Census pay predictions

Rajesh Haridas

December 15, 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

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

```
# glimpse(adultpay)

# Rows: 32,561
```

```
# Columns: 13

# $ age <int> 90, 82, 66, 54, 41, 34, 38, 74, 68, 41, 45, 38, 52, 32, 51, 46,~

# $ workclass <chr> '?', 'Private', '?', 'Private', 'Private', 'Private',
# 'Private'~

# $ fnlwgt <int> 77053, 132870, 186061, 140359, 264663, 216864, 150601, 88638,
# 4~

# $ education <chr> 'HS-grad', 'HS-grad', 'Some-college', '7th-8th',
# 'Some-college'~

# $ education.num <int> 9, 9, 10, 4, 10, 9, 6, 16, 9, 10, 16, 15, 13, 14, 16,
# 15, 7, 14~

# $ marital.status <chr> 'Widowed', 'Widowed', 'Widowed', 'Divorced',
# 'Separated', 'Divo~

# $ occupation <chr> '?', 'Exec-managerial', '?', 'Machine-op-inspct',
# 'Prof-special~

# $ relationship <chr> 'Not-in-family', 'Not-in-family', 'Unmarried',
# 'Unmarried', 'Ow~

# $ race <chr> 'White', 'White', 'Black', 'White', 'White', 'White', 'White', ~

# $ sex <chr> 'Female', 'Female', 'Female', 'Female', 'Female', 'Female', 'Ma~

# $ hours.per.week <int> 40, 18, 40, 40, 40, 45, 40, 20, 40, 60, 35, 45, 20,
# 55, 40, 40,~

# $ native.country <chr> 'United-States', 'United-States', 'United-States',
# 'United-Stat~

# $ income <chr> '<=50K', '<=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 one of these. 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 be the outcome and the rest 12 attributes being the predictors or features. Looking at 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, knn, random forest with some tuning.

```
# > dim(adultpayclean)

# [1] 29170 13
```

```
# > dim(adultpayclean_train)

# [1] 26252 13

# > dim(adultpayclean_validation)

# [1] 2918 13
```

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 is sufficient degree of variability.

```
# summary(adultpayclean)

# age fnlwgt education eduyears maritalstatus

# Min.  :17.00 Min.  : 12285 HSgrad :9702 Min.  : 1.00 Divorced : 4162

# 1st Qu.:28.00 1st Qu.: 115895 Somecollege:6740 1st Qu.: 9.00 MarriedAFspouse
# : 23

# Median :37.00 Median : 176730 Bachelors :4766 Median :10.00 Marriedcivspouse
# :13368

# Mean :38.66 Mean : 187069 Masters :1527 Mean :10.17 Marriedspouseabsent: 253

# 3rd Qu.:48.00 3rd Qu.: 234139 Assocvoc :1289 3rd Qu.:12.00 Nevermarried :
# 9579

# Max.  :90.00 Max.  :1484705 11th :1067 Max.  :16.00 Separated : 883

# (Other) :4079 Widowed : 902

# occupation relationship race sex hoursperweek

# Execmanagerial:3735 Husband :11861 Amer-Indian-Eskimo: 296 Female: 9682 Min.
# : 1.00

# Profspecialty :3693 Notinfamily : 7528 Asian-Pac-Islander: 292 Male :19488
# 1st Qu.:40.00

# Craftrepair :3685 Otherrelative: 696 Black : 2832 Median :40.00

# Admclerical :3449 Ownchild : 4691 Other : 129 Mean :40.45

# Sales :3364 Unmarried : 3033 White :25621 3rd Qu.:45.00

# Otherservice :2777 Wife : 1361 Max.  :99.00

# (Other) :8467

# native income class
```

```
# Length:29170 Above50K : 7171 Private :20135

# Class :character AtBelow50K:21999 Selfempnotinc: 2313

# Mode :character Localgov : 1956

# Unknown : 1659

# Stategov : 1210

# Selfempinc : 991

# (Other) : 906
```

We can also see the number or above 50K income is significantly less than (30%) the at or below 50K income. This implies that we will encounter some prevalence towards the at or below 50K group

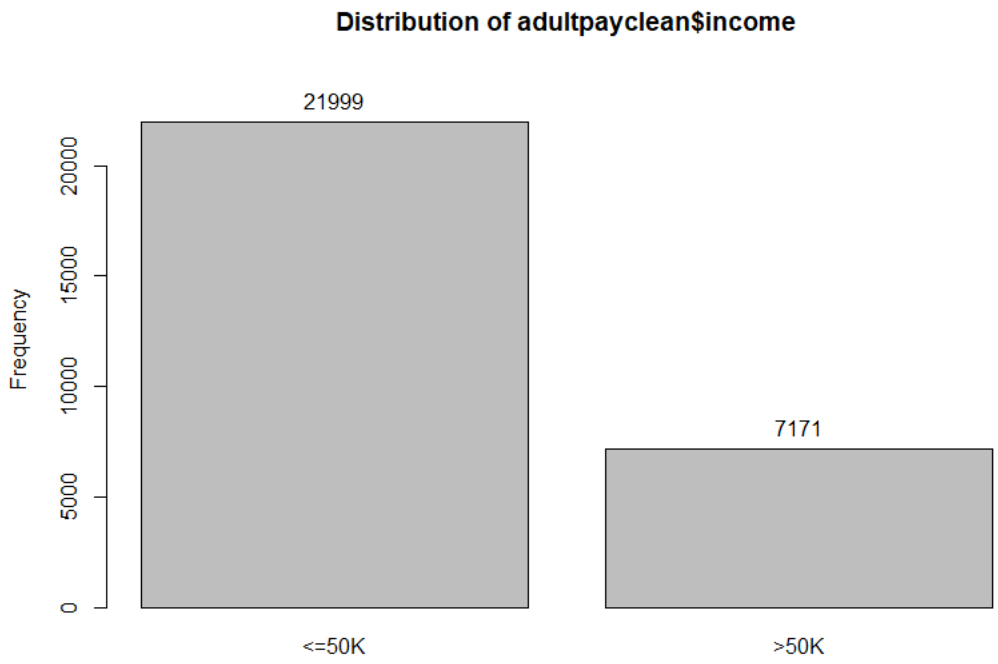


Figure 1: Income distribution

So we begin...

Analysis

Data Exploration, cleaning, 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
3) Remove intermediate columns and columns that are not used in the project
4) Rename columns to exclude non-alphanumeric characters
5) convert character labels to factors

```
# adultpayclean <-

# adultpay %>% filter (native.country == 'United-States') %>%

# mutate (class = ifelse(workclass == '?', 'Unknown',
# str_replace_all(workclass, '-', ''))) %>%

# 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 == '?', 'Unknown',
# str_replace_all(maritalstatus, '-', '') )) %>%

# mutate (occupation = ifelse( occupation == '?', 'Unknown',
# str_replace_all(occupation, '-', '') )) %>%

# mutate (education = ifelse(education == '?', 'Unknown',
# str_replace_all(education, '-', ''))) %>%

# mutate (relationship = ifelse( relationship == '?', 'Unknown',
# str_replace_all(relationship, '-', '') )) %>%

# mutate (native = ifelse(native == '?', 'Unknown', str_replace_all(native,
# '-', ''))) %>%

# mutate (income = ifelse( income == '?', 'Unknown', 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

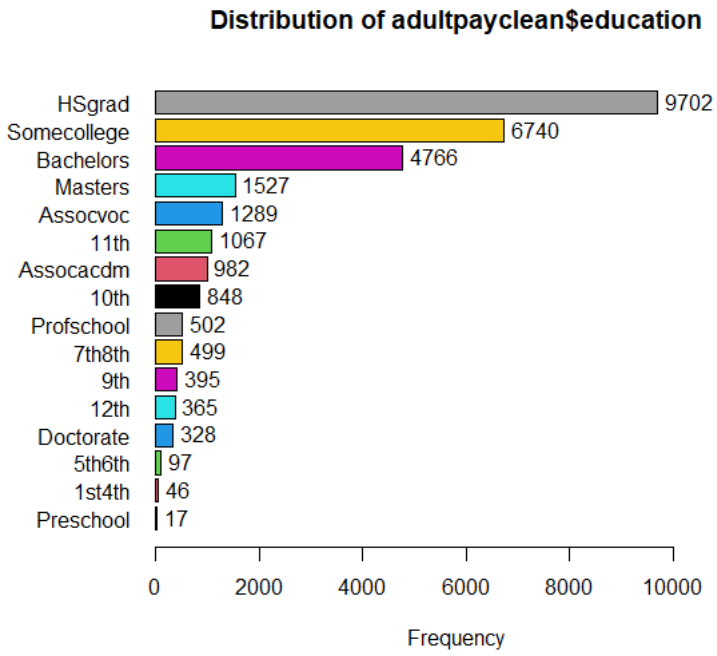


Figure 2: Education distribution

	Frequency	Percent	Cum. percent
HSgrad	9702	33.3	33.3
Somecollege	6740	23.1	56.4
Bachelors	4766	16.3	72.7
Masters	1527	5.2	77.9
Assocvoc	1289	4.4	82.4
11th	1067	3.7	86.0
Assocacdm	982	3.4	89.4
10th	848	2.9	92.3
Profschool	502	1.7	94.0
7th8th	499	1.7	95.7
9th	395	1.4	97.1
12th	365	1.3	98.3
Doctorate	328	1.1	99.5
5th6th	97	0.3	99.8
1st4th	46	0.2	99.9
Preschool	17	0.1	100.0
Total	29170	100.0	100.0

Figure 3: Education distribution

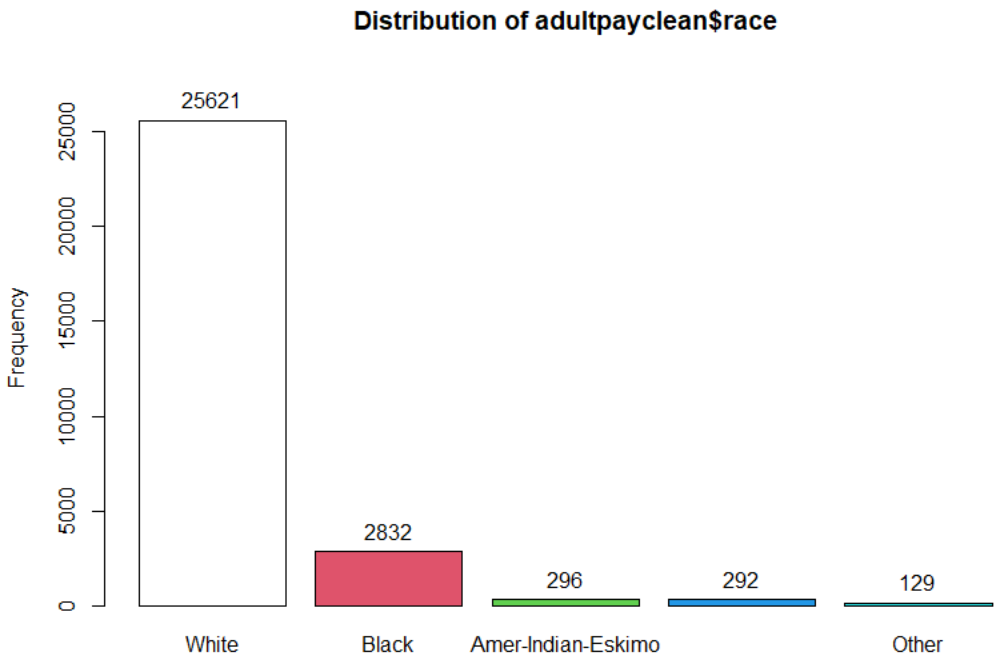


Figure 4: Race distribution

	Frequency	Percent	Cum. percent
White	25621	87.8	87.8
Black	2832	9.7	97.5
Amer-Indian-Eskimo	296	1.0	98.6
Asian-Pac-Islander	292	1.0	99.6
Other	129	0.4	100.0
Total	29170	100.0	100.0

Figure 5: Race distribution

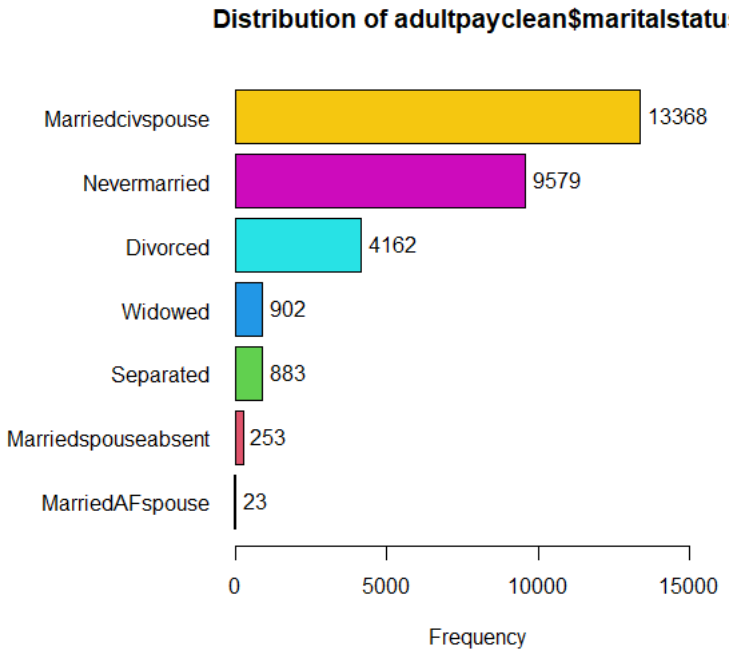


Figure 6: Marital Status distribution

	Frequency	Percent	Cum. percent
Marriedcivspouse	13368	45.8	45.8
Nevermarried	9579	32.8	78.7
Divorced	4162	14.3	92.9
Widowed	902	3.1	96.0
Separated	883	3.0	99.1
Marriedspouseabsent	253	0.9	99.9
MarriedAFspouse	23	0.1	100.0
Total	29170	100.0	100.0

Figure 7: Marital Status distribution

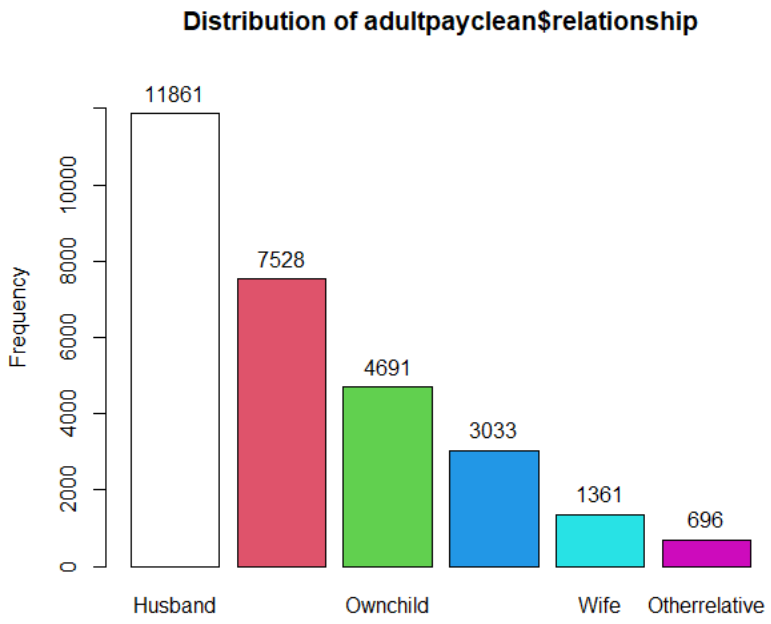


Figure 8: Relationship distribution

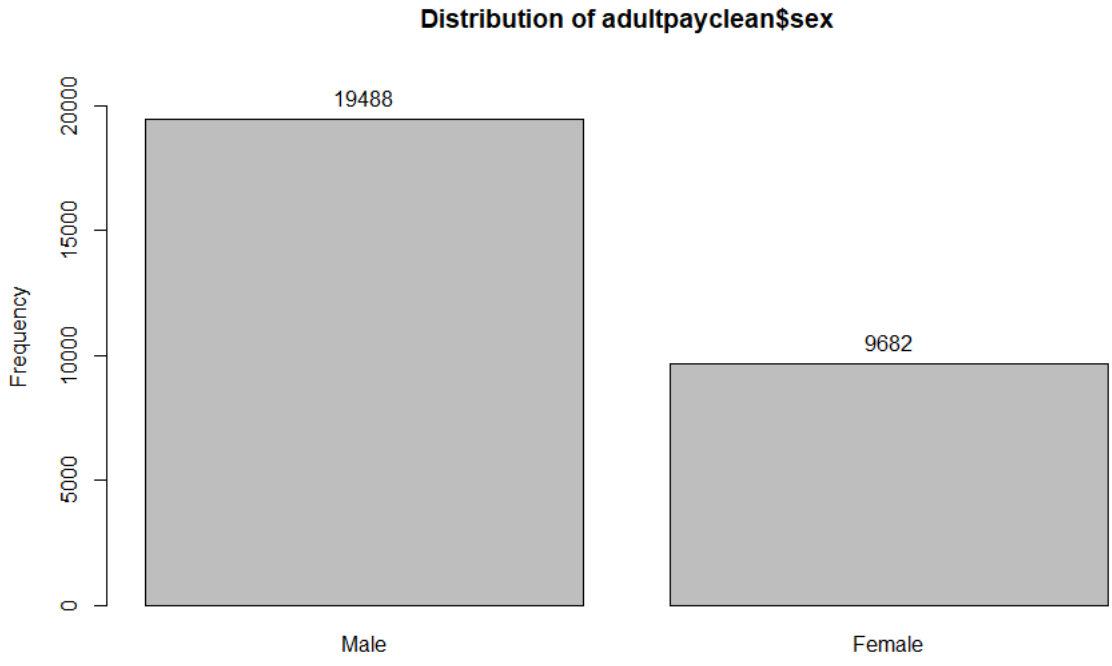


Figure 9: Sex distribution

	Frequency	Percent	Cum. percent
Male	19488	66.8	66.8
Female	9682	33.2	100.0
Total	29170	100.0	100.0

Figure 10: Sex distribution

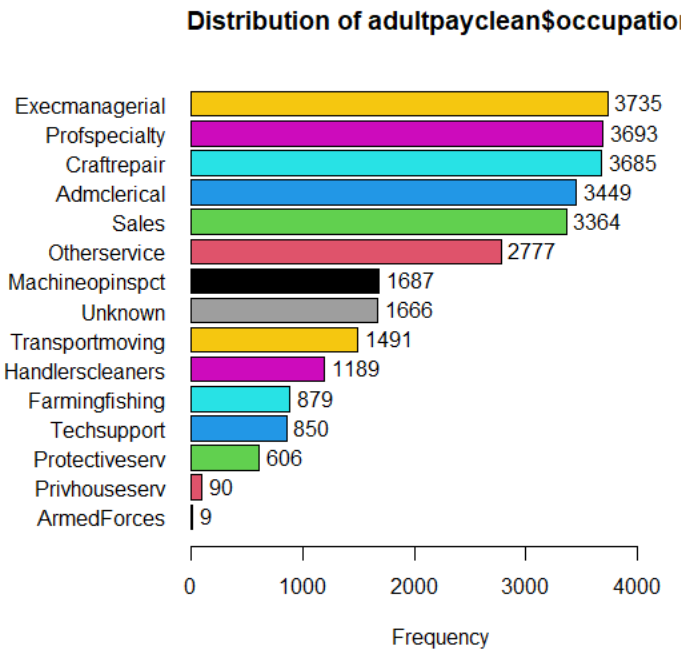


Figure 11: Occupation distribution

	Frequency	Percent	Cum. percent
Execmanagerial	3735	12.8	12.8
Profspecialty	3693	12.7	25.5
Craftrepair	3685	12.6	38.1
Admclerical	3449	11.8	49.9
Sales	3364	11.5	61.5
Otherservice	2777	9.5	71.0
Machineopinspct	1687	5.8	76.8
Unknown	1666	5.7	82.5
Transportmoving	1491	5.1	87.6
Handlerscleaners	1189	4.1	91.7
Farmingfishing	879	3.0	94.7
Techsupport	850	2.9	97.6
Protectiveserv	606	2.1	99.7
Privhouseserv	90	0.3	100.0
ArmedForces	9	0.0	100.0
Total	29170	100.0	100.0

Figure 12: Occupation distribution

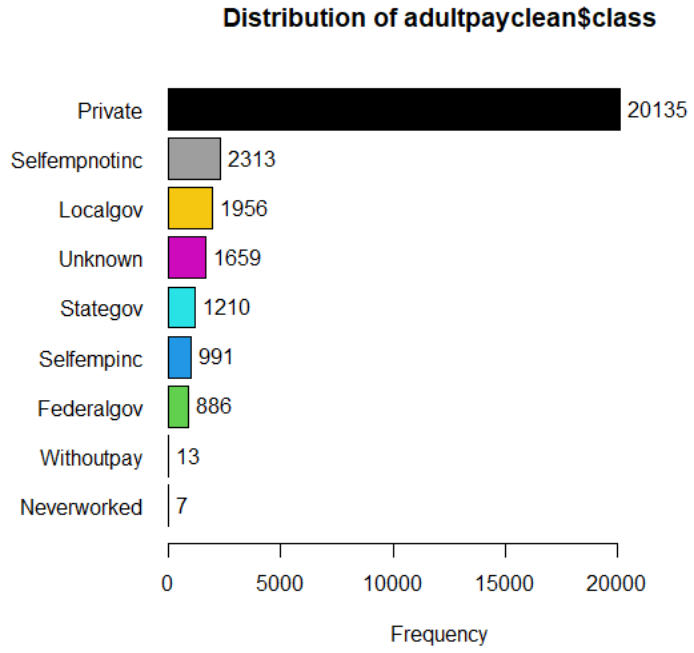


Figure 13: Class distribution

	Frequency	Percent	Cum. percent
Private	20135	69.0	69.0
Selfempnotinc	2313	7.9	77.0
Localgov	1956	6.7	83.7
Unknown	1659	5.7	89.3
Stategov	1210	4.1	93.5
Selfempinc	991	3.4	96.9
Federalgov	886	3.0	99.9
Withoutpay	13	0.0	100.0
Neverworked	7	0.0	100.0
Total	29170	100.0	100.0

Figure 14: Class distribution

As shown above the occupation and class columns have more than 1600 Unknown data. Adding this to the predictor columns could skew the predictions and introduce errors. We will exclude this from our list of predictors. 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.

Process

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.

```
# 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)

# dim(adultpayclean_train)

# dim(adultpayclean_validation)

# > 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.

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.

```
# > 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

# build a confusion matrix for this simple model

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

## 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


# 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
```

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 in 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.

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, eduyears, sex, race, maritalstatus, relationship and hoursperweek.

```
# lm_fit <- adultpayclean_train %>%

# mutate(y = as.numeric(income == 'Above50K')) %>%

# lm(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
# relationship, 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] 0.8167

## Confusion Matrix and Statistics Reference Prediction Above50K AtBelow50K
## Above50K 344 161 AtBelow50K 374 2039 Accuracy : 0.8167 95% CI : (0.8021,
## 0.8305) No Information Rate : 0.7539 P-Value [Acc > NIR] : 2.788e-16 Kappa :
## 0.451 Mcnemar's Test P-Value : < 2.2e-16 Sensitivity : 0.4791 Specificity :
## 0.9268 Pos Pred Value : 0.6812 Neg Pred Value : 0.8450 Prevalence : 0.2461
## Detection Rate : 0.1179 Detection Prevalence : 0.1731 Balanced Accuracy :
## 0.7030 'Positive' Class : Above50K
```

The accuracy for this model is 0.8167 however, sensitivity is 0.4791 and specificity is 0.9268. This indicates this model has higher ratio of negative outcomes than positive outcomes. Prevalence remains the same. Lets see if we can do better

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

```
# glm_fit <- adultpayclean_train %>%

# mutate(y = as.numeric(income == 'Above50K')) %>%

# glm(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
# relationship, 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] 0.8098

## Confusion Matrix and Statistics Reference Prediction Above50K AtBelow50K
## Above50K 279 116 AtBelow50K 439 2084 Accuracy : 0.8098 95% CI : (0.7951,
## 0.8239) No Information Rate : 0.7539 P-Value [Acc > NIR] : 3.442e-13 Kappa :
## 0.3958 Mcnemar's Test P-Value : < 2.2e-16 Sensitivity : 0.38858 Specificity
## : 0.94727 Pos Pred Value : 0.70633 Neg Pred Value : 0.82600 Prevalence :
## 0.24606 Detection Rate : 0.09561 Detection Prevalence : 0.13537 Balanced
## Accuracy : 0.66793 'Positive' Class : Above50K
```

glm produces accuracy of 0.8098, however, specificity is still higher than sensitivity and prevalence about the same. We can do better!

naive bayes

We will now experiment with generative models like naive bayes classification. It assumes the features that go into the model is 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.

```
## train_nb <- adultpayclean_train %>% mutate(y = as.factor(income ==
## 'Above50K')) %>% naiveBayes(y ~ age + eduyears + sex + race + hoursperweek +
## maritalstatus + relationship,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 : 0.7718

## Confusion Matrix and Statistics y_hat_nb FALSE TRUE FALSE 1698 502 TRUE 164
## 554 Accuracy : 0.7718 95% CI : (0.7561, 0.7869) No Information Rate : 0.6381
## P-Value [Acc > NIR] : < 2.2e-16 Kappa : 0.469 Mcnemar's Test P-Value : <
## 2.2e-16 Sensitivity : 0.9119 Specificity : 0.5246 Pos Pred Value : 0.7718
## Neg Pred Value : 0.7716 Prevalence : 0.6381 Detection Rate : 0.5819
## Detection Prevalence : 0.7539 Balanced Accuracy : 0.7183 'Positive' Class :
## FALSE
```

Naive bayes gives us an accuracy of 0.7718 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. We can try other advanced algorithms and compare this ahead.

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

```
# temp <- adultpayclean_train %>%

# mutate(y = as.factor(income == 'Above50K'))

# set.seed(2008)

# control <- trainControl(method = 'cv', number = 10, p = .9)

# train_knn <- train(y ~ age + eduyears + sex + race + hoursperweek +
# maritalstatus + relationship, 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] 0.8081
```

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

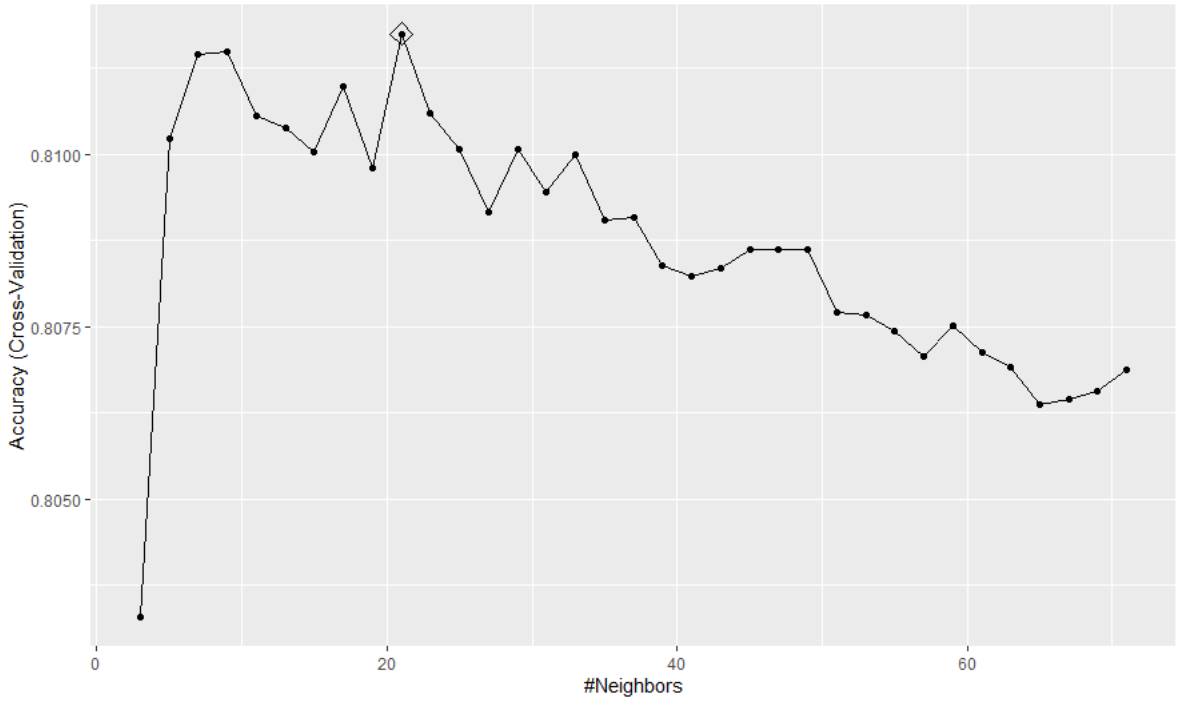


Figure 15: KNN tuning

Here is the confusion matrix for the knn tuned raw model

```
## cm_knn

## Confusion Matrix and Statistics Reference Prediction FALSE TRUE FALSE 1988
## 348 TRUE 212 370 Accuracy : 0.8081 95% CI : (0.7933, 0.8222) No Information
## Rate : 0.7539 P-Value [Acc > NIR] : 1.777e-12 Kappa : 0.4475 Mcnemar's Test
## P-Value : 1.165e-08 Sensitivity : 0.9036 Specificity : 0.5153 Pos Pred Value
## : 0.8510 Neg Pred Value : 0.6357 Prevalence : 0.7539 Detection Rate : 0.6813
## Detection Prevalence : 0.8005 Balanced Accuracy : 0.7095 'Positive' Class :
## FALSE
```

knn raw model produces same accuracy of 0.8081, however, specificity is lower than sensitivity and prevalence is now 0.753. We now see the positive prediction is better. Balanced accuracy is better. Can we do better with accuracy?

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.

```
# 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, 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'))

## cm_knntune

## Confusion Matrix and Statistics Reference Prediction FALSE TRUE FALSE 1987
## 342 TRUE 213 376 Accuracy : 0.8098 95% CI : (0.7951, 0.8239) No Information
## Rate : 0.7539 P-Value [Acc > NIR] : 3.442e-13 Kappa : 0.4544 Mcnemar's Test
## P-Value : 5.532e-08 Sensitivity : 0.9032 Specificity : 0.5237 Pos Pred Value
## : 0.8532 Neg Pred Value : 0.6384 Prevalence : 0.7539 Detection Rate : 0.6809
## Detection Prevalence : 0.7981 Balanced Accuracy : 0.7134 'Positive' Class :
## FALSE
```

The accuracy of the knn tuned classification is 0.8098 with a sensitivity of 0.9032 and specificity of 0.5237. This is better than the previous models. Prevalence is just about the same as raw knn. Balanced accuracy has improved.

Recursive partitioning with rpart

Next we use classification trees, or decision trees. We use the recursive partitioning library rpart for this.

```
# train_rpart <- train(y ~ age + eduyears + sex + race + hoursperweek +
# maritalstatus + relationship, method = 'rpart', tuneGrid = data.frame(cp =
# seq(0.0, 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

## 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 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.

We will next look at random forest algorithm.

Random forests

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.

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.

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

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


# > accuracy_rf Accuracy 0.8235

## Confusion Matrix and Statistics Reference Prediction FALSE TRUE FALSE 2016
## 331 TRUE 184 387 Accuracy : 0.8235 95% CI : (0.8092, 0.8372) No Information
## Rate : 0.7539 P-Value [Acc > NIR] : < 2.2e-16 Kappa : 0.4891 Mcnemar's Test
## P-Value : 1.247e-10 Sensitivity : 0.9164 Specificity : 0.5390 Pos Pred Value
## : 0.8590 Neg Pred Value : 0.6778 Prevalence : 0.7539 Detection Rate : 0.6909
## Detection Prevalence : 0.8043 Balanced Accuracy : 0.7277 'Positive' Class :
## FALSE
```

The accuracy of random forest is at 0.8235, sensitivity is at 0.9164 and specificity is at 0.5390. Prevalence

is about the same when compared to knn and rpart classification models. The confidence intervals is about the same as rpart.

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

```
# 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 })

# 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 0.8241947

## Confusion Matrix and Statistics Reference Prediction FALSE TRUE FALSE 2031
## 348 TRUE 169 370 Accuracy : 0.8228 95% CI : (0.8085, 0.8365) No Information
## Rate : 0.7539 P-Value [Acc > NIR] : < 2.2e-16 Kappa : 0.4787 Mcnemar's Test
## P-Value : 4.94e-15 Sensitivity : 0.9232 Specificity : 0.5153 Pos Pred Value
## : 0.8537 Neg Pred Value : 0.6865 Prevalence : 0.7539 Detection Rate : 0.6960
## Detection Prevalence : 0.8153 Balanced Accuracy : 0.7193 'Positive' Class :
## FALSE
```

The accuracy of random forest with tuning is at 0.8228, sensitivity is at 0.9232 and specificity is at 0.5153. 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.

Results

This project created a machine learning model that predicts income level of adults based on 7 shortlisted attributes. This model was run on a test dataset that included an income level indicator, allowing us to compare the predicted and actual value.

The model returned an accuracy of 82% while predicting annual income more than $50,000 annually. A summary table of results is given below:

	Method	Accuracy	Sensitivity	Specificity	Prevalence
1.	Plain old guess	0.50034	0.48329	0.50591	0.24606
2.	linear model	0.81666	0.47911	0.92682	0.24606
3.	General linear model	0.8098	0.38858	0.94727	0.24606
4.	naive bayes	0.77176	0.91192	0.52462	0.63811
5.	knn	0.80809	0.90364	0.51532	0.75394
6.	knn3	0.80843	0.90136	0.52368	0.75394
7.	knn tune	0.81151	0.90318	0.52368	0.75394
8.	rpart	0.82111	0.91	0.54875	0.75394
9.	rf	0.82351	0.91636	0.539	0.75394
10.	rf tune	0.82248	0.92318	0.51532	0.75394

Figure 16: Final Results

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
- sex
- race
- hoursperweek
- maritalstatus
- relationship

Conclusion

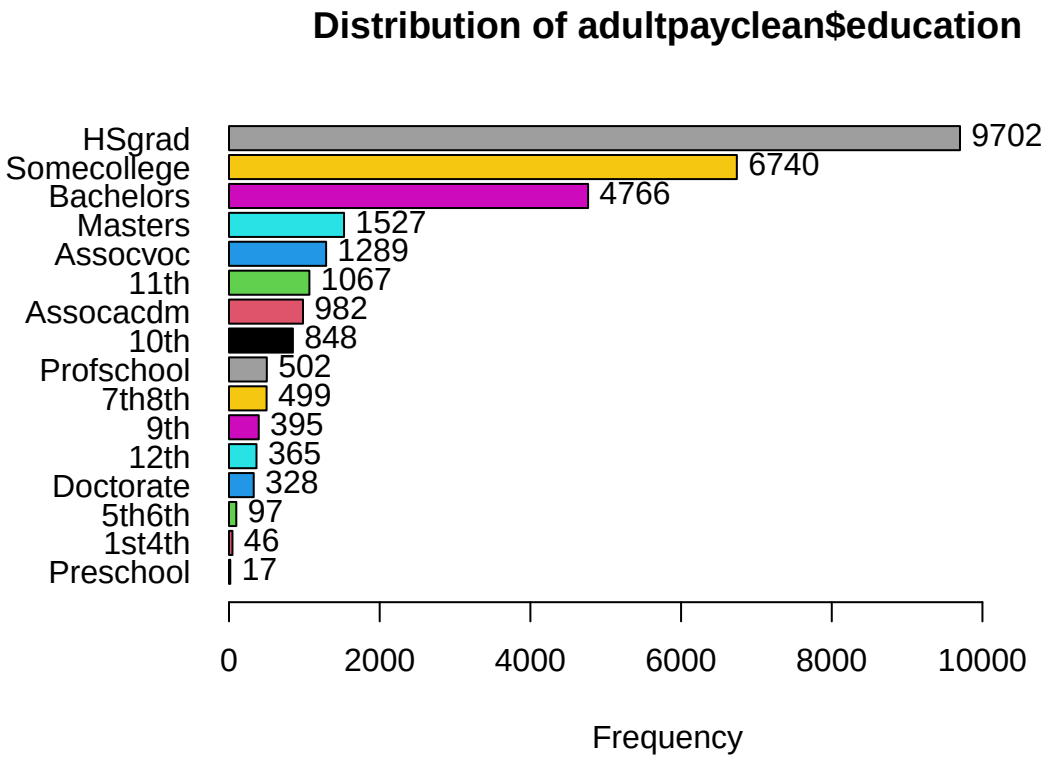
This machine learning model was able to predict annual incomes of persons in US based (1994) on 7 parameters with an accuracy of 82%. This model can be applied to data from other census years as well. The model will perform better if the training set is updated with new data that is confirmed for correctness, that is, the label value is the real life value, and not the predicted value. The model can continuously learn from changing data in the training set to adapt to new parameters, thus improving its accuracy and other metrics

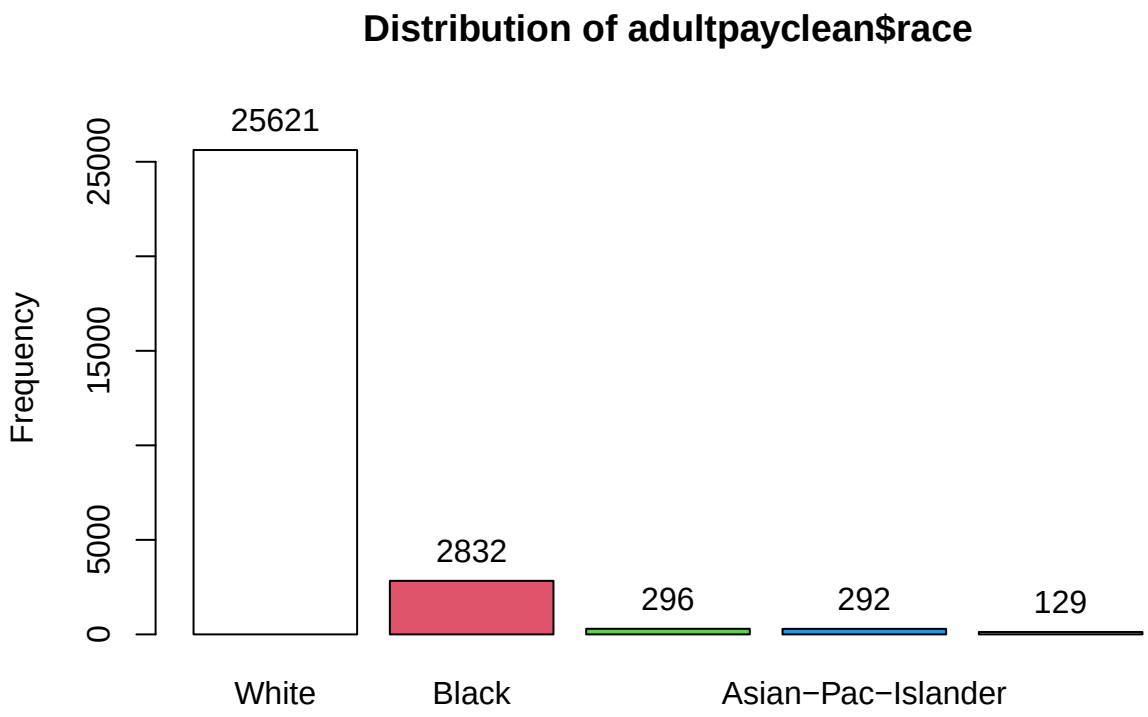
Appendix A - Complete code

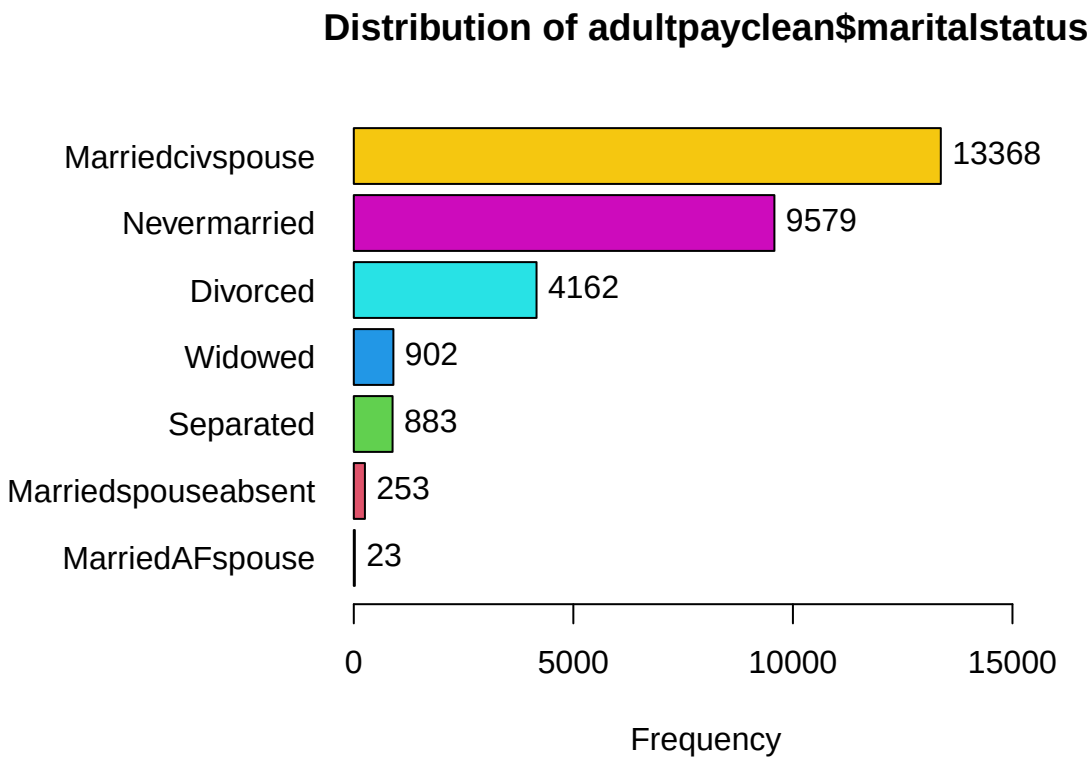
```
## @knitr CensusPayR

# 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
```

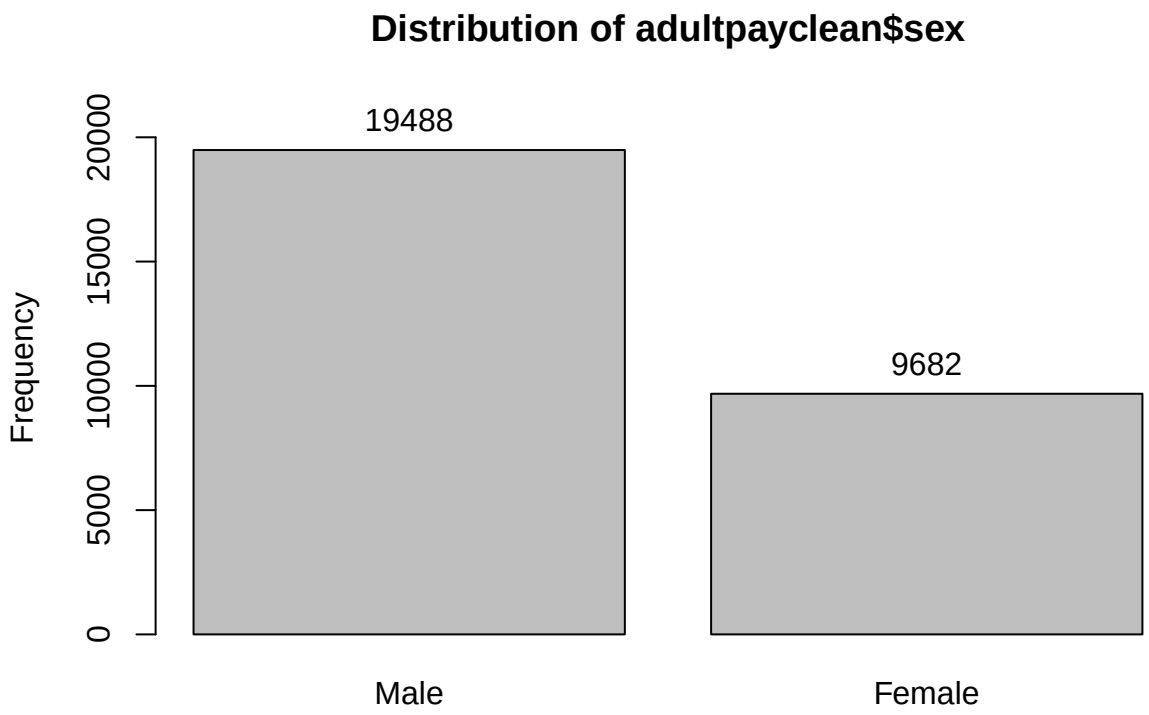
```r
# Execute the given source code for the project
source("DatasetProcessingCode.R")
```

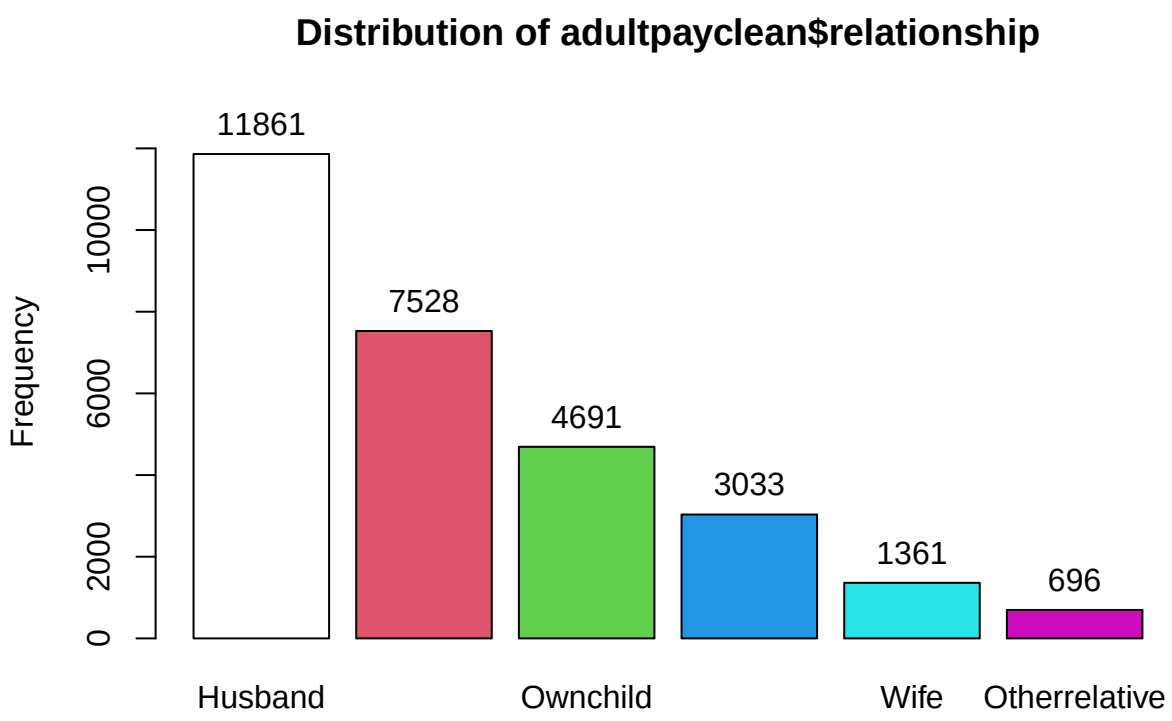
```
## Rows: 32,561
## Columns: 15
## $ age            <int> 90, 82, 66, 54, 41, 34, 38, 74, 68, 41, 45, 38, 52, 32,~
## $ workclass      <chr> "?", "Private", "?", "Private", "Private", "Private", "~
## $ fnlwgt         <int> 77053, 132870, 186061, 140359, 264663, 216864, 150601, ~
## $ education      <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", "~
## Rows: 29,170
## Columns: 13
## $ age           <int> 90, 82, 66, 54, 41, 34, 38, 74, 68, 45, 38, 52, 32, 51, ~
## $ fnlwgt        <int> 77053, 132870, 186061, 140359, 264663, 216864, 150601, 8~
## $ education     <fct> HSgrad, HSgrad, Somecollege, 7th8th, Somecollege, HSgrad~
## $ eduyears      <int> 9, 9, 10, 4, 10, 9, 6, 16, 9, 16, 15, 13, 14, 16, 15, 7,~
## $ maritalstatus <fct> Widowed, Widowed, Widowed, Divorced, Separated, Divorced~
## $ occupation    <fct> Unknown, Execmanagerial, Unknown, Machineopinspct, Profs~
## $ relationship  <fct> Notinfamily, Notinfamily, Unmarried, Unmarried, Ownchild~
## $ race          <fct> White, White, Black, White, White, White, White, White, ~
## $ sex           <fct> Female, Female, Female, Female, Female, Female, Male, Fe~
## $ hoursperweek  <int> 40, 18, 40, 40, 40, 45, 40, 20, 40, 35, 45, 20, 55, 40, ~
## $ native        <chr> "UnitedStates", "UnitedStates", "UnitedStates", "UnitedS~
## $ income        <fct> AtBelow50K, AtBelow50K, AtBelow50K, AtBelow50K, AtBelow5~
## $ class         <fct> Unknown, Private, Unknown, Private, Private, Private, Pr~
```

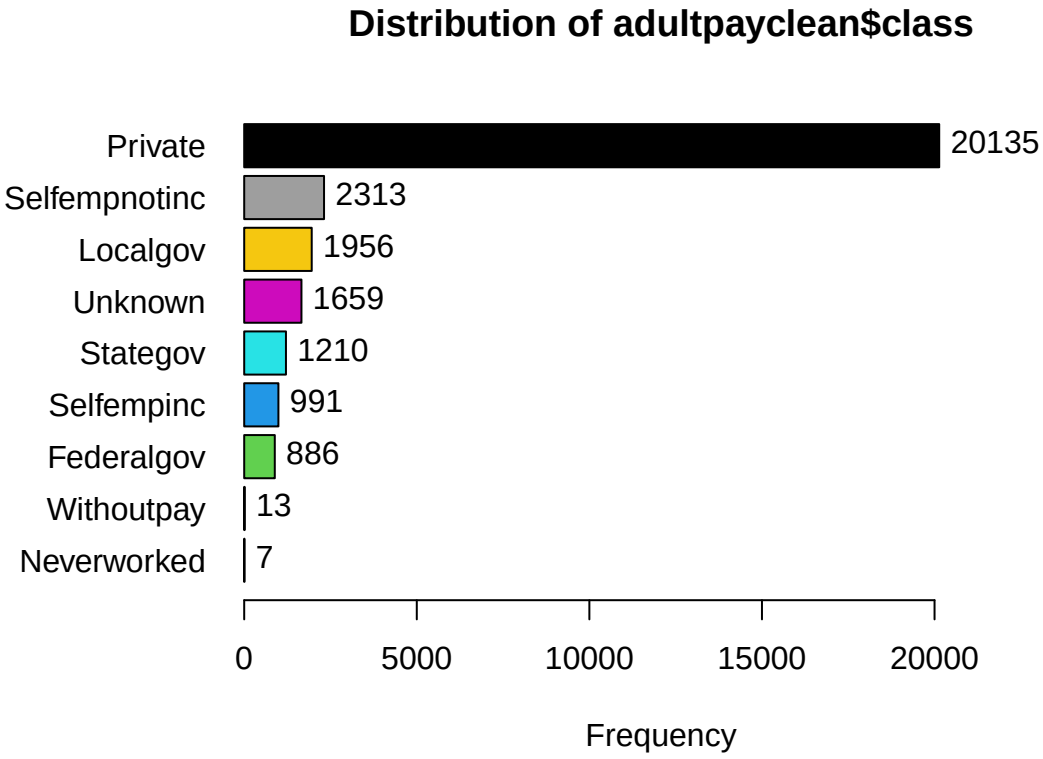

Distribution of adultpayclean$education
HSgrad
Somecollege
Bachelors
Masters
Assocvoc
11th
Assocacdm
10th
Profschool
7th8th
9th
12th
Doctorate
5th6th
1st4th
Preschool
9702
6740
4766
1527
1289
1067
982
848
502
499
395
365
328
97
46
17
0
2000
4000
6000
8000
10000
Frequency

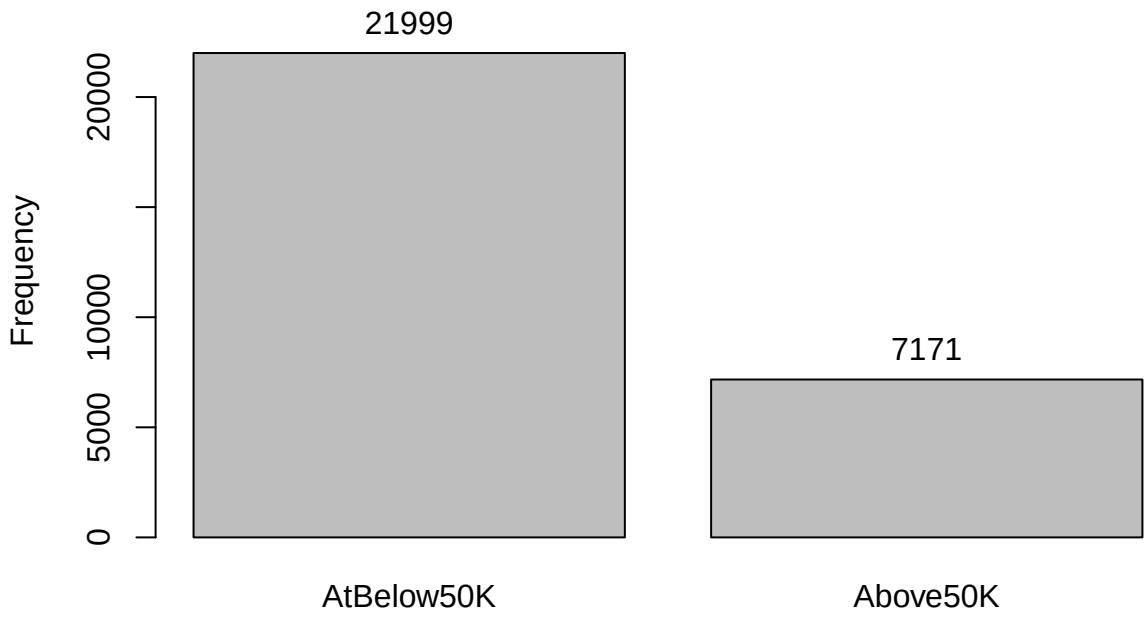

Distribution of adultpayclean$race
Frequency
0
5000
15000
25000
25621
2832
296
292
129
White
Black
Asian−Pac−Islander


Distribution of adultpayclean$maritalstatus
Marriedcivspouse
Nevermarried
Divorced
Widowed
Separated
Marriedspouseabsent
MarriedAFspouse
13368
9579
4162
902
883
253
23
0
5000
10000
15000
Frequency


Distribution of adultpayclean$sex
Frequency
0
5000
10000
15000
20000
19488
9682
Male
Female


Distribution of adultpayclean$relationship
Frequency
0
2000
6000
10000
11861
7528
4691
3033
1361
696
Husband
Ownchild
Wife
Otherrelative


Distribution of adultpayclean$class
Private
Selfempnotinc
Localgov
Unknown
Stategov
Selfempinc
Federalgov
Withoutpay
Neverworked
20135
2313
1956
1659
1210
991
886
13
7
0
5000
10000
15000
20000
Frequency




```
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")

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


set.seed(1996, 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$income))
accuracy_guess <- mean(seat_of_the_pants == adultpayclean_validation$income)

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

```
##             actual
```

```
## predicted    Above50K AtBelow50K
##   Above50K        347       1087
##   AtBelow50K      371       1113
```

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

```
## # A tibble: 2 x 2
##   income     accuracy
##   <fct>         <dbl>
## 1 Above50K      0.483
## 2 AtBelow50K    0.506
```

```
# confusion matrix using R function
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
##
```

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

```r
# 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,
        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
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Above50K AtBelow50K
##   Above50K        344        161
##   AtBelow50K      374       2039
##
##                Accuracy : 0.8167
##                  95% CI : (0.8021, 0.8305)
##     No Information Rate : 0.7539
##     P-Value [Acc > NIR] : 2.788e-16
##
##                   Kappa : 0.451
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4791
##             Specificity : 0.9268
##          Pos Pred Value : 0.6812
##          Neg Pred Value : 0.8450
##              Prevalence : 0.2461
##          Detection Rate : 0.1179
##    Detection Prevalence : 0.1731
##       Balanced Accuracy : 0.7030
##
##        'Positive' Class : Above50K
##
```

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

```r
# 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,
        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
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Above50K AtBelow50K
##   Above50K        279        116
##   AtBelow50K      439       2084
##
##                Accuracy : 0.8098
##                  95% CI : (0.7951, 0.8239)
##     No Information Rate : 0.7539
##     P-Value [Acc > NIR] : 3.442e-13
##
##                   Kappa : 0.3958
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.38858
##             Specificity : 0.94727
##          Pos Pred Value : 0.70633
##          Neg Pred Value : 0.82600
##              Prevalence : 0.24606
##          Detection Rate : 0.09561
##    Detection Prevalence : 0.13537
##       Balanced Accuracy : 0.66793
##
##        'Positive' Class : Above50K
##
```

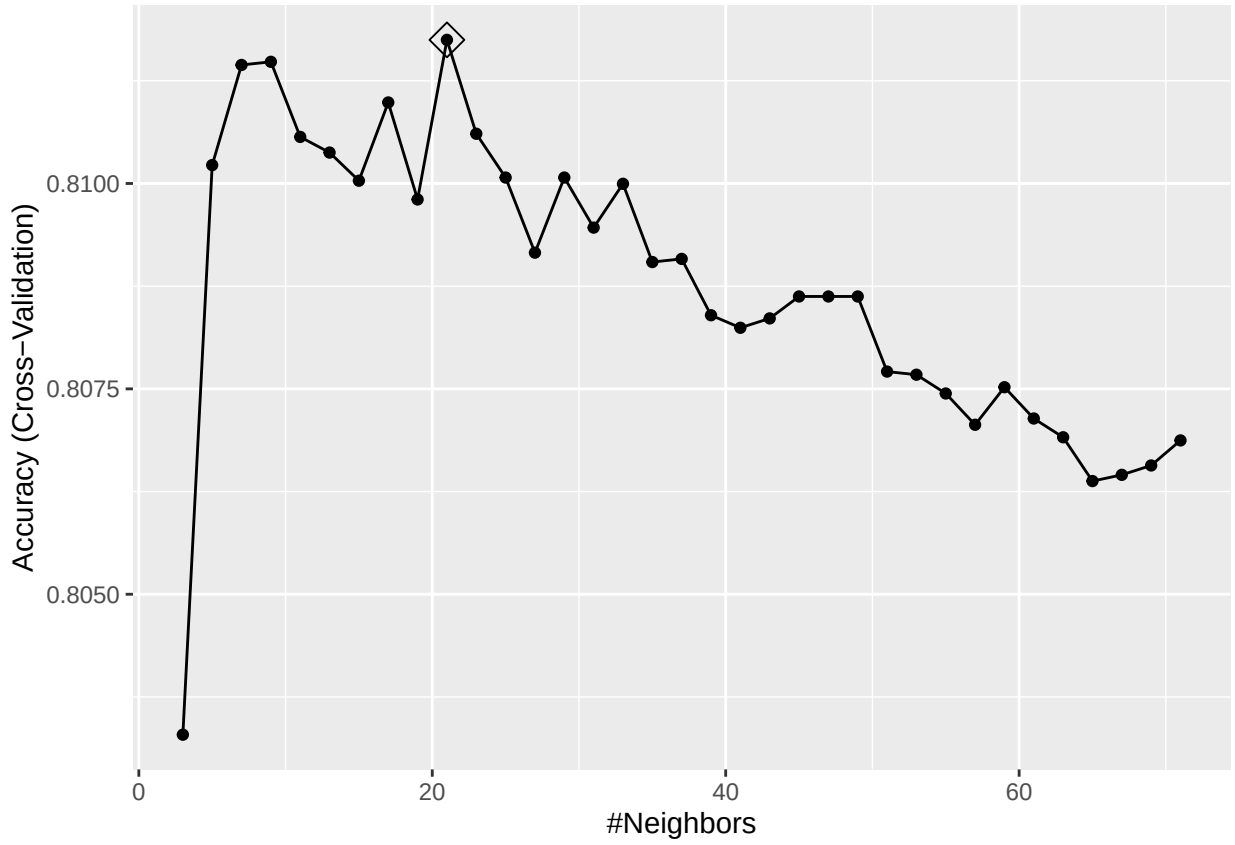
```r
# record the sensitivity, specificity, and prevalence
sensitivity_glm <- cm_glm$byClass[["Sensitivity"]]
specificity_glm <- cm_glm$byClass[["Specificity"]]
prevalence_glm <- cm_glm$byClass[["Prevalence"]]
```

```r
# Naive bayes

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

```
## Confusion Matrix and Statistics
##
##        y_hat_nb
##         FALSE TRUE
##   FALSE  1698  502
##   TRUE    164  554
##
##                Accuracy : 0.7718
##                  95% CI : (0.7561, 0.7869)
##     No Information Rate : 0.6381
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.469
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9119
##             Specificity : 0.5246
##          Pos Pred Value : 0.7718
##          Neg Pred Value : 0.7716
##              Prevalence : 0.6381
##          Detection Rate : 0.5819
##    Detection Prevalence : 0.7539
##       Balanced Accuracy : 0.7183
##
##        'Positive' Class : FALSE
##
```

```r
accuracy_nb <- cm_nb$overall[["Accuracy"]]
sensitivity_nb <- cm_nb$byClass[["Sensitivity"]]
specificity_nb <- cm_nb$byClass[["Specificity"]]
prevalence_nb <- cm_nb$byClass[["Prevalence"]]

# 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, 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
```

```
##     k
## 10 21
```

```
train_knn$finalModel
```

```
## 21-nearest neighbor model
## Training set outcome distribution:
##
## FALSE  TRUE
## 19799  6453
```

```r
# 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
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  1988  348
##      TRUE    212  370
##
##                Accuracy : 0.8081
##                  95% CI : (0.7933, 0.8222)
##     No Information Rate : 0.7539
##     P-Value [Acc > NIR] : 1.777e-12
##
##                   Kappa : 0.4475
##
##  Mcnemar's Test P-Value : 1.165e-08
##
##             Sensitivity : 0.9036
##             Specificity : 0.5153
##          Pos Pred Value : 0.8510
##          Neg Pred Value : 0.6357
##              Prevalence : 0.7539
##          Detection Rate : 0.6813
##    Detection Prevalence : 0.8005
##       Balanced Accuracy : 0.7095
##
##        'Positive' Class : FALSE
##
```

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

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

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

cm_knntune
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  1987  342
##      TRUE    213  376
##
##                Accuracy : 0.8098
##                  95% CI : (0.7951, 0.8239)
##     No Information Rate : 0.7539
##     P-Value [Acc > NIR] : 3.442e-13
##
##                   Kappa : 0.4544
##
##  Mcnemar's Test P-Value : 5.532e-08
##
##             Sensitivity : 0.9032
##             Specificity : 0.5237
##          Pos Pred Value : 0.8532
##          Neg Pred Value : 0.6384
##              Prevalence : 0.7539
##          Detection Rate : 0.6809
##    Detection Prevalence : 0.7981
##       Balanced Accuracy : 0.7134
```

```
##
##        'Positive' Class : FALSE
##
```

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


# 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, method = "rpart", tuneGrid = data.frame(cp = seq(0, 0.1, len = 25)),
    data = temp)
# predict the outcomes with this model
y_hat <- predict(train_rpart, adultpayclean_validation)
# confusion matrix for the rpart model
cm_rpart <- confusionMatrix(y_hat, as.factor(adultpayclean_validation$income == "Above50K"))
# get the accuracy
accuracy_rpart <- confusionMatrix(y_hat, as.factor(adultpayclean_validation$income ==
    "Above50K"))$overall["Accuracy"]

cm_rpart
```

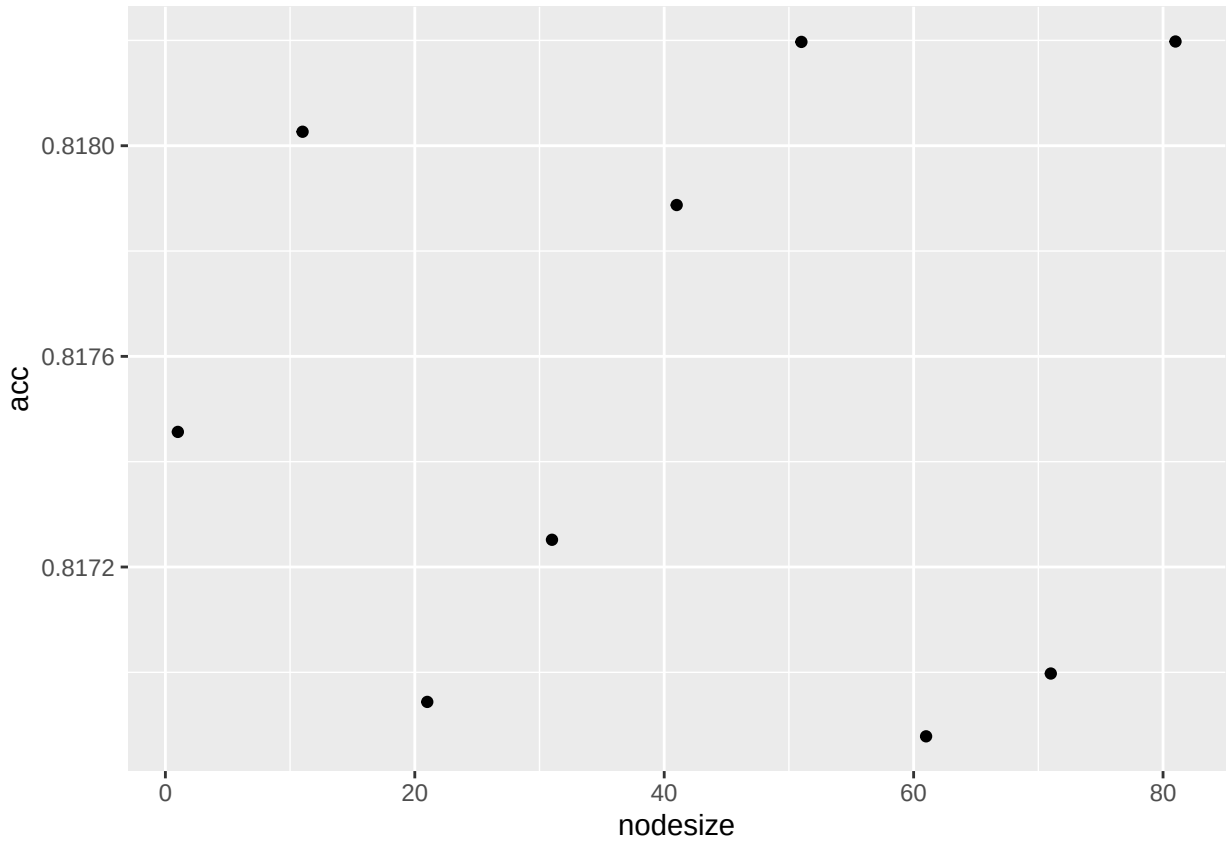
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  2002  324
##      TRUE    198  394
##
##                Accuracy : 0.8211
##                  95% CI : (0.8067, 0.8349)
##     No Information Rate : 0.7539
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   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
##
```

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

# random forest
set.seed(2008)
# train the vanilla random forest model
train_rf <- randomForest(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
    relationship, data = temp)
# create the confusionMatrix
cm_rf <- confusionMatrix(predict(train_rf, adultpayclean_validation), as.factor(adultpayclean_validatio
    "Above50K"))
# get the accuracy
accuracy_rf <- confusionMatrix(predict(train_rf, adultpayclean_validation), as.factor(adultpayclean_vali
    "Above50K"))$overall["Accuracy"]

cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  2016  331
##      TRUE    184  387
##
##                Accuracy : 0.8235
##                  95% CI : (0.8092, 0.8372)
##     No Information Rate : 0.7539
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4891
##
##  Mcnemar's Test P-Value : 1.247e-10
##
##             Sensitivity : 0.9164
##             Specificity : 0.5390
##          Pos Pred Value : 0.8590
##          Neg Pred Value : 0.6778
##              Prevalence : 0.7539
##          Detection Rate : 0.6909
##    Detection Prevalence : 0.8043
##       Balanced Accuracy : 0.7277
##
##        'Positive' Class : FALSE
##
```

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

```
# 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,
        method = "rf", data = temp, tuneGrid = data.frame(mtry = 2), nodesize = ns)$results$Accuracy
})
qplot(nodesize, acc)
```




```
# get the trained model for the max node size
train_rf_2 <- randomForest(y ~ age + eduyears + sex + race + hoursperweek + maritalstatus +
    relationship, 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(predict(train_rf_2, adultpayclean_validation), as.factor(adultpayclean_valida
    "Above50K"))
# get the accuracy
accuracy_rftune <- confusionMatrix(predict(train_rf_2, adultpayclean_validation),
    as.factor(adultpayclean_validation$income == "Above50K"))$overall["Accuracy"]

cm_rf2
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction FALSE TRUE
##      FALSE  2031  348
##      TRUE    169  370
##
##                Accuracy : 0.8228
##                  95% CI : (0.8085, 0.8365)
##     No Information Rate : 0.7539
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4787
##
##  Mcnemar's Test P-Value : 4.94e-15
##
##             Sensitivity : 0.9232
##             Specificity : 0.5153
##          Pos Pred Value : 0.8537
##          Neg Pred Value : 0.6865
##              Prevalence : 0.7539
##          Detection Rate : 0.6960
##    Detection Prevalence : 0.8153
##       Balanced Accuracy : 0.7193
##
##        'Positive' Class : FALSE
##
```

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

# 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), "linear model",
    round(accuracy_lm, 5), round(sensitivity_lm, 5), round(specificity_lm, 5), round(prevalence_lm,
        5), "General linear model", round(accuracy_glm, 5), round(sensitivity_glm,
        5), round(specificity_glm, 5), round(prevalence_glm, 5), "naive bayes", round(accuracy_nb,
        5), round(sensitivity_nb, 5), round(specificity_nb, 5), round(prevalence_nb,
        5), "knn", round(accuracy_knn, 5), round(sensitivity_knn, 5), round(specificity_knn,
        5), round(prevalence_knn, 5), "knn tune", round(accuracy_knntune, 5), round(sensitivity_knntune
        5), round(specificity_knntune, 5), round(prevalence_knntune, 5), "rpart",
    round(accuracy_rpart, 5), round(sensitivity_rpart, 5), round(specificity_rpart,
        5), round(prevalence_rpart, 5), "rf", round(accuracy_rf, 5), round(sensitivity_rf,
        5), round(specificity_rf, 5), round(prevalence_rf, 5), "rf tune", round(accuracy_rftune,
        5), round(sensitivity_rf2, 5), round(specificity_rf2, 5), round(prevalence_rf2,
        5)), nrow = 9, ncol = 5, byrow = TRUE, dimnames = list(c("1.", "2.", "3.",
    "4.", "5.", "6.", "7.", "8.", "9."), c("Method", "Accuracy", "Sensitivity", "Specificity",
    "Prevalence")))
# style the table with knitr
accuracy_results %>%
    knitr::kable() %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

	Method	Accuracy	Sensitivity	Specificity	Prevalence
1.	Plain old guess	0.50034	0.48329	0.50591	0.24606
2.	linear model	0.81666	0.47911	0.92682	0.24606
3.	General linear model	0.8098	0.38858	0.94727	0.24606
4.	naive bayes	0.77176	0.91192	0.52462	0.63811
5.	knn	0.80809	0.90364	0.51532	0.75394
6.	knn tune	0.81151	0.90318	0.52368	0.75394
7.	rpart	0.82111	0.91	0.54875	0.75394
8.	rf	0.82351	0.91636	0.539	0.75394
9.	rf tune	0.82248	0.92318	0.51532	0.75394

Appendix B - 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