

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

Preface	1
Introduction to Census income level predictions	2
Analysis	4
Data Exploration, cleaning, processing and, feature engineering	4
Process	6
Model Creation	6
Results	14
Conclusion	14
Appendix A - Complete code	14
Appendix B - Links	23
Citations	23

List of Figures

1 KNN tuning	10
---------------------------	----

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', 'Dw~

# $ 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 Marriedciuspouse :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 Dunchild : 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
```

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

```
# table(adultpayclean$education) 10th 11th 12th 1st4th 5th6th 7th8th 9th
# Assocacdm

# 848 1067 365 46 97 499 395 982 Assocvoc Bachelors Doctorate HSgrad Masters
# Preschool Profschool Somecollege

# 1289 4766 328 9702 1527 17 502 6740

# table(adultpayclean$maritalstatus) Divorced MarriedAFspouse Marriedciuspouse
# Marriedspouseabsent Nevermarried Separated

# 4162 23 13368 253 9579 883 Widowed

# 902
```

```

# table (adultpayclean$occupation) Admclerical ArmedForces Craftrepair
# Execmanagerial Farmingfishing Handlerscleaners Machineopinspt

# 3449 9 3685 3735 879 1189 1687

# Otherservice Privhouseserv Profspecialty Protectiveserv Sales Techsupport
# Transportmoving

# 2777 90 3693 606 3364 850 1491

# Unknown

# 1666

# table(adultpayclean$relationship) Husband Notinfamily Otherrelative Ownchild
# Unmarried Wife

# 11861 7528 696 4691 3033 1361

# table(adultpayclean$race) Amer-Indian-Eskimo Asian-Pac-Islander Black Other
# White

# 296 292 2832 129 25621 table(adultpayclean$sex) Female Male

# 9682 19488 table(adultpayclean$class) Federalgov Localgov Neverworked Private
# Selfempinc Selfempnotinc Stategov Unknown Withoutpay

# 886 1956 7 20135 991 2313 1210 1659 13

```

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.k.a. 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 generate 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') # if using R 3.5 or earlier, use
# `set.seed(1)` test_index <- createDataPartition( y = adu1tpayclean$income,
# times = 1, p = 0.1, list = FALSE ) adu1tpayclean_train <-
# adu1tpayclean[-test_index, ] adu1tpayclean_validation <-
# adu1tpayclean[test_index, ]

# dim(adu1tpayclean) dim(adu1tpayclean_train) dim(adu1tpayclean_validation)

# > dim(adu1tpayclean) [1] 29170 13 > dim(adu1tpayclean_train) [1] 26252 13 >
# dim(adu1tpayclean_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.

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. It's 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(adu1tpayclean_validation$income))

# > mean(seat_of_the_pants == adu1tpayclean_validation$income)

# [1] 0.5010281

# build a confusion matrix for this simple model

# cm <- confusionMatrix(data = seat_of_the_pants , reference =
# adu1tpayclean_validation$income)
```

```

# Reference Prediction Above50K AtBelow50K Above50K 332 1102 AtBelow50K 386
# 1098

# Accuracy : 0.4901 95% CI : (0.4718, 0.5084) No Information Rate : 0.7539
# P-Value [Acc > NIR] : 1

# Kappa : -0.0288

# McNemar's Test P-Value : <2e-16

# Sensitivity : 0.4624 Specificity : 0.4991 Pos Pred Value : 0.2315 Neg Pred
# Value : 0.7399 Prevalence : 0.2461 Detection Rate : 0.1138 Detection
# Prevalence : 0.4914 Balanced Accuracy : 0.4807

# '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, edueyears, sex, race, maritalstatus, relationship and hoursperweek.

```

# lm_fit <- adultpayclean_train %>%

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

# lm(y ~ age + edueyears + 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] 0.8166552

# 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] 0.8098012

# 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!

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 + eduyyears + 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] 0.8098012

```

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

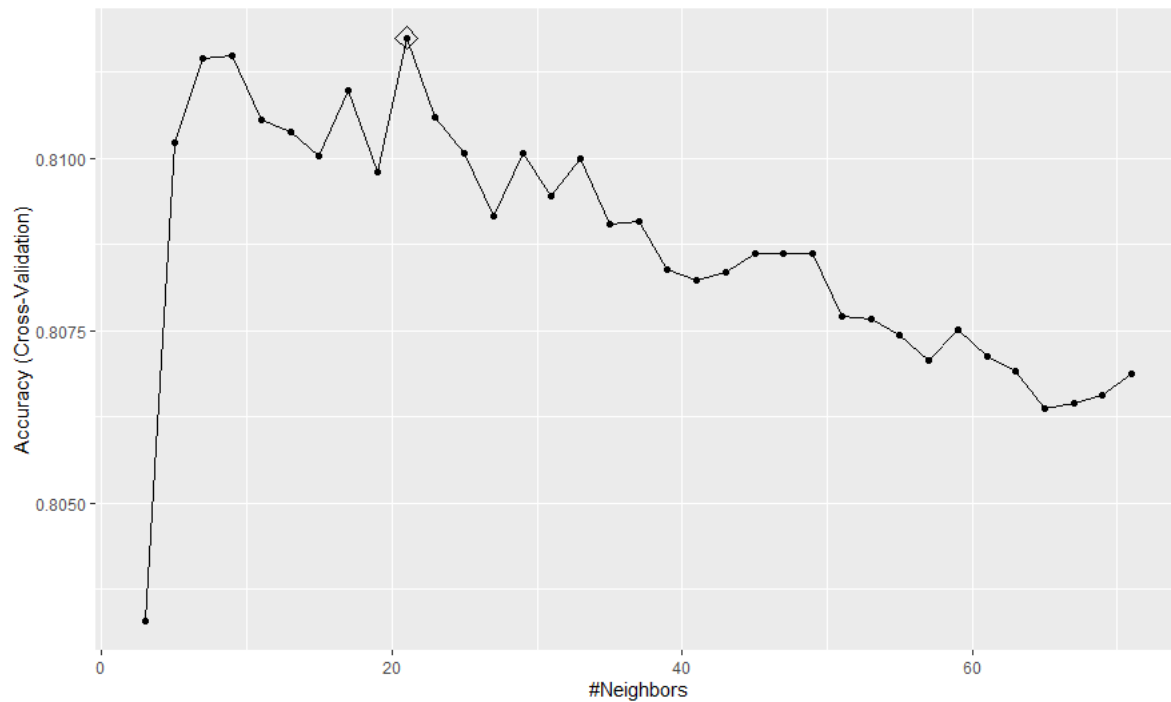


Figure 1: KNN tuning

Here is the confusion matrix for the knn tuned raw model

```

# 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
```

knn raw model produces same accuracy of 0.8098, 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)

# > ks[which.max(knntune$test)] [1] 17 > knn_fit <- knn3(y ~ age + eduyears +
# sex + race + hoursperweek + maritalstatus + relationship, data = temp, k =
# 17)

# > y_hat <- predict(knn_fit, temp, type = 'class') > cm_train <-
# confusionMatrix(y_hat, temp$y) > cm_train

# Confusion Matrix and Statistics

# Reference Prediction FALSE TRUE FALSE 18001 2772 TRUE 1798 3681

# Accuracy : 0.8259
```

```
# 95% CI : (0.8213, 0.8305) No Information Rate : 0.7542 P-Value [Acc > NIR] :
# < 2.2e-16

# Kappa : 0.5053

# McNemar's Test P-Value : < 2.2e-16

# Sensitivity : 0.9092 Specificity : 0.5704 Pos Pred Value : 0.8666 Neg Pred
# Value : 0.6718 Prevalence : 0.7542 Detection Rate : 0.6857 Detection
# Prevalence : 0.7913 Balanced Accuracy : 0.7398

# 'Positive' Class : FALSE
```

The accuracy of the knn3 classification is 0.8259 with a sensitivity of 0.9092 and specificity of 0.5704. 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 0.8211103

# 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 slightly lower prevalence. Balanced accuracy is slightly lower. Between rpart and knn3 classification it appears knn3 looks better as the confidence intervals for above 50k is higher when compared to rpart.

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, adu1tpayclean_validation),  
# as.factor(adu1tpayclean_validation$income == 'Above50K'))$overall['Accuracy']  
  
# > accuracy_rf Accuracy 0.8235093  
  
# 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 of KNN classification is still better than vanilla random forest model.

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, adu1tpayclean_validation)

# accuracy_rftune <- confusionMatrix(predict(train_rf_2,
# adu1tpayclean_validation), as.factor(adu1tpayclean_validation$income ==
# 'Above50K'))$overall['Accuracy']

# > accuracy_rftune Accuracy 0.8241947

# Confusion Matrix and Statistics

# Reference Prediction FALSE TRUE FALSE 2025 338 TRUE 175 380

# Accuracy : 0.8242 95% CI : (0.8099, 0.8378) No Information Rate : 0.7539
# P-Value [Acc > NIR] : < 2.2e-16

# Kappa : 0.4869

# Mcnemar's Test P-Value : 8.523e-13

# Sensitivity : 0.9205 Specificity : 0.5292 Pos Pred Value : 0.8570 Neg Pred
# Value : 0.6847 Prevalence : 0.7539 Detection Rate : 0.6940 Detection
# Prevalence : 0.8098 Balanced Accuracy : 0.7249

# 'Positive' Class : FALSE

```

The accuracy of random forest with tuning is at 0.8242, sensitivity is at 0.9205 and specificity is at 0.5292. Prevalence is about the same when compared to knn and rpart classification models. The confidence intervals of KNN classification is still better than tuned random forest model.

Results

Conclusion

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

# 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~
## $ 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~

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

library(caret)
library(gridExtra)
library(kableExtra)
library(randomForest)

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"]]

# 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 = adu1tpayclean_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, adu1tpayclean_validation$income)
accuracy_lm <- confusionMatrix(y_hat_logit, adu1tpayclean_validation$income)$overall[["Accuracy"]]

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

# general linear model create the glm model
glm_fit <- adu1tpayclean_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 = adu1tpayclean_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, adu1tpayclean_validation$income)
accuracy_glm <- confusionMatrix(y_hat_logit, adu1tpayclean_validation$income)$overall[["Accuracy"]]

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

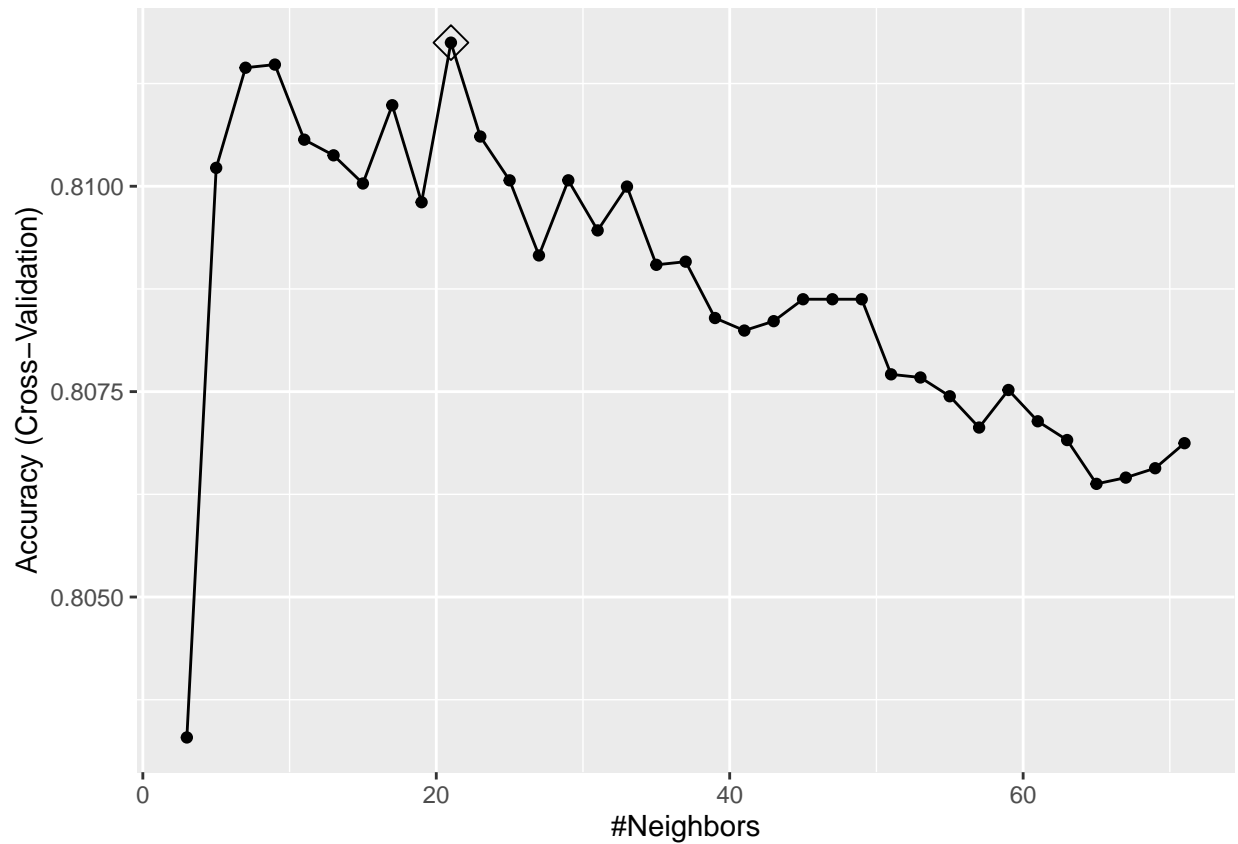
# translate income factor into binary outcome
temp <- adu1tpayclean_train %>%
  mutate(y = as.factor(income == "Above50K"))

# 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
```

```
# use this trained model to predict raw knn predictions
y_hat_knn <- predict(train_knn, adu1tpayclean_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"]]

# 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 knn3 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 = 17)

y_hat <- predict(knn_fit, temp, type = "class")
cm_knntune <- confusionMatrix(y_hat, temp$y)

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

# k-nearest using knn3
set.seed(2008)

```

```

knn3_fit <- knn3(y ~ age + edueyears + sex + race + hoursperweek + maritalstatus +
  relationship, data = temp, k = 17)
y_hat_knn3 <- predict(knn3_fit, adu1tpayclean_validation, type = "class")

cm_knn3 <- confusionMatrix(y_hat_knn3, as.factor(adu1tpayclean_validation$income ==
  "Above50K"))
accuracy_knn3 <- confusionMatrix(y_hat_knn3, as.factor(adu1tpayclean_validation$income ==
  "Above50K"))$overall["Accuracy"]

# record the sensitivity, specificity, and prevalence
sensitivity_knn3 <- cm_knn3$byClass[["Sensitivity"]]
specificity_knn3 <- cm_knn3$byClass[["Specificity"]]
prevalence_knn3 <- cm_knn3$byClass[["Prevalence"]]

# recursive partitioning using rpart
set.seed(2008)
# train the model with the recursive partitioning
train_rpart <- train(y ~ age + edueyears + 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, adu1tpayclean_validation)
# confusion matrix for the rpart model
cm_rpart <- confusionMatrix(y_hat, as.factor(adu1tpayclean_validation$income == "Above50K"))
# get the accuracy
accuracy_rpart <- confusionMatrix(y_hat, as.factor(adu1tpayclean_validation$income ==
  "Above50K"))$overall["Accuracy"]

# 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 + edueyears + sex + race + hoursperweek + maritalstatus +
  relationship, data = temp)
# create the confusionMatrix
cm_rf <- confusionMatrix(predict(train_rf, adu1tpayclean_validation), as.factor(adu1tpayclean_validation$income ==
  "Above50K"))
# get the accuracy
accuracy_rf <- confusionMatrix(predict(train_rf, adu1tpayclean_validation), as.factor(adu1tpayclean_validation$income ==
  "Above50K"))$overall["Accuracy"]

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

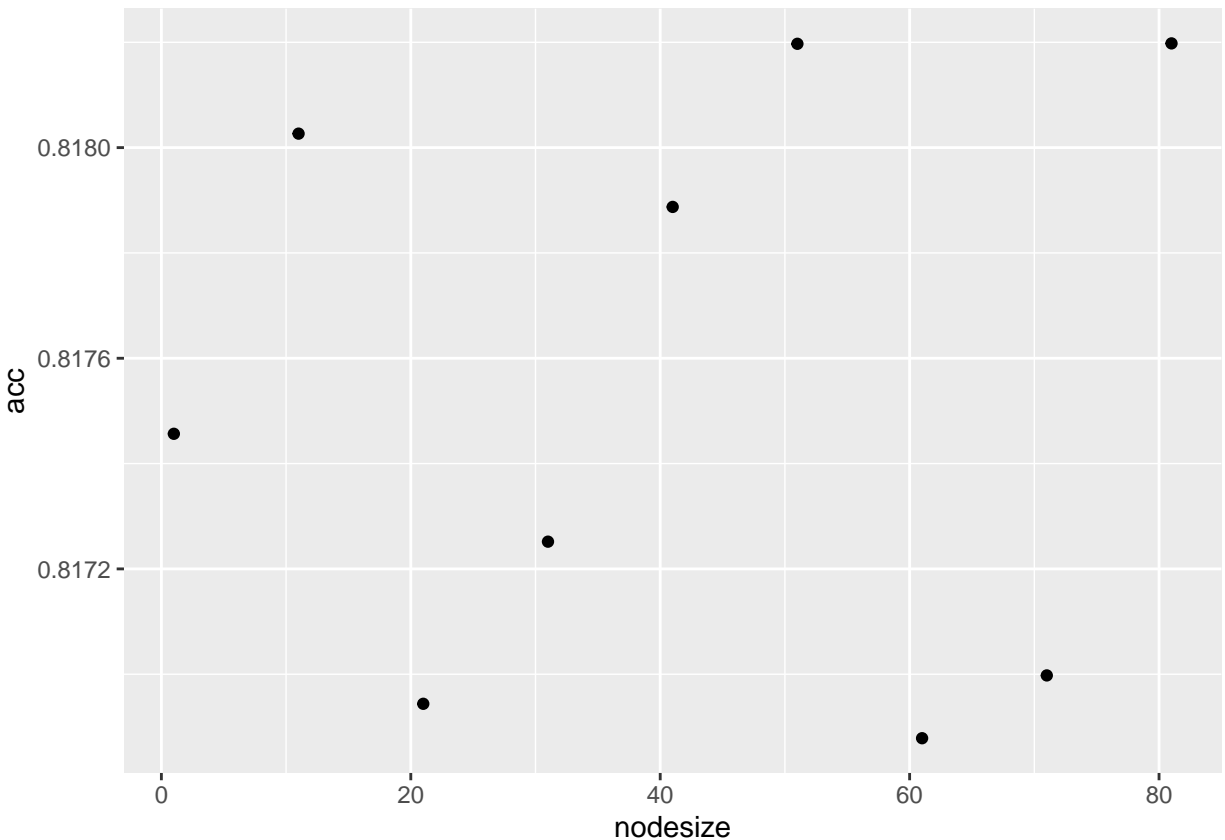
# random forest with tuning

```

```

nodesize <- seq(1, 90, 10)
acc <- sapply(nodesize, function(ns) {
  # train the model with tuning
  train(y ~ age + edueyears + 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 + edueyears + sex + race + hoursperweek + maritalstatus +
  relationship, data = temp, nodesize = nodesize[which.max(acc)])
# predict the outcomes
y_hat_rf2 <- predict(train_rf_2, adu1tpayclean_validation)
# get the confusion matrix for random forest model
cm_rf2 <- confusionMatrix(predict(train_rf_2, adu1tpayclean_validation), as.factor(adu1tpayclean_validation$income ==
  "Above50K"))
# get the accuracy
accuracy_rftune <- confusionMatrix(predict(train_rf_2, adu1tpayclean_validation),
  as.factor(adu1tpayclean_validation$income == "Above50K"))$overall["Accuracy"]

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

```

	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.	knn	0.80809	0.90364	0.51532	0.75394
5.	knn3	0.80843	0.90136	0.52368	0.75394
6.	knn tune	0.81151	0.90919	0.57043	0.75419
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

```

# 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), "knn", round(accuracy_knn,
5), round(sensitivity_knn, 5), round(specificity_knn, 5), round(prevalence_knn,
5), "knn3", round(accuracy_knn3, 5), round(sensitivity_knn3, 5), round(specificity_knn3,
5), round(prevalence_knn3, 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"))

```

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>