_hat <- sample(c("Above50K", "AtBelow50K"), n, replace = TRUE, prob = c(p, 1 - p)) %>%
factor(levels = levels(adultpayclean_validation$income))

mean(y_hat == adultpayclean_validation$income)

# [1] 0.7076765

p <- 0.9

n <- length(test_index)

y_hat <- sample(c("Above50K", "AtBelow50K"), n, replace = TRUE, prob = c(p, 1 - p)) %>%
factor(levels = levels(adultpayclean_validation$income))

mean(y_hat == adultpayclean_validation$income)

# [1] 0.2964359
```

```
f1_guess

# [1] 0.3224907

# Area under the curve

auc_guess <- auc(ifelse(adultpayclean_validation$income == "Above50K", 1, 2),
↪  ifelse(seat_of_the_pants ==
    "Above50K", 1, 2))

auc_guess

# Area under the curve: 0.4946
```

We then construct the confusion matrix, which basically tabulates each combination of prediction and actual value. We see that the above-50K and at-or-below-50K are almost evenly distributed with a slightly higher prevalence of the at-or-below-50K income level.

We can verify this by adjusting the probability of our sampling to skew towards above-50K vs at-or-below-50K and vice-versa.

Prevalence can result in skewed results. We will keep an eye on the other metrics like sensitivity and specificity in addition to accuracy. In this case low prevalence matches with the expected accuracy.

The F1 score for this guessing is 0.3224 and Area under the curve is 0.4946. This is in line with our expectations for mere guessing of the outcome.

Our goal is to improve the accuracy > 80% while keeping sensitivity and specificity under check. Hence we further analyze the impact of other features on the income levels.

Logistic regression using linear models

We will start with a simple logistic model - linear model. We use the features age, education, eduyears, sex, race, maritalstatus, relationship and hoursperweek.

Both linear and logistic regressions provide an estimate for the conditional expectation:

$$E(\,Y \mid X = x)$$

which in the case of binary data is equivalent to the conditional probability:

$$Pr(\,Y = 1 \mid X = x)$$

We can use this to arrive at y_hat_logit. Since we have 8 features $X = x$ is more like $X_i = x_i$ where i is from 1 to 8.

The lm function applies this formula on adultpay dataset. Where y is the outcome and ϵ is the error. Sum of $\beta_1 x_1$ through $\beta_8 x_8$ is a linear combination of x_1 through x_8.

y = $\alpha + \beta_1$(age)$+\beta_2$(eduyears)$+\beta_3$(sex)$+\beta_4$(race)$+\beta_5$(hoursperweek)$+\beta_6$(maritalstatus)$+\beta_7$(relationship)$+\beta_8$(education)$+\epsilon$

```r
# linear model

lm_fit <- adultpayclean_train %>%
    mutate(y = as.numeric(income == "Above50K")) %>%
    lm(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus + relationship +
        education, data = .)

p_hat_logit <- predict(lm_fit, newdata = adultpayclean_validation)

y_hat_logit <- ifelse(p_hat_logit > 0.5, "Above50K", "AtBelow50K") %>%
    factor

accuracy_lm <- confusionMatrix(y_hat_logit,
↪  adultpayclean_validation$income)$overall[["Accuracy"]]

accuracy_lm

# [1] 0.8153

f1_lm

# [1] 0.54128

# Area under the curve: 0.81709
```

```
Confusion Matrix and Statistics

             Reference
    Prediction   Above50K AtBelow50K
    Above50K        318        139
    AtBelow50K      400       2061

                Accuracy : 0.8153
                  95% CI : (0.8007, 0.8292)
     No Information Rate : 0.7539
     P-Value [Acc > NIR] : 1.243e-15
                   Kappa : 0.4327
  Mcnemar's Test P-Value : < 2.2e-16
             Sensitivity : 0.4429
             Specificity : 0.9368
          Pos Pred Value : 0.6958
          Neg Pred Value : 0.8375
              Prevalence : 0.2461
          Detection Rate : 0.1090
    Detection Prevalence : 0.1566
       Balanced Accuracy : 0.6899
        'Positive' Class : Above50K
```

The accuracy for this model is 0.8153 however, sensitivity is 0.4429 and specificity is 0.9368. This indicates this model has higher ratio of negative outcomes than positive outcomes. It does have better accuracy, confidence interval, and F1 score than plain old guessing. Prevalence remains the same. AUC has improved as well. Lets see if we can do better

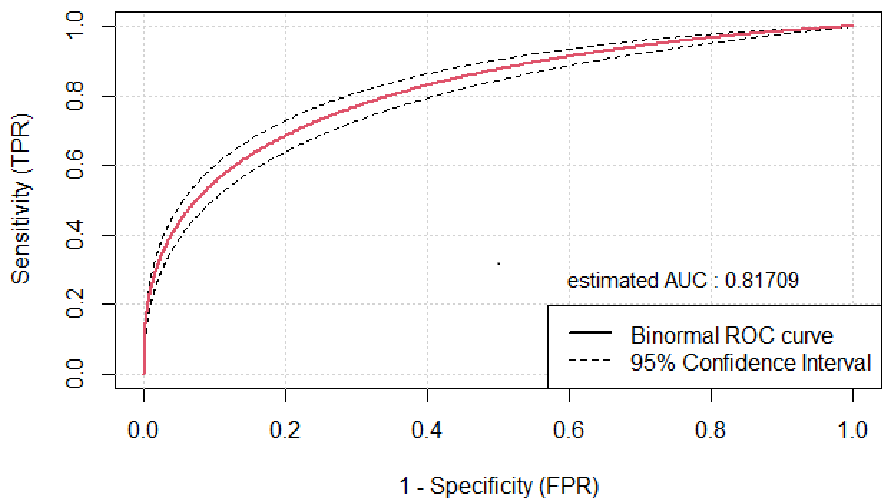


Figure 24: LM classification summary

We will now experiment with the general linear model (glm). We will be consistent with the features used across all algorithms. Glm uses the following formula

$$\log\left[\frac{P(\mathrm{y}=1)}{1-P(\mathrm{y}=1)}\right]=\alpha+\beta_1(\mathrm{age})+\beta_2(\mathrm{eduyears})+\beta_3(\mathrm{sex})+\beta_4(\mathrm{race})+\beta_5(\mathrm{hoursperweek})+\beta_6(\mathrm{maritalstatus})+\beta_7(\mathrm{relationship})+\beta_8(\mathrm{education}))$$

Where y is the outcome and ϵ is the error. Sum of $\beta_1 x_1$ through $\beta_8 x_8$ is a linear combination of x_1 through x_8

Logistics model assures that the estimate $Pr(Y = 1 \mid X = x)$ is between 0 and 1. Glm converts probability into log odds - $\log\left[\frac{P(\mathrm{y}=1)}{1-P(\mathrm{y}=1)}\right]$. We will then translate the probability into a decision rule to predict Above50K if probability is > 0.5. We then compare the predictions to the outcomes using confusionMatrix.

```
# general linear model

glm_fit <- adultpayclean_train %>%  glm_fit <- adultpayclean_train %>%
mutate(y = as.numeric(income == "Above50K")) %>%  mutate(y = as.numeric(income
mutate(y = as.numeric(income == "Above50K")) %>%  == "Above50K")) %>%
glm(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus + relationship +
    education, data = ., family = "binomial")

p_hat_logit <- predict(glm_fit, newdata = adultpayclean_validation)

y_hat_logit <- ifelse(p_hat_logit > 0.5, "Above50K", "AtBelow50K") %>%
    factor

accuracy_glm <- confusionMatrix(y_hat_logit,
↪   adultpayclean_validation$income)$overall[["Accuracy"]]

accuracy_glm
# [1] 0.8105
```

```r
f1_glm

# [1] 0.50135

# Area under the curve: 0.81817
```

```
 Confusion Matrix and Statistics

             Reference
 Prediction   Above50K AtBelow50K
   Above50K        278        113
   AtBelow50K      440       2087

                Accuracy : 0.8105
                  95% CI : (0.7958, 0.8246)
     No Information Rate : 0.7539
     P-Value [Acc > NIR] : 1.758e-13
                   Kappa : 0.3967
  Mcnemar's Test P-Value : < 2.2e-16
             Sensitivity : 0.38719
             Specificity : 0.94864
          Pos Pred Value : 0.71100
          Neg Pred Value : 0.82588
              Prevalence : 0.24606
          Detection Rate : 0.09527
    Detection Prevalence : 0.13400
       Balanced Accuracy : 0.66791
        'Positive' Class : Above50K
```

glm produces accuracy of 0.8105, however, specificity 0.94864 is still higher than sensitivity 0.38719 and prevalence about the same. CI and F1 scores can be better. AUC can also do better. We can do better!

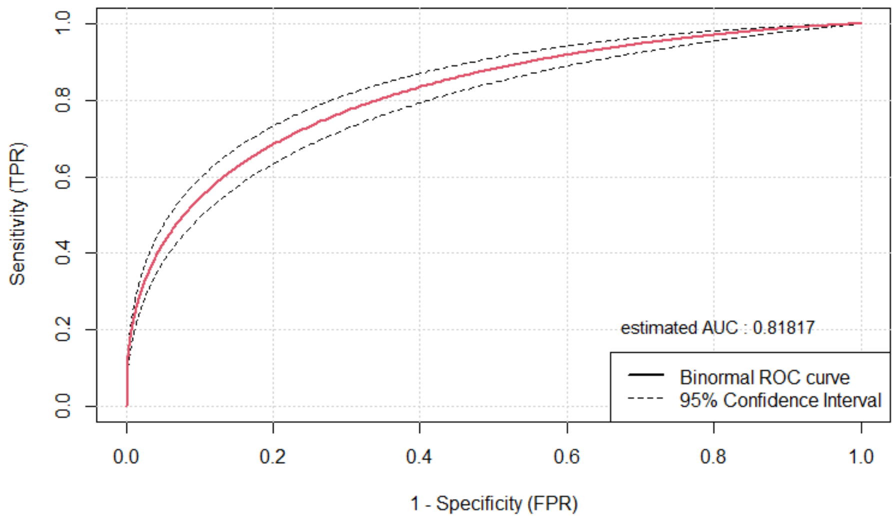


Figure 25: GLM classification summary

Naive Bayes

We will now experiment with generative models like naive bayes classification. It assumes the features that go into the model are independent of each other.That is changing the value of one feature, does not directly influence or change the value of any of the other features used in the algorithm. When we analyzed the dataset it didn't appear to have independent features. An example of dependent feature would be age and education years, age and martial status etc. However, lets see how naive bayes performs with this assumption.

```
# Naive Bayes

train_nb <- adultpayclean_train %>%
    mutate(y = as.factor(income == "Above50K")) %>%
    naiveBayes(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
    ↪  relationship +
        education, data = .)

y_hat_nb <- predict(train_nb, newdata = adultpayclean_validation)

cm_tab <- table(adultpayclean_validation$income == "Above50K", y_hat_nb)

cm_nb <- confusionMatrix(cm_tab)

## Accuracy : 0.8009

f1_nb

# [1] 0.86064

# Area under the curve: 0.80143
```

```
Confusion Matrix and Statistics

       y_hat_nb
        FALSE TRUE
  FALSE  1794  406
  TRUE    175  543

               Accuracy : 0.8009
                 95% CI : (0.7859, 0.8152)
    No Information Rate : 0.6748
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5158

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9111
            Specificity : 0.5722
         Pos Pred Value : 0.8155
         Neg Pred Value : 0.7563
             Prevalence : 0.6748
         Detection Rate : 0.6148
   Detection Prevalence : 0.7539
      Balanced Accuracy : 0.7417

       'Positive' Class : FALSE
```

Naive bayes gives us an accuracy of 0.8009 with a lower prevalence than other algorithms but higher than plain guessing. However the confidence interval is lower and accuracy of the prediction is lower too. There are more false negatives as well. F1 score has increased. This indicates there are more true positives as well. However, given that this is an imbalanced dataset, we can try other advanced algorithms and compare this later.

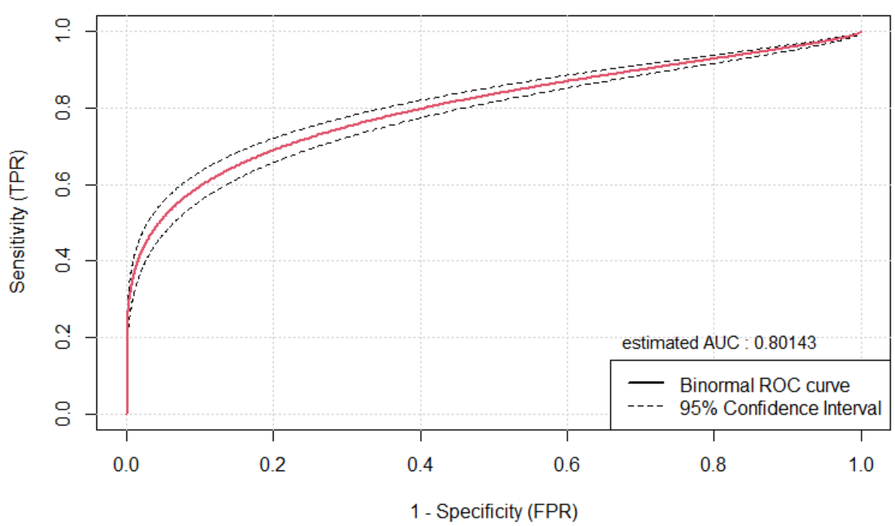


Figure 26: naive bayes classification summary

Lets explore k-nearest model now with same feature set. We will use cross-validation to tune the k parameter. By default, the cross validation is performed by taking 25 bootstrap samples comprised of 25% of the observations. For the kNN method, the default is to try k=5,7,9. We change this using the tuneGrid parameter. We will try the k values in the following sequence k = seq(3, 71, 2). Running this code will take several seconds. This is because when we run the algorithm, we will have to compute a distance between each observation in the test set and each observation in the training set. There are a lot of computations. Therefore, we use the trainControl function to make the code above go a bit faster by using, 10-fold cross validation. This means we have 10 samples using 10% of the observations each. We set the seed because cross validation is a random procedure and we want to make sure the result here is reproducible

K-Nearest Neighbors

```
# KNN

temp <- adultpayclean_train %>%
    mutate(y = as.factor(income == "Above50K"))

set.seed(2008)

control <- trainControl(method = "cv", number = 10, p = 0.9)

train_knn <- train(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
    relationship + education, method = "knn", data = temp, tuneGrid = data.frame(k =
↪   seq(3,
    71, 2)), trControl = control)

train_knn$bestTune

y_hat_knn <- predict(train_knn, adultpayclean_validation, type = "raw")

accuracy_knn <- confusionMatrix(y_hat_knn, as.factor(adultpayclean_validation$income ==
    "Above50K"))$overall[["Accuracy"]]

ggplot(train_knn, highlight = TRUE)

accuracy_knn

# [1] 0.80535

f1_knn

# [1] 0.87522

# Area under the curve: 0.78721
```

The k parameter that lead to maximum accuracy can be obtained by bestTune. The plot for that is shown in the figure "KNN tuning".

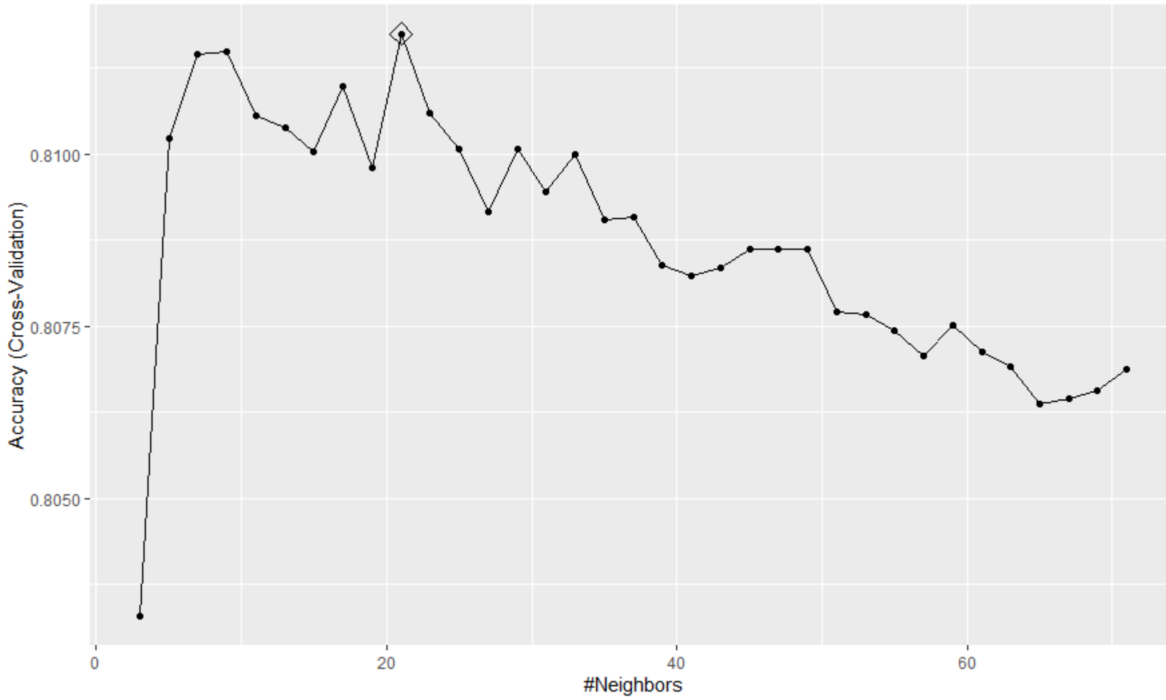


Figure 27: knn tuning

Here is the confusion matrix for the knn tuned raw model

```
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE  1992  360
     TRUE    208  358

               Accuracy : 0.8053
                 95% CI : (0.7905, 0.8196)
    No Information Rate : 0.7539
    P-Value [Acc > NIR] : 2.195e-11

                  Kappa : 0.4351

 Mcnemar's Test P-Value : 2.361e-10

            Sensitivity : 0.9055
            Specificity : 0.4986
         Pos Pred Value : 0.8469
         Neg Pred Value : 0.6325
             Prevalence : 0.7539
         Detection Rate : 0.6827
   Detection Prevalence : 0.8060
      Balanced Accuracy : 0.7020

       'Positive' Class : FALSE
```

knn raw model produces accuracy of 0.8053, however, specificity is lower than sensitivity and prevalence is now 0.753. We now see the positive prediction is better. F1 score has improved. However, the AUC score is less than naive bayes. This indicates that the number of true positives are lesser than before. Balanced accuracy is better. Can we do better with accuracy and precision?

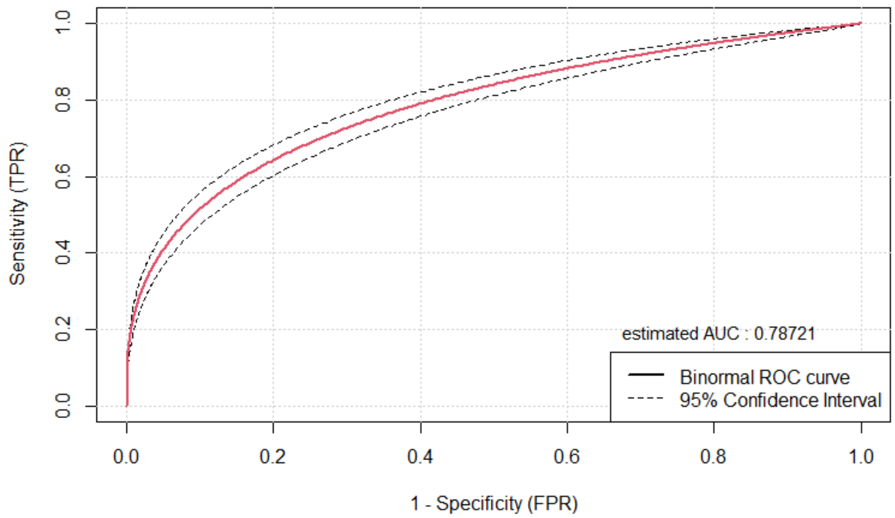


Figure 28: knn classification summary

We will now try to use classification with knn3. We will again use different values of k but using map_df function to repeat the above for each one. Running this classification model is going to be slow as it has to iterate through all the k values and find the one that is the highest.

```r
ks <- seq(3, 251, 2)

knntune <- map_df(ks, function(k) {

    temp <- adultpayclean_train %>%
        mutate(y = as.factor(income == "Above50K"))

    temp_test <- adultpayclean_validation %>%
        mutate(y = as.factor(income == "Above50K"))

    knn_fit <- knn3(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
        relationship + education, data = temp, k = k)

    y_hat <- predict(knn_fit, temp, type = "class")

    cm_train <- confusionMatrix(y_hat, temp$y)

    train_error <- cm_train$overall["Accuracy"]

    y_hat <- predict(knn_fit, temp_test, type = "class")

    cm_test <- confusionMatrix(y_hat, temp_test$y)

    test_error <- cm_test$overall["Accuracy"]
```

```
    tibble(train = train_error, test = test_error)

})

accuracy_knntune <- max(knntune$test)

# get the confusion matrix for that k

knn_fit <- knn3(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
    relationship, data = temp, k = ks[which.max(knntune$test)])

y_hat <- predict(knn_fit, adultpayclean_validation, type = "class")

cm_knntune <- confusionMatrix(y_hat, as.factor(adultpayclean_validation$income ==
    "Above50K"))

f1_knntune

# [1] 0.87745

# Area under the curve: 0.7941
```

```
 Confusion Matrix and Statistics

           Reference
 Prediction FALSE TRUE
      FALSE  1994  351
      TRUE    206  367

               Accuracy : 0.8091
                 95% CI : (0.7944, 0.8232)
    No Information Rate : 0.7539
    P-Value [Acc > NIR] : 6.680e-13
                  Kappa : 0.448
  Mcnemar's Test P-Value : 1.051e-09
            Sensitivity : 0.9064
            Specificity : 0.5111
         Pos Pred Value : 0.8503
         Neg Pred Value : 0.6405
             Prevalence : 0.7539
         Detection Rate : 0.6833
   Detection Prevalence : 0.8036
      Balanced Accuracy : 0.7088
       'Positive' Class : FALSE
```

The accuracy of the knn tuned classification is 0.8091 with a sensitivity of 0.9064 and specificity of 0.5111. This is better than the previous models. Prevalence is just about the same as raw knn. Balanced accuracy has improved. F1 score is also at ~ 88% and AUC has improved a bit.

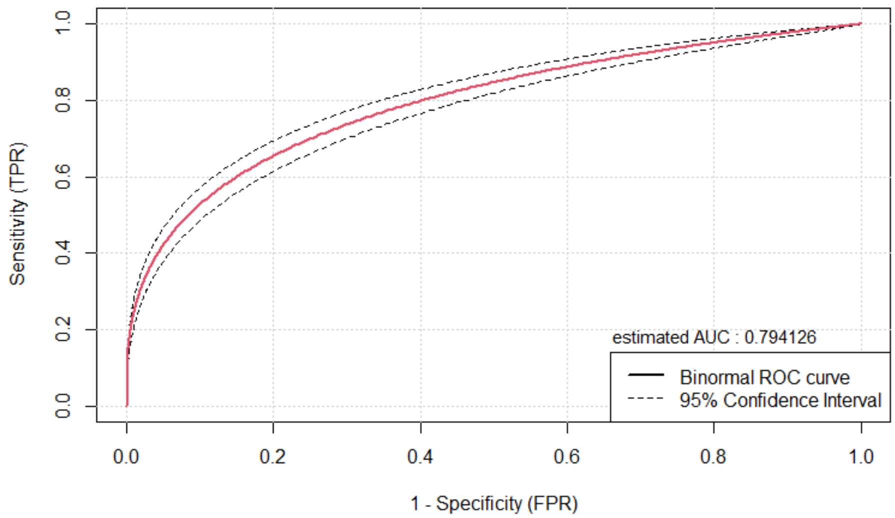


Figure 29: knn tune classification summary

Recursive partitioning with rpart

Next we use classification trees, or decision trees. We use the recursive partitioning library rpart for this. The general idea is to define an algorithm that uses data to create trees with predictions at the ends, referred to as nodes. Decision trees operate by predicting a categorical outcome variable Y by partitioning the predictors.

```
train_rpart <- train(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
    relationship + education, method = "rpart", tuneGrid = data.frame(cp = seq(0,
    0.1, len = 25)), data = temp)

y_hat <- predict(train_rpart, adultpayclean_validation)

accuracy_rpart <- confusionMatrix(y_hat, as.factor(adultpayclean_validation$income ==
    "Above50K"))$overall["Accuracy"]

accuracy_rpart

# Accuracy

# 0.8211

f1_rpart

# [1] 0.8846664

# Area under the curve: 0.81573
```

```
Confusion Matrix and Statistics
```

```
          Reference
Prediction FALSE TRUE
     FALSE  2002  324
     TRUE    198  394

               Accuracy : 0.8211
                 95% CI : (0.8067, 0.8349)
    No Information Rate : 0.7539
    P-Value [Acc > NIR] : < 2.2e-16
                  Kappa : 0.4876
 Mcnemar's Test P-Value : 4.472e-08
            Sensitivity : 0.9100
            Specificity : 0.5487
         Pos Pred Value : 0.8607
         Neg Pred Value : 0.6655
             Prevalence : 0.7539
         Detection Rate : 0.6861
   Detection Prevalence : 0.7971
      Balanced Accuracy : 0.7294
       'Positive' Class : FALSE
```

The results of the tuning of the decision trees can be seen in this figure

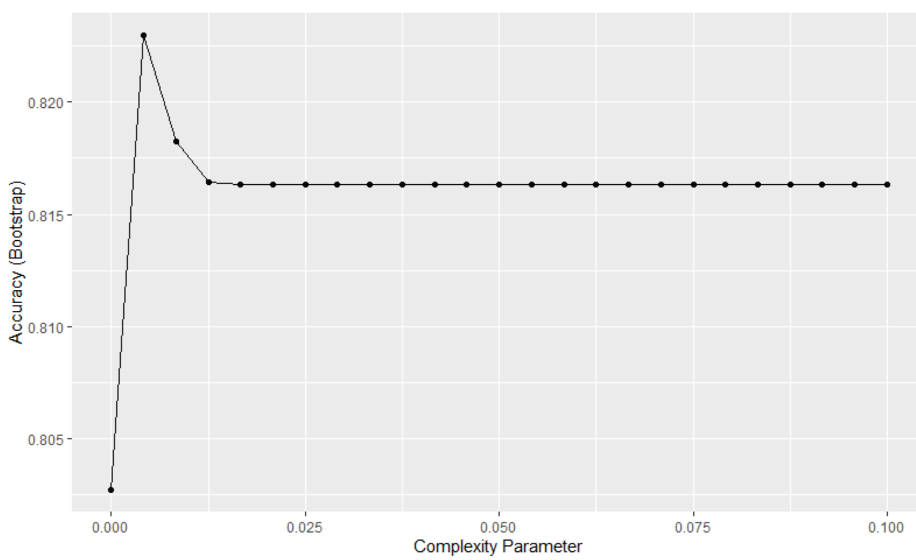


Figure 30: rpart accuracy

The rpart algorithm followed the rule below to classify the dataset

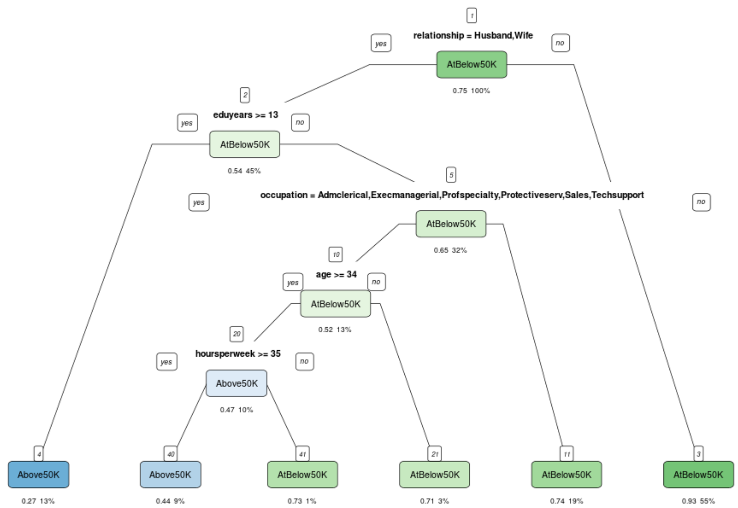


Figure 31: rpart decision tree

The accuracy of recursive partitioning is 0.8211, sensitivity is higher at 0.91 and specificity is at 0.5487 with prevalence about the same. Balanced accuracy has improved. Confidence intervals for positive and negative has improved as well. The F1 score for rpart is 0.88467 and AUC is 0.81573 which is a positive sign.

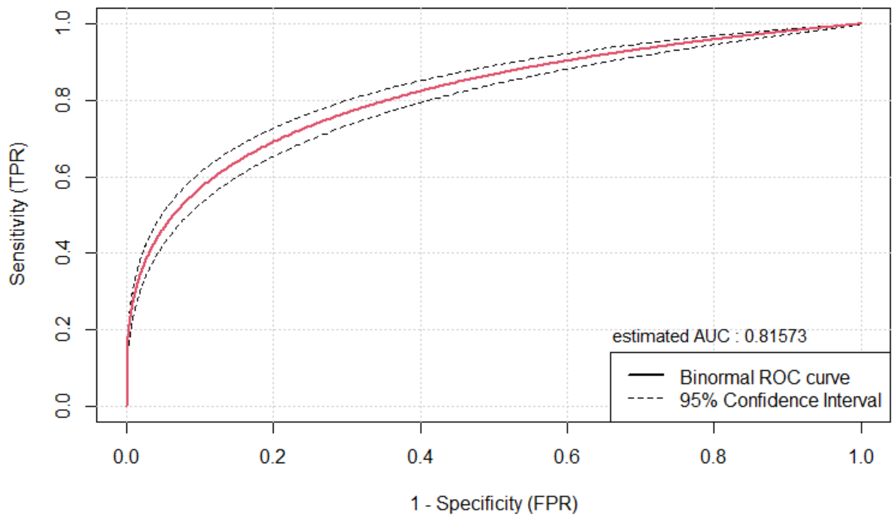


Figure 32: rpart classification summary

Classification trees have certain advantages that make them very useful. They are highly interpretable, even more so than linear models. They are easy to visualize (if small enough). Finally, they can model human decision processes and don't require use of dummy predictors for categorical variables. On the other hand, the approach via recursive partitioning can easily over-train and is therefore a bit harder to train than, for example, linear regression or kNN. Furthermore, in terms of accuracy, it is rarely the best performing method since it is not very flexible and is highly unstable to changes in training data. Random forests, explained next, improve on several of these shortcomings. We will next look at random forest algorithm.

Random forests

The goal of random forest is to improve prediction performance and reduce instability by averaging multiple decision trees. The first step is bootstrap aggregation or bagging. The general idea is to generate many predictors, each using regression or classification trees, and then forming a final prediction based on the average prediction of all these trees.To assure that the individual trees are not the same, we use the bootstrap to induce randomness. Bagging or bootstrapping is very useful in our scenario as we have an imbalanced dataset.

```
train_rf <- randomForest(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
    relationship + education, data = temp)

accuracy_rf <- confusionMatrix(predict(train_rf, adultpayclean_validation),
↪   as.factor(adultpayclean_validation$income ==
    "Above50K"))$overall["Accuracy"]


accuracy_rf

# Accuracy
```

```r
# 0.8218

f1_rf

# [1] 0.88556

# Area under the curve: 0.81804

varImp(train_rf)
```

```
Variable Importance

Overall

age           696.37032
eduyears      606.59083
sex           114.11631
race           77.66081
hoursperweek  484.89859
maritalstatus 923.10055
relationship  991.96874
education     652.20573


 Confusion Matrix and Statistics

           Reference
 Prediction FALSE TRUE
      FALSE  2012  332
      TRUE    188  38

                Accuracy : 0.8218
                  95% CI : (0.8074, 0.8355)
     No Information Rate : 0.7539
     P-Value [Acc > NIR] : < 2.2e-16
                   Kappa : 0.4849
  Mcnemar's Test P-Value : 3.588e-10
             Sensitivity : 0.9145
             Specificity : 0.5376
          Pos Pred Value : 0.8584
          Neg Pred Value : 0.6725
              Prevalence : 0.7539
          Detection Rate : 0.6895
    Detection Prevalence : 0.8033
       Balanced Accuracy : 0.7261
        'Positive' Class : FALSE


 With "Occupation" as one of the feature

 Confusion Matrix and Statistics

          Reference
```

```
Prediction FALSE TRUE
     FALSE  1990  293
     TRUE    210  425

               Accuracy : 0.8276
                 95% CI : (0.8134, 0.8412)
    No Information Rate : 0.7539
    P-Value [Acc > NIR] : < 2.2e-16
                  Kappa : 0.5166
 Mcnemar's Test P-Value : 0.000256
            Sensitivity : 0.9045
            Specificity : 0.5919
         Pos Pred Value : 0.8717
         Neg Pred Value : 0.6693
             Prevalence : 0.7539
         Detection Rate : 0.6820
   Detection Prevalence : 0.7824
      Balanced Accuracy : 0.7482
       'Positive' Class : FALSE
```

The accuracy of random forest is at 0.8218, sensitivity is at 0.9145 and specificity is at 0.5376. Prevalence is about the same when compared to knn and rpart classification models. The confidence intervals is about the same as rpart. The F1 score is 0.88556 is the best so far. AUC also on the positive side. Due to the randomization of features during the random forest bootstrapping, its hard to know if all the features will be used. Fortunately, we can investigate into how often a specific feature is used in the predictions using variable importance

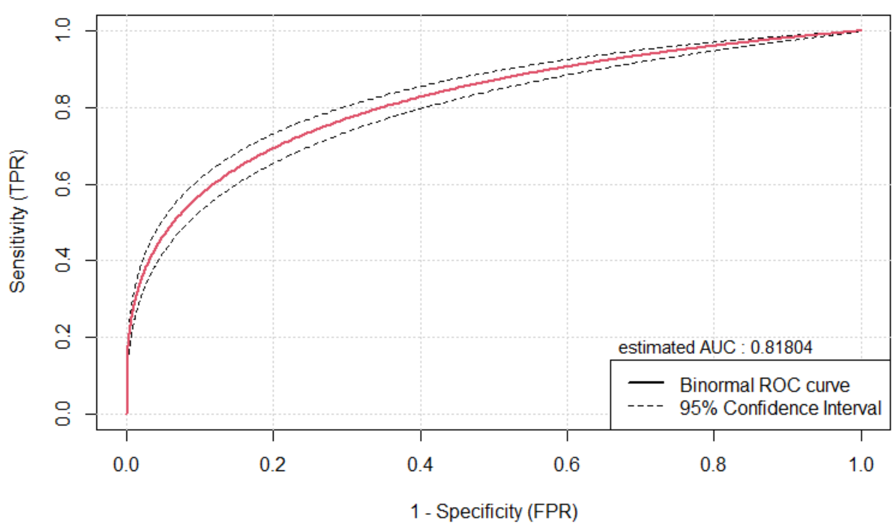


Figure 33: random forest classification summary

The out-of-bag errors and errors for Above-50K and AtBelow50K classes are shown in the classification errors figure. You can see the errors reduce as the number of trees are added and then plateau after a while. The errors are around 16% and are within acceptable limits.

When we add "Occupation" as one the feature (with some missing occupation data (5%)) we see that the accuracy is 0.8276 and sensitivity and specificity are 0.9045 and 0.5919 respectively with a F1 score of 0.88779.

When we add "class" as one the feature (with some missing occupation data (5%)) we see that the accuracy is 0.8283 and sensitivity and specificity are 0.9077 and 0.5850 respectively with a F1 score of 0.88854.

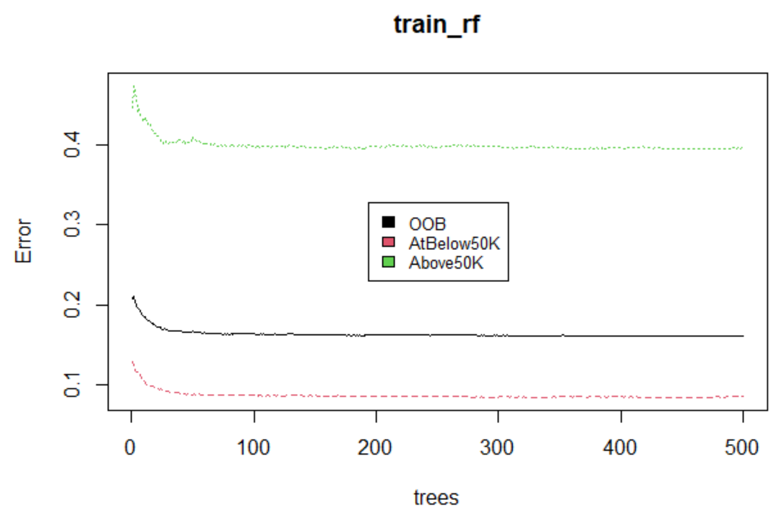


Figure 34: random forest classification errors

We can see the variable importance with occupation and class.

```
varImp(train_rf)

Overall

age             1296.5653
eduyears         655.8616
sex              123.8927
race             143.0220
hoursperweek     770.9115
maritalstatus    863.0918
relationship    1192.2686
education        633.8376
occupation       905.2539
class            409.0696
```

We can see that age, years of education, relationship,occupation and maritalstatus are the most used and sex and race are the least used features. We can also see that occupation did have a greater impact on decisioning even though we had 5% unknown data.

Lets tune this model just like the KNN3 classification and see if we can do better.

```r
nodesize <- seq(1, 51, 10)

acc <- sapply(nodesize, function(ns) {
    train(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus + relationship,
        method = "rf", data = temp, tuneGrid = data.frame(mtry = 2), nodesize =
         ↪  ns)$results$Accuracy
```

```r
})

qplot(nodesize, acc)

train_rf_2 <- randomForest(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus
↪  +
    relationship, data = temp, nodesize = nodesize[which.max(acc)])

y_hat_rf2 <- predict(train_rf_2, adultpayclean_validation)

accuracy_rftune <- confusionMatrix(predict(train_rf_2, adultpayclean_validation),
    as.factor(adultpayclean_validation$income == "Above50K"))$overall["Accuracy"]


accuracy_rftune

# Accuracy

# 0.8221

f1_rf2

# [1] \t0.88596

# Area under the curve: 0.81917
```

```
 Confusion Matrix and Statistics

            Reference
  Prediction FALSE TRUE
       FALSE  2016  335
       TRUE    184  383

                Accuracy : 0.8221
                  95% CI : (0.8078, 0.8359)
     No Information Rate : 0.7539
     P-Value [Acc > NIR] : < 2.2e-16
                   Kappa : 0.4841
   Mcnemar's Test P-Value : 4.571e-11
             Sensitivity : 0.9164
             Specificity : 0.5334
          Pos Pred Value : 0.8575
          Neg Pred Value : 0.6755
              Prevalence : 0.7539
          Detection Rate : 0.6909
    Detection Prevalence : 0.8057
       Balanced Accuracy : 0.7249
        'Positive' Class : FALSE

  With "Occupation" and "class" features

  Confusion Matrix and Statistics
```

```
          Reference
Prediction FALSE TRUE
     FALSE  2025  293
     TRUE    175  425

               Accuracy : 0.83996
                 95% CI : (0.8261, 0.8531)
    No Information Rate : 0.7539
    P-Value [Acc > NIR] : < 2.2e-16
                  Kappa : 0.5441
 Mcnemar's Test P-Value : 1.325e-07
            Sensitivity : 0.9200
            Specificity : 0.5947
         Pos Pred Value : 0.8743
         Neg Pred Value : 0.7081
             Prevalence : 0.7539
         Detection Rate : 0.6936
   Detection Prevalence : 0.7934
      Balanced Accuracy : 0.7574
       'Positive' Class : FALSE
```

Variable Importance

```
varImp(train_rf_2)
```

	Overall
age	491.55809
eduyears	595.07607
sex	101.73626
race	49.67436
hoursperweek	340.57118
maritalstatus	839.06711
relationship	981.20678
education	598.69678

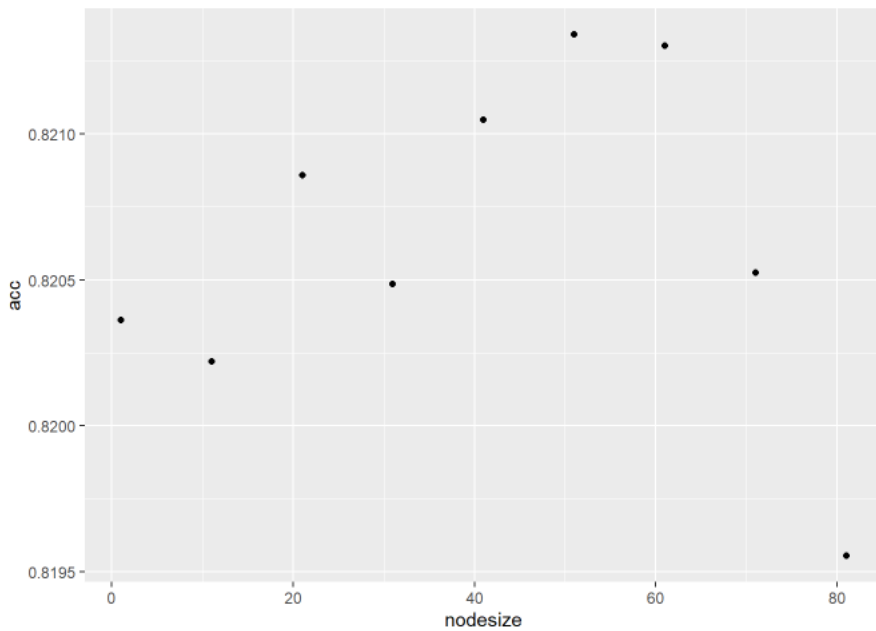


Figure 35: rf tuning

The accuracy of random forest with tuning is at 0.8221, sensitivity is at 0.916 and specificity is at 0.533. Prevalence is about the same when compared to knn and rpart classification models. The confidence intervals is just about the same as rpart and random forest without tuning.AUC improved slightly. In addition to the accuracy, precision increased to 0.8859.

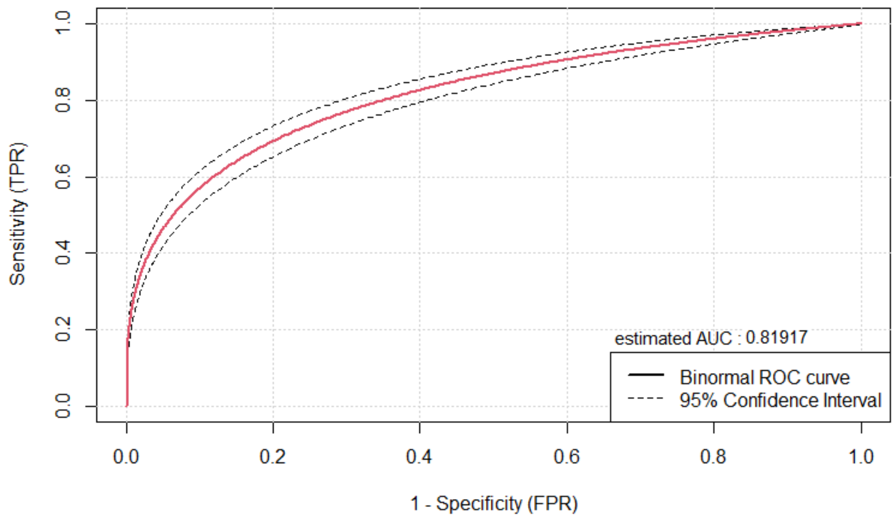


Figure 36: random forest tuned classification summary

When we added the experimental occupation and class data into the dataset for rf tuned test we see that the accuracy improves to 0.83996 and sensitivity is 0.9205 and specificity is 0.5947 with prevalence being the same as before and balanced accuracy improved to 0.7574. The F1 score for this is 0.896

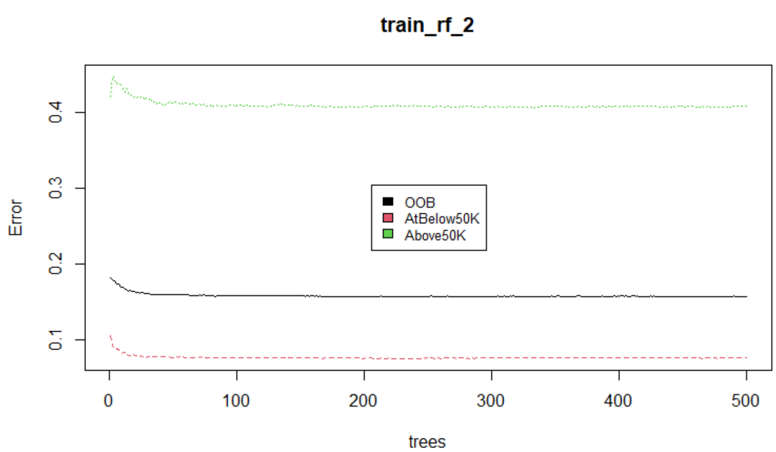


Figure 37: tuned random forest classification errors

The out-of-bag errors and errors for Above-50K and AtBelow50K classes are shown in the classification errors figure. You can see the errors reduce as the number of trees are added and then plateau after a while. The errors are around 16% and are within acceptable limits.

We can see the variable importance with occupation and class.

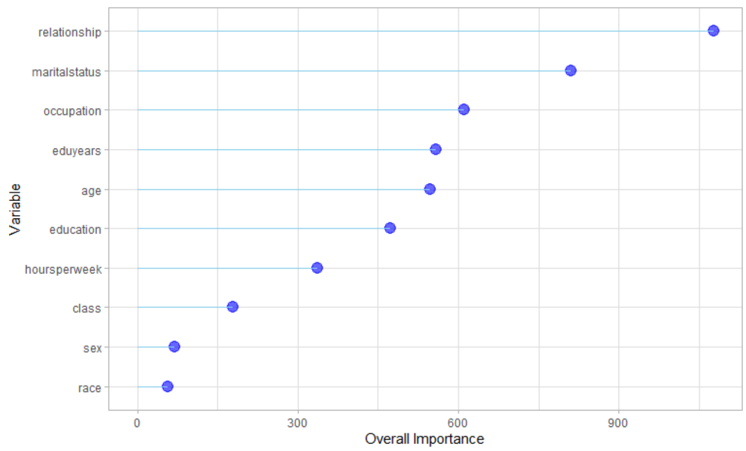


Figure 38: tuned random forest variable importance

When we compare our final model with glm, naive bayes, knn and rpart the random forest model took quite a bit of time to generate. Hence the cost of generation is high. However, on the flip side the accuracy, CI, precision is higher than other models. Prevalence plateaus once the accuracy and precision are high. The logistic and generative models like GLM, and Naive Bayes could return a faster response but sacrifices accuracy and precision and has a decent prevalence. Naive Bayes achieves a better accuracy than glm and has compatible sensitivity and specificity and lower prevalence when compared to random forest and knn. KNN is a mid-range model with speed faster than random forest but with lower accuracy and precision. It is slower than Naive Bayes and Glm.

If my objective were to get the outcomes faster with slightly lesser accuracy then I would have gone with Naive Bayes. However, if you look closer at the classification models, there is a high degree of imbalance in the dataset and naive bayes doesn't do very well when the dataset is imbalanced. F1 score of random forest is higher than naive bayes. For this reason I chose random forest over naive bayes. Further more as this is a Census pay prediction application, the frequency of runs would be lesser compared to applications like detecting if a person will get a heart attack in the next 24 hours. In such a scenario getting a faster prediction at a lower accuracy can be used to err on the safe side.

Results

This project created a machine learning model that predicts income level of adults based on 8 shortlisted attributes and 2 experimental attributes. This model was run on a test dataset that included an income level indicator, allowing us to compare the predicted and actual value.

We chose random forest as the final model after comparison. The model returned an accuracy of 82 to 83% while predicting annual income more than $50,000 annually. If occupation and class (with 5% unknown data) are included as features then accuracy changes to 83 to 84%. A summary table of results is given below:

	Method	Accuracy	Sensitivity	Specificity	Prevalence	F1
1.	Plain old guess	0.50822	0.50279	0.51	0.24606	0.33472
2.	linear model	0.81528	0.4429	0.93682	0.24606	0.54128
3.	General linear model	0.81049	0.38719	0.94864	0.24606	0.50135
4.	naive bayes	0.80089	0.91112	0.57218	0.67478	0.86064
5.	knn	0.80535	0.90545	0.49861	0.75394	0.87522
6.	knn tune	0.8098	0.90636	0.51114	0.75394	0.87745
7.	rpart	0.82111	0.91	0.54875	0.75394	0.88467
8.	rf	0.8218	0.91455	0.5376	0.75394	0.88556
9.	rf tune	0.83996	0.92	0.59471	0.75394	0.89657

Figure 39: Final Results

The accuracy of the predictions is backed by sensitivity of 0.9163 and specificity of 0.540. The F1 score 0.8869 vouches for the sensitivity and specificity. AUC is around 0.818 and is considered as acceptable considering some degree of imbalance in the dataset.

This model can be used to determine income levels for adults in any year with similar attribute sets and achieve comparable accuracies. The following attributes were determined to influence annual adult incomes in the US:

- age
- eduyears
- education
- sex
- race
- hoursperweek
- maritalstatus
- relationship
- occupation and class (experimental with 5% unknown data each)

Conclusion

This machine learning model was able to predict annual incomes of persons in US based (1994) on 8 parameters with an accuracy of 82 to 83%. This model can be applied to data from other census years as well. The model will perform better if the training set is updated with new data that is confirmed for correctness, that is, the label value is the real life value, and not the predicted value. Additional data for certain features like marital status, occupation, and class will make the model better. The model can continuously learn from changing data in the training set to adapt to new parameters, thus improving its accuracy and other metrics

Further improvements could be made by using random oversampling and/or under-sampling techniques to fill the void created by imbalanced datasets.

Another approach for improving accuracy would be to create an ensemble based on multiple algorithms like rpart, random forest and glm.

Additionally, we could also use advanced algorithms like Ada Boost Gradient Boost Trees, support Vector Machines, Neural Networks, and Deep Learning to improve the predicted outcomes.

Appendix A - Complete code

```r
## @knitr CensusPayR

# Note: This script will take a while to run. In particular the knn and random
# forest algorithms with tuning grids will take more time. please be patient if
# you happen to execute it. The execution report is available in the github
# location as well

# Execute the given source code for the project
source("DatasetProcessingCode.R")

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

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

if (!require(e1071)) install.packages("e1071")

if (!require(pROC)) install.packages("pRoc")

if (!require(ROCit)) install.packages("ROCit")

library(caret)
library(gridExtra)
library(kableExtra)
library(randomForest)
library(purrr)
library(e1071)
library(caTools)
library(pROC)
library(ROCit)

# set the seed for reproducible results
set.seed(2008, sample.kind = "Rounding")

# the simplest possible machine algorithm: guessing the outcome
seat_of_the_pants <- sample(c("Above50K", "AtBelow50K"), length(test_index), replace =
↪  TRUE) %>%
    factor(levels = levels(adultpayclean_validation$income))
# calculate the accuracy of this sampling
accuracy_guess <- mean(seat_of_the_pants == adultpayclean_validation$income)

# build a confusion matrix for this simple model
table(predicted = seat_of_the_pants, actual = adultpayclean_validation$income)

# tabulate accuracy by income levels
adultpayclean_validation %>%
    mutate(y_hat = seat_of_the_pants) %>%
```

```
    group_by(income) %>%
    summarize(accuracy = mean(y_hat == income))

# confusion matrix using R function
cm <- confusionMatrix(data = seat_of_the_pants, reference =
↪  adultpayclean_validation$income)
# display the confusion matrix
cm

# record the sensitivity, specificity, and prevalence
sensitivity_guess <- cm$byClass[["Sensitivity"]]
specificity_guess <- cm$byClass[["Specificity"]]
prevalence_guess <- cm$byClass[["Prevalence"]]
f1_guess <- cm$byClass[["F1"]]

# find the area under the curve/ROC
auc(ifelse(adultpayclean_validation$income == "Above50K", 1, 0), ifelse(seat_of_the_pants
↪  ==
    "Above50K", 1, 0))

set.seed(2008)
# logistic linear model create the model
lm_fit <- adultpayclean_train %>%
    mutate(y = as.numeric(income == "Above50K")) %>%
    lm(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus + relationship +
        education, data = .)

# predict using test set
p_hat_logit <- predict(lm_fit, newdata = adultpayclean_validation)

# translate predicted data into factor
y_hat_logit <- ifelse(p_hat_logit > 0.5, "Above50K", "AtBelow50K") %>%
    factor

# compare the predicted vs observed values and use confusionMatrix to get the
# accuracy and other metrics
cm_lm <- confusionMatrix(y_hat_logit, adultpayclean_validation$income)
accuracy_lm <- confusionMatrix(y_hat_logit,
↪  adultpayclean_validation$income)$overall[["Accuracy"]]

cm_lm

# record the sensitivity, specificity, and prevalence
sensitivity_lm <- cm_lm$byClass[["Sensitivity"]]
specificity_lm <- cm_lm$byClass[["Specificity"]]
prevalence_lm <- cm_lm$byClass[["Prevalence"]]
f1_lm <- cm_lm$byClass[["F1"]]

# Find the ROC and plot it. Show the AUC as well
pROC_bin <- ROCit::rocit(ifelse(adultpayclean_validation$income == "Above50K", 1,
    0), ifelse(unname(y_hat_logit) == "Above50K", 1, 0), method = "bin")
ciROC_bin95 <- ROCit::ciROC(pROC_bin, level = 0.95)
plot(ciROC_bin95, col = 1, values = TRUE)
```

```r
lines(ciROC_bin95$TPR ~ ciROC_bin95$FPR, col = 2, lwd = 2)
ROCit::ciAUC(pROC_bin)

set.seed(2008)
# general linear model create the glm model
glm_fit <- adultpayclean_train %>%
    mutate(y = as.numeric(income == "Above50K")) %>%
    glm(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus + relationship +
        education, data = ., family = "binomial")

# predict using validation set
p_hat_logit <- predict(glm_fit, newdata = adultpayclean_validation)

# translate the predicted data into factor
y_hat_logit <- ifelse(p_hat_logit > 0.5, "Above50K", "AtBelow50K") %>%
    factor

# compare the predicted vs observed values and use confusionMatrix to get the
# accuracy and other metrics for the glm model
cm_glm <- confusionMatrix(y_hat_logit, adultpayclean_validation$income)
accuracy_glm <- confusionMatrix(y_hat_logit,
↪  adultpayclean_validation$income)$overall[["Accuracy"]]

cm_glm

# record the sensitivity, specificity, and prevalence
sensitivity_glm <- cm_glm$byClass[["Sensitivity"]]
specificity_glm <- cm_glm$byClass[["Specificity"]]
prevalence_glm <- cm_glm$byClass[["Prevalence"]]
f1_glm <- cm_glm$byClass[["F1"]]

# Find the ROC and plot it. Show the AUC as well
pROC_bin <- ROCit::rocit(ifelse(adultpayclean_validation$income == "Above50K", 1,
    0), ifelse(unname(y_hat_logit) == "Above50K", 1, 0), method = "bin")
ciROC_bin95 <- ROCit::ciROC(pROC_bin, level = 0.95)
plot(ciROC_bin95, col = 1, values = TRUE)
lines(ciROC_bin95$TPR ~ ciROC_bin95$FPR, col = 2, lwd = 2)
ROCit::ciAUC(pROC_bin)

# Naive bayes
set.seed(2008)
# create the naive bayes model
train_nb <- adultpayclean_train %>%
    mutate(y = as.factor(income == "Above50K")) %>%
    naiveBayes(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
    ↪  relationship +
        education, data = .)

# predict using the validation dataset
y_hat_nb <- predict(train_nb, newdata = adultpayclean_validation)
# create the confusion matrix
cm_tab <- table(adultpayclean_validation$income == "Above50K", y_hat_nb)
cm_nb <- confusionMatrix(cm_tab)
```

```r
cm_nb

# get the accuracy, sensitivity, specificity, prevalence and, F1 score
accuracy_nb <- cm_nb$overall[["Accuracy"]]
sensitivity_nb <- cm_nb$byClass[["Sensitivity"]]
specificity_nb <- cm_nb$byClass[["Specificity"]]
prevalence_nb <- cm_nb$byClass[["Prevalence"]]
f1_nb <- cm_nb$byClass[["F1"]]

# Find the ROC and plot it. Show the AUC as well
pROC_bin <- ROCit::rocit(ifelse(adultpayclean_validation$income == "Above50K", 1,
    0), ifelse(unname(y_hat_nb) == "TRUE", 1, 0), method = "bin")
ciROC_bin95 <- ROCit::ciROC(pROC_bin, level = 0.95)
plot(ciROC_bin95, col = 1, values = TRUE)
lines(ciROC_bin95$TPR ~ ciROC_bin95$FPR, col = 2, lwd = 2)
ROCit::ciAUC(pROC_bin)

# translate income factor into binary outcome
temp <- adultpayclean_train %>%
    mutate(y = as.factor(income == "Above50K"))

# k-nearest neighbors with a train control and tuning
set.seed(2008)
# train control to use 10% of the observations each to speed up computations
control <- trainControl(method = "cv", number = 10, p = 0.9)
# train the model using knn. choose the best k value using tuning algorithm
train_knn <- train(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
    relationship + education, method = "knn", data = temp, tuneGrid = data.frame(k =
↪   seq(3,
    71, 2)), trControl = control)

# plot the resulting model
ggplot(train_knn, highlight = TRUE)
# verify which k value was used
train_knn$bestTune
train_knn$finalModel

# use this trained model to predict raw knn predictions
y_hat_knn <- predict(train_knn, adultpayclean_validation, type = "raw")

# compare the predicted and observed values using confusionMatrix to get the
# accuracy and other metrics
cm_knn <- confusionMatrix(y_hat_knn, as.factor(adultpayclean_validation$income ==
    "Above50K"))
accuracy_knn <- confusionMatrix(y_hat_knn, as.factor(adultpayclean_validation$income ==
    "Above50K"))$overall[["Accuracy"]]

cm_knn

# record the sensitivity, specificity, and prevalence
sensitivity_knn <- cm_knn$byClass[["Sensitivity"]]
specificity_knn <- cm_knn$byClass[["Specificity"]]
prevalence_knn <- cm_knn$byClass[["Prevalence"]]
```

```r
f1_knn <- cm_knn$byClass[["F1"]]

# Find the ROC and plot it. Show the AUC as well
pROC_bin <- ROCit::rocit(ifelse(adultpayclean_validation$income == "Above50K", 1,
    0), ifelse(unname(y_hat_knn) == "TRUE", 1, 0), method = "bin")
ciROC_bin95 <- ROCit::ciROC(pROC_bin, level = 0.95)
plot(ciROC_bin95, col = 1, values = TRUE)
lines(ciROC_bin95$TPR ~ ciROC_bin95$FPR, col = 2, lwd = 2)
ROCit::ciAUC(pROC_bin)



# k-nearest classification using tuning function
set.seed(2008)

# train the model using knn3 classification
ks <- seq(3, 251, 2)
knntune <- map_df(ks, function(k) {
    temp <- adultpayclean_train %>%
        mutate(y = as.factor(income == "Above50K"))
    temp_test <- adultpayclean_validation %>%
        mutate(y = as.factor(income == "Above50K"))
    # create the kkn3 model
    knn_fit <- knn3(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
        relationship + education, data = temp, k = k)
    # predict the model for the current k
    y_hat <- predict(knn_fit, temp, type = "class")
    # get the confusionmatrix for the current k
    cm_train <- confusionMatrix(y_hat, temp$y)
    train_error <- cm_train$overall["Accuracy"]
    # do the same for test model
    y_hat <- predict(knn_fit, temp_test, type = "class")
    cm_test <- confusionMatrix(y_hat, temp_test$y)
    test_error <- cm_test$overall["Accuracy"]

    tibble(train = train_error, test = test_error)
})
# get the accuracy for the k with maximum accuracy
accuracy_knntune <- max(knntune$test)
# get the confusion matrix for that k
knn_fit <- knn3(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
    relationship + education, data = temp, k = ks[which.max(knntune$test)])
# predict the knn tune using the model for the k neighbor
y_hat_knntune <- predict(knn_fit, adultpayclean_validation, type = "class")
cm_knntune <- confusionMatrix(y_hat_knntune, as.factor(adultpayclean_validation$income ==
    "Above50K"))

cm_knntune

# record the sensitivity, specificity, and prevalence
sensitivity_knntune <- cm_knntune$byClass[["Sensitivity"]]
specificity_knntune <- cm_knntune$byClass[["Specificity"]]
prevalence_knntune <- cm_knntune$byClass[["Prevalence"]]
```

```r
f1_knntune <- cm_knntune$byClass[["F1"]]

# Find the ROC and plot it. Show the AUC as well
pROC_bin <- ROCit::rocit(ifelse(adultpayclean_validation$income == "Above50K", 1,
    0), ifelse(unname(y_hat_knntune) == "TRUE", 1, 0), method = "bin")
ciROC_bin95 <- ROCit::ciROC(pROC_bin, level = 0.95)
plot(ciROC_bin95, col = 1, values = TRUE)
lines(ciROC_bin95$TPR ~ ciROC_bin95$FPR, col = 2, lwd = 2)
ROCit::ciAUC(pROC_bin)

# recursive partitioning using rpart
set.seed(2008)
# train the model with the recursive partitioning
train_rpart <- train(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
    relationship + education, method = "rpart", tuneGrid = data.frame(cp = seq(0,
    0.1, len = 25)), data = temp)
# predict the outcomes with this model
y_hat_rpart <- predict(train_rpart, adultpayclean_validation)
# confusion matrix for the rpart model
cm_rpart <- confusionMatrix(y_hat_rpart, as.factor(adultpayclean_validation$income ==
    "Above50K"))
# get the accuracy
accuracy_rpart <- confusionMatrix(y_hat_rpart, as.factor(adultpayclean_validation$income
↪   ==
    "Above50K"))$overall["Accuracy"]

cm_rpart
# record the sensitivity, specificity, and prevalence
sensitivity_rpart <- cm_rpart$byClass[["Sensitivity"]]
specificity_rpart <- cm_rpart$byClass[["Specificity"]]
prevalence_rpart <- cm_rpart$byClass[["Prevalence"]]
f1_rpart <- cm_rpart$byClass[["F1"]]

# Find the ROC and plot it. Show the AUC as well
pROC_bin <- ROCit::rocit(ifelse(adultpayclean_validation$income == "Above50K", 1,
    0), ifelse(unname(y_hat_rpart) == "TRUE", 1, 0), method = "bin")
ciROC_bin95 <- ROCit::ciROC(pROC_bin, level = 0.95)
plot(ciROC_bin95, col = 1, values = TRUE)
lines(ciROC_bin95$TPR ~ ciROC_bin95$FPR, col = 2, lwd = 2)
ROCit::ciAUC(pROC_bin)

# random forest
set.seed(2008)
# train the vanilla random forest model
train_rf <- randomForest(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
    relationship + education, data = temp)

y_hat_rf <- predict(train_rf, adultpayclean_validation)

# create the confusionMatrix
cm_rf <- confusionMatrix(y_hat_rf, as.factor(adultpayclean_validation$income ==
↪   "Above50K"))
# get the accuracy
```

```
accuracy_rf <- confusionMatrix(y_hat_rf, as.factor(adultpayclean_validation$income ==
    "Above50K"))$overall["Accuracy"]

cm_rf

# record the sensitivity, specificity, and prevalence
sensitivity_rf <- cm_rf$byClass[["Sensitivity"]]
specificity_rf <- cm_rf$byClass[["Specificity"]]
prevalence_rf <- cm_rf$byClass[["Prevalence"]]
f1_rf <- cm_rf$byClass[["F1"]]

# Find the ROC and plot it. Show the AUC as well
pROC_bin <- ROCit::rocit(ifelse(adultpayclean_validation$income == "Above50K", 1,
    0), ifelse(unname(y_hat_rf) == "TRUE", 1, 0), method = "bin")
ciROC_bin95 <- ROCit::ciROC(pROC_bin, level = 0.95)
plot(ciROC_bin95, col = 1, values = TRUE)
lines(ciROC_bin95$TPR ~ ciROC_bin95$FPR, col = 2, lwd = 2)
ROCit::ciAUC(pROC_bin)

# Plot the error rate chart for the random forest
plot(train_rf)
legend("center", ifelse(colnames(train_rf$err.rate) == "FALSE", "AtBelow50K",
↪  ifelse(colnames(train_rf$err.rate) ==
    "TRUE", "Above50K", "OOB")), col = 1:4, cex = 0.8, fill = 1:4)

set.seed(2008)
# random forest with tuning
nodesize <- seq(1, 90, 10)
acc <- sapply(nodesize, function(ns) {
    # train the model with tuning
    train(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus + relationship +
        education, method = "rf", data = temp, tuneGrid = data.frame(mtry = 2), nodesize
↪  = ns)$results$Accuracy
})
qplot(nodesize, acc)

set.seed(2008)
# get the trained model for the max node size
train_rf_2 <- randomForest(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus
↪  +
    relationship + education, data = temp, nodesize = nodesize[which.max(acc)])
# predict the outcomes
y_hat_rf2 <- predict(train_rf_2, adultpayclean_validation)
# get the confusion matrix for random forest model
cm_rf2 <- confusionMatrix(y_hat_rf2, as.factor(adultpayclean_validation$income ==
    "Above50K"))
# get the accuracy
accuracy_rftune <- confusionMatrix(y_hat_rf2, as.factor(adultpayclean_validation$income
↪  ==
    "Above50K"))$overall["Accuracy"]

cm_rf2
```

```
# record the sensitivity, specificity, and prevalence
sensitivity_rf2 <- cm_rf2$byClass[["Sensitivity"]]
specificity_rf2 <- cm_rf2$byClass[["Specificity"]]
prevalence_rf2 <- cm_rf2$byClass[["Prevalence"]]
f1_rf2 <- cm_rf2$byClass[["F1"]]

# Find the ROC and plot it. Show the AUC as well
pROC_bin <- ROCit::rocit(ifelse(adultpayclean_validation$income == "Above50K", 1,
    0), ifelse(unname(y_hat_rf2) == "TRUE", 1, 0), method = "bin")
ciROC_bin95 <- ROCit::ciROC(pROC_bin, level = 0.95)
plot(ciROC_bin95, col = 1, values = TRUE)
lines(ciROC_bin95$TPR ~ ciROC_bin95$FPR, col = 2, lwd = 2)
ROCit::ciAUC(pROC_bin)

# Plot the error rate chart for the random forest
plot(train_rf_2)
legend("center", ifelse(colnames(train_rf_2$err.rate) == "FALSE", "AtBelow50K",
↪  ifelse(colnames(train_rf_2$err.rate) ==
    "TRUE", "Above50K", "OOB")), col = 1:4, cex = 0.8, fill = 1:4)

# tabulate all the accuracy results with sensitivity and specificity
accuracy_results <- matrix(c("Plain old guess", round(accuracy_guess, 5),
↪  round(sensitivity_guess,
    5), round(specificity_guess, 5), round(prevalence_guess, 5), round(f1_guess,
    5), "linear model", round(accuracy_lm, 5), round(sensitivity_lm, 5),
    ↪  round(specificity_lm,
    5), round(prevalence_lm, 5), round(f1_lm, 5), "General linear model",
    ↪  round(accuracy_glm,
    5), round(sensitivity_glm, 5), round(specificity_glm, 5), round(prevalence_glm,
    5), round(f1_glm, 5), "naive bayes", round(accuracy_nb, 5), round(sensitivity_nb,
    5), round(specificity_nb, 5), round(prevalence_nb, 5), round(f1_nb, 5), "knn",
    round(accuracy_knn, 5), round(sensitivity_knn, 5), round(specificity_knn, 5),
    round(prevalence_knn, 5), round(f1_knn, 5), "knn tune", round(accuracy_knntune,
        5), round(sensitivity_knntune, 5), round(specificity_knntune, 5),
        ↪  round(prevalence_knntune,
        5), round(f1_knntune, 5), "rpart", round(accuracy_rpart, 5),
        ↪  round(sensitivity_rpart,
        5), round(specificity_rpart, 5), round(prevalence_rpart, 5), round(f1_rpart,
        5), "rf", round(accuracy_rf, 5), round(sensitivity_rf, 5), round(specificity_rf,
        5), round(prevalence_rf, 5), round(f1_rf, 5), "rf tune", round(accuracy_rftune,
        5), round(sensitivity_rf2, 5), round(specificity_rf2, 5), round(prevalence_rf2,
        5), round(f1_rf2, 5)), nrow = 9, ncol = 6, byrow = TRUE, dimnames = list(c("1.",
    "2.", "3.", "4.", "5.", "6.", "7.", "8.", "9."), c("Method", "Accuracy",
    ↪  "Sensitivity",
    "Specificity", "Prevalence", "F1")))
# style the table with knitr
accuracy_results %>%
    knitr::kable() %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Appendix B - Code Execution

The code in this project takes long time to execute. Please find the execution summary at the link below.

https://github.com/rajeshharidas/havardxwork2/blob/main/CensusPayExecutionSummary.pdf

https://github.com/rajeshharidas/havardxwork2/blob/main/DatasetProcessingCode.pdf

https://github.com/rajeshharidas/havardxwork2/blob/main/CensusPayExecutionSummary.html

https://github.com/rajeshharidas/havardxwork2/blob/main/DatasetProcessingCode.html

Appendix C - Links

https://www.edx.org/professional-certificate/harvardx-data-science–

https://www.crcpress.com/Introduction-to-Data-Science-Data-Analysis-and-Prediction-Algorithms-with/Irizarry/p/book/9780367357986–

https://leanpub.com/datasciencebook–

Citations

Irizarry, Rafael A., "Introduction to Data Science: Data Analysis and Prediction Algorithms in R" https://rafalab.github.io/dsbook/

ML-Friendly kaggle dataset for adult census income - https://www.kaggle.com/uciml/adult-census-income