Customer Salary Prediction

Collin Guidry

11/3/2021

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(moments)
library(ggplot2)
library(lattice)
library(caret)
set.seed(1234)
library(naivebayes)
```

```
## naivebayes 0.9.7 loaded
```

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(caTools)
library(rpart.plot)
```

```
## Loading required package: rpart
```

```r
library(rpart)
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
#import the original adult csv from canvas here
df = read.csv("data/adult-salaries.csv")
```

```
#import the original adult csv from canvas here

names(df) <- c("age","workclass","fnlwgt","education", "education_num","marital_status","occupation","re
```

```
names(df)
```

```
##  [1] "age"            "workclass"      "fnlwgt"         "education"
##  [5] "education_num"  "marital_status" "occupation"     "relationship"
##  [9] "race"           "sex"            "capital_gain"   "capital_loss"
## [13] "hours_per_week" "native_country" "salary"
```

```
# create a data set with 500,000 by sampling the data we are given
# The probability of any row being generated is based on the "fnlwgt" column as a weight

df = sample_n( df, size = 200000, weight = fnlwgt, replace=TRUE)

#export this
write.csv(df,"data/adult_sampled.csv", row.names = FALSE)
```

```
#import the data that was previously sampled.
#We did the sampling once and stored it here, no need to do it again.

df = read.csv("data/adult_sampled.csv")
```

```
print( paste('There are',nrow(df),'rows and',ncol(df),'columns') )
```

```
## [1] "There are 200000 rows and 15 columns"
```

```
data.frame( data_type = sapply(df, class) )
```

```
##                data_type
## age              integer
## workclass      character
## fnlwgt           integer
## education      character
## education_num    integer
## marital_status character
## occupation     character
## relationship   character
## race           character
## sex            character
## capital_gain     integer
## capital_loss     integer
## hours_per_week   integer
## native_country character
## salary         character
```

"education_num" can be used to designate the levels when converting "education" to a factor

```r
data.frame( num_unique = sapply(df, n_distinct) ,
            data_type = sapply(df, class) ) %>%

    filter(data_type == 'integer') %>%
    arrange(num_unique)
```

```
##                num_unique data_type
## education_num          16   integer
## age                    73   integer
## capital_loss           92   integer
## hours_per_week         93   integer
## capital_gain          118   integer
## fnlwgt              21000   integer
```

For numeric variables, produce a table of statistics including missing values, min, max, median, mean, standard deviation, skewness and kurtosis.

"capital_gain" and "capital_loss" have the highest kurtosis, yet the most common value is zero.

If we remove their outliers, we are left with all zeroes.

We can replace the zeroes with NA, remove the outliers, then add the zeroes back

```r
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}

numeric_df =  select_if(df, is.numeric)

numeric_stats <- data.frame(
            unique = sapply(numeric_df, n_distinct),
            isNA = sapply(numeric_df, function(x) sum(is.na(x))),
            data_type = sapply(numeric_df, class),
            min = sapply(numeric_df, min),
            max = sapply(numeric_df, max),
            mean = round( sapply(numeric_df, mean) ,0),
            median = sapply(numeric_df, median),
            std = round( sapply(numeric_df, sd) ,0),
            skew = sapply(numeric_df, skewness),
            kurt = sapply(numeric_df, kurtosis),
            mode = sapply(numeric_df, getmode)

            ) %>%
    arrange(unique)

numeric_stats
```

```
##                unique isNA data_type   min     max   mean median    std
## education_num      16    0   integer     1      16     10     10      3
## age                73    0   integer    17      90     38     36     13
## capital_loss       92    0   integer     0    4356     86      0    399
## hours_per_week     93    0   integer     1      99     40     40     12
## capital_gain      118    0   integer     0   99999   1052      0   7245
## fnlwgt          21000    0   integer 14878 1484705 248501 217892 129179
##                      skew       kurt   mode
```

```
## education_num  -0.3589245   3.670237      9
## age             0.5971491   2.887081     23
## capital_loss    4.6610280  24.147707      0
## hours_per_week  0.2171465   6.038608     40
## capital_gain   12.1599732 163.620481      0
## fnlwgt          2.1340939  13.481092 241998
```

Outlier removal and imputation:

We do not want to remove outliers in cases where the max value is normal or where kurtosis is low.

```r
# Statistics generated on the outliers

data.frame(
  num_outliers = sapply(numeric_df, function(x){
        length(boxplot.stats(x)$out) }),

  outlier_mean = sapply(numeric_df, function(x){
        round(mean(boxplot.stats(x)$out),0) }),

  outlier_min = sapply(numeric_df, function(x){
        round(min(boxplot.stats(x)$out),0) }),

  outlier_max = round(sapply(numeric_df, max),0)
  ) %>%
  arrange(-num_outliers)
```

```
##                num_outliers outlier_mean outlier_min outlier_max
## hours_per_week        53840           37           1          99
## capital_gain          16489        12765         114       99999
## capital_loss           9145         1870         155        4356
## education_num          8218            3           1          16
## fnlwgt                 5969       690497      526528     1484705
## age                     870           83          78          90
```

Although it's abnormal, working 99 **hours__per__week** is possible.

- Don't remove

capital__gain and **capital__loss** need outliers removed, but 3 stds is not going to work. We will create **capital__net** that represents both the value gained or lost for the population.

- Remove

The highest **education__num** ber (doctorate) is normal and the lowest is also normal. They shouldn't skew out predictions.

- Don't remove

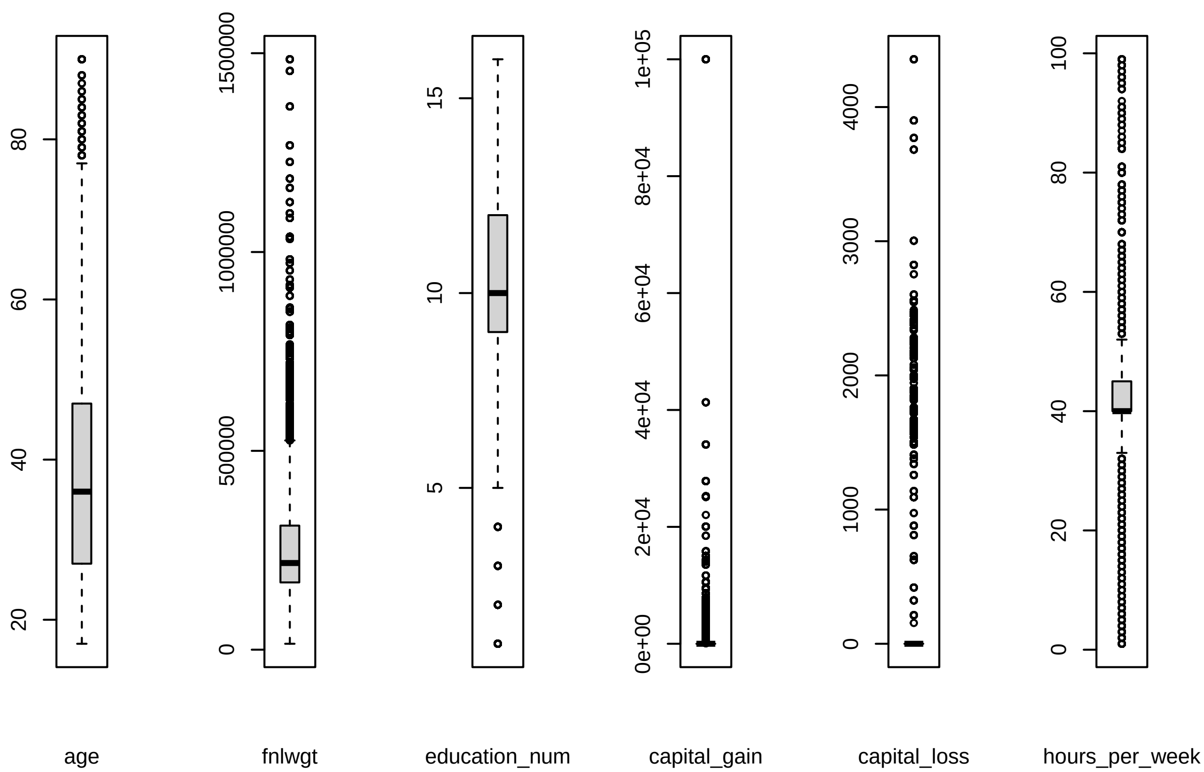
fnlwgt represents the size of the population with the row's criteria. Removing outliers is possible

A max **age** of 90 is normal.

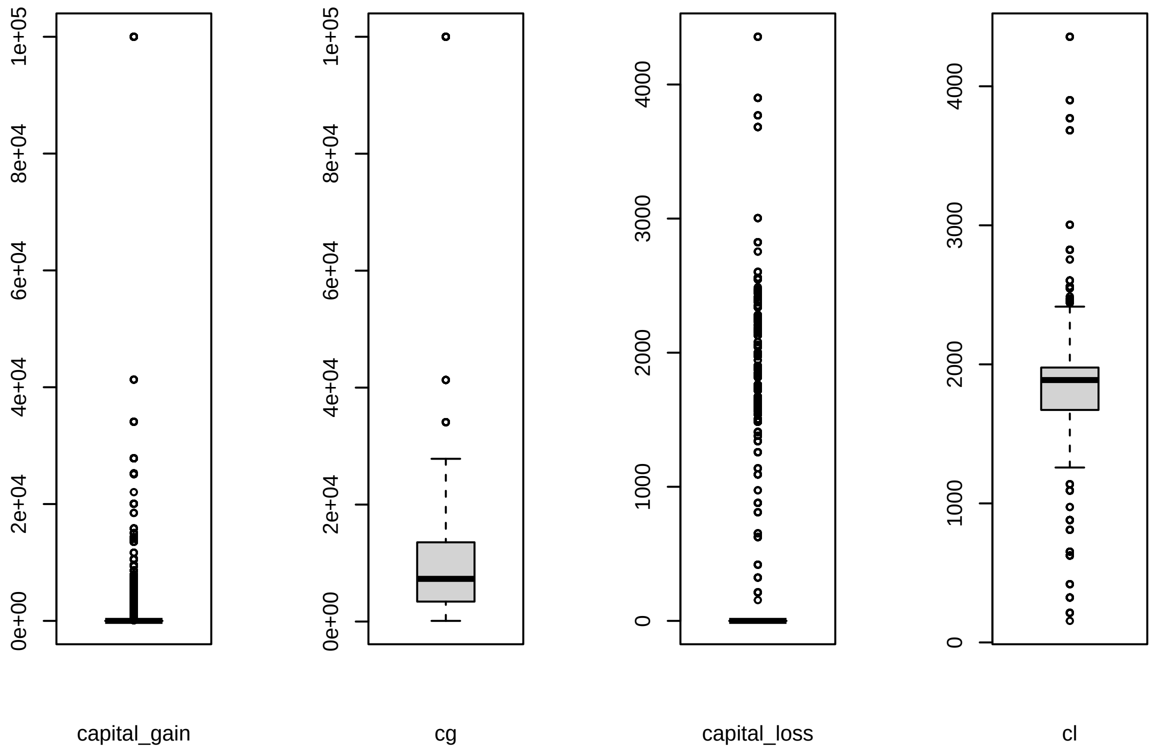
- Don't remove

```r
numeric_cols = names(numeric_df)

par(mfrow = c(1, length(numeric_cols)))
for (i in numeric_cols){
  boxplot(
    df[i],
```

```
    xlab=c(i),
    coef = 1
    )
}
```



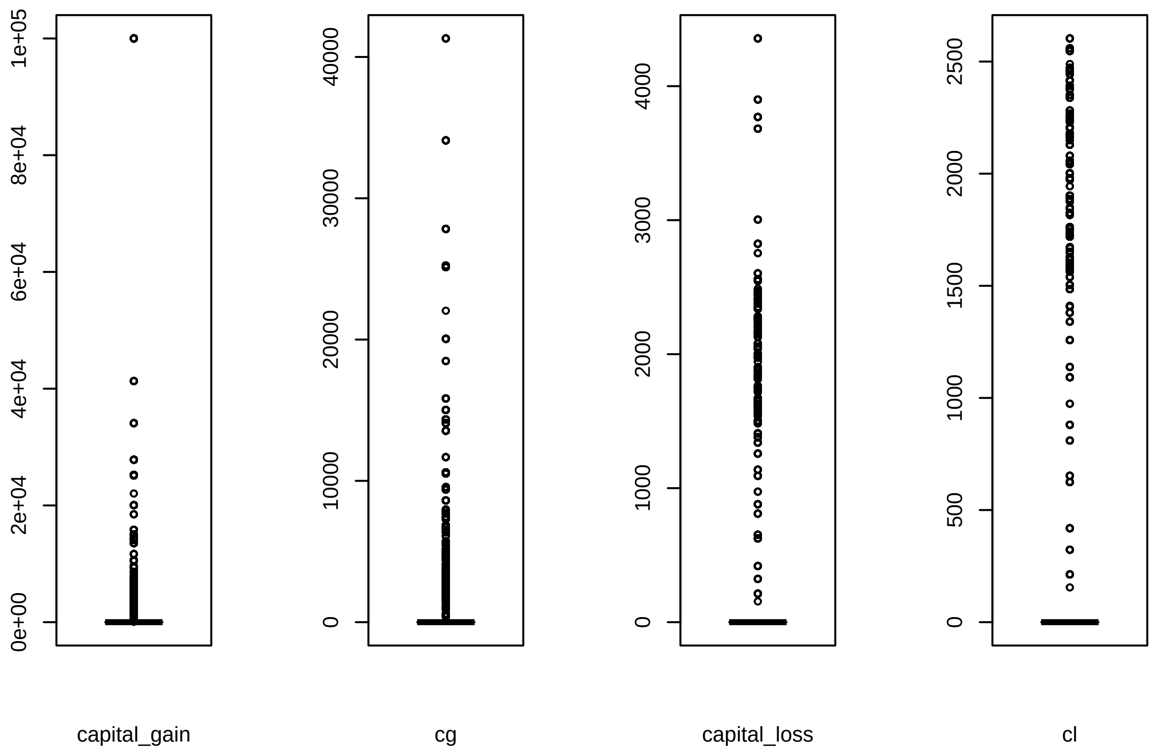

```
#make duplicate column for capital_gain
df$cg = df$capital_gain
#set zero to NA
df$cg[df$cg==0]<-NA

#make duplicate column for capital_loss
df$cl = df$capital_loss
#set zero to NA
df$cl[df$cl==0]<-NA


outlier_plot_cols = c('capital_gain','cg','capital_loss','cl')

par(mfrow = c(1, length(outlier_plot_cols)))
for (i in outlier_plot_cols){
  boxplot(
    df[i],
    xlab=c(i),
    coef = 1
    )
}
```



Boxplot shows before and after having removed zeroes. After removing zeroes, we will remove any values above 2.5 standard deviations above the mean to eliminate very large values.

```
outlier_cols = c('cg','cl')

#remove outliers
#df[outlier_cols] <- data.frame(lapply( df[outlier_cols], function(x) {
# ifelse(x %in% boxplot.stats(x)$out, NA, x) }))


df[outlier_cols] <- data.frame( lapply(df[outlier_cols],
  function(x, na.rm = TRUE)
    {ifelse( (x < 0) | x > (mean(x, na.rm = TRUE) + 2 *sd(x, na.rm = TRUE)), NA, x) }))


#add zeroes back into duplicate col
df$cg [df$capital_gain==0]<-0
df$cl [df$capital_loss==0]<-0

outlier_plot_cols = c('capital_gain','cg','capital_loss','cl')

par(mfrow = c(1, length(outlier_plot_cols)))
for (i in outlier_plot_cols){
  boxplot(
    df[i],
    xlab=c(i),
    coef = 1
```

```
    )
}
```



Plot shows before vs after having removed zeroes, then removed outliers, and added zeroes back.

EDA / Sanity check :

Many values are marked with "?" – so, where are they? and set them to NA.

Remove "?"s or no?

Show count of unique categorical values and NA coun

```
cat_stats <- data.frame(
            unique = sapply(df, n_distinct),
            NA_count = sapply(df, function(x) sum(is.na(x))),
            data_type = sapply(df, class)
            ) %>%
    filter(data_type == 'character') %>%
    select(unique, NA_count) %>%
    arrange(unique)

cat_stats
```

```
##                  unique NA_count
## sex                   2        0
## salary                2        0
## race                  5        0
## relationship          6        0
```

```
## marital_status      7        0
## workclass           9        0
## occupation         15        0
## education          16        0
## native_country     42        0
```

Any strange number of unique values?

- Martital status should be reduced to married vs not married.
- Education can be reduced to lower, middle, high school, some college, college, grad school.
- Any others? Workclass?

```
df = df %>% mutate(
  married = case_when(
    marital_status == ' Divorced' ~ "N",
    marital_status == ' Married-AF-spouse' ~ "Y",
    marital_status == ' Married-civ-spouse' ~ "Y",
    marital_status == ' Married-spouse-absent' ~ "Y",
    marital_status == ' Never-married' ~ "N",
    marital_status == ' Separated' ~ "Y",
    marital_status == ' Widowed' ~ "N"
  )
)


df = df %>% mutate(
  education_simple = case_when(
    education_num <=4 ~ "Below High School",
    education_num >=5 & education_num <=8 ~ "Some High School",
    education_num ==9 ~ "High School",
    education_num ==10 ~ "Some College",
    education_num >=11 & education_num <=13 ~ "College",
    education_num >=14 ~ "Masters or Above"

  )
)
```

Impute the missing values.

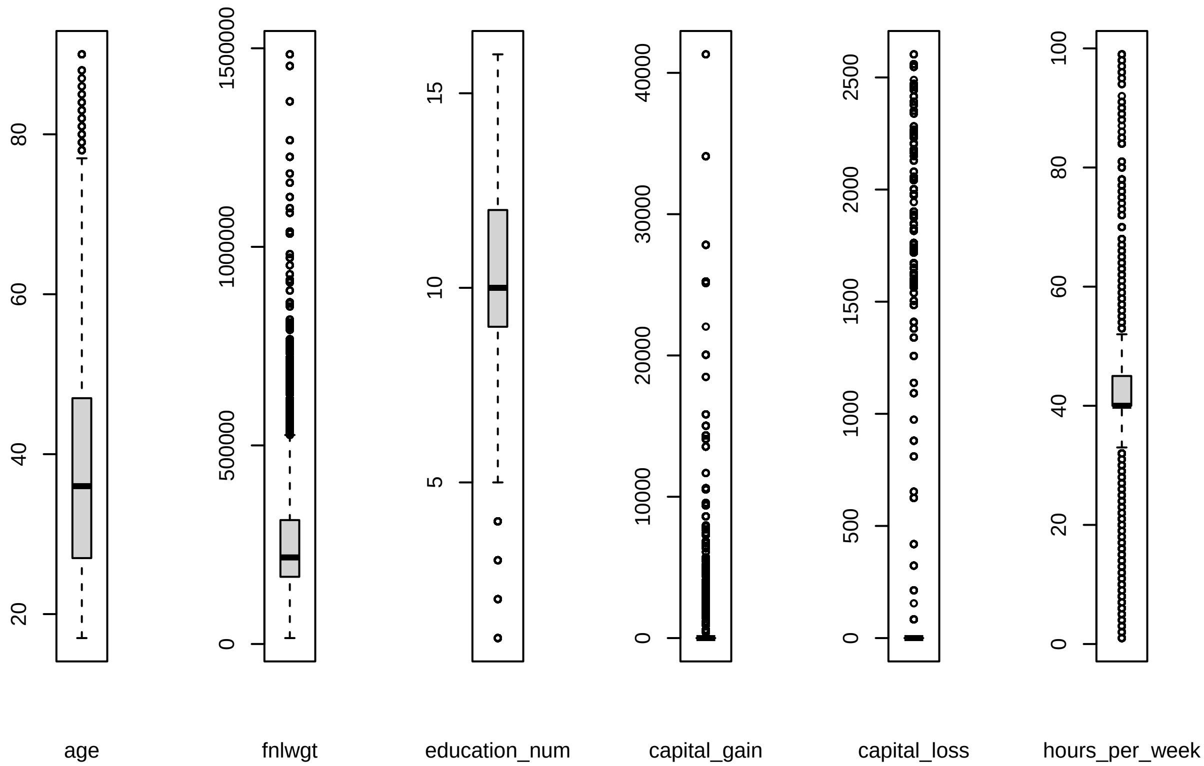
Replaced NAs with the mean.

```
outlier_cols = c('cg','cl')

df[outlier_cols] <- data.frame(lapply(df[outlier_cols], function(x) {
  ifelse(is.na(x), mean(x, na.rm = TRUE), x) }))


# overwrite capital_gain and capital_loss with cg and gl, then drop cg and cl
df$capital_gain = df$cg
df$capital_loss = df$cl
df = select(df, -cg, -cl)
```

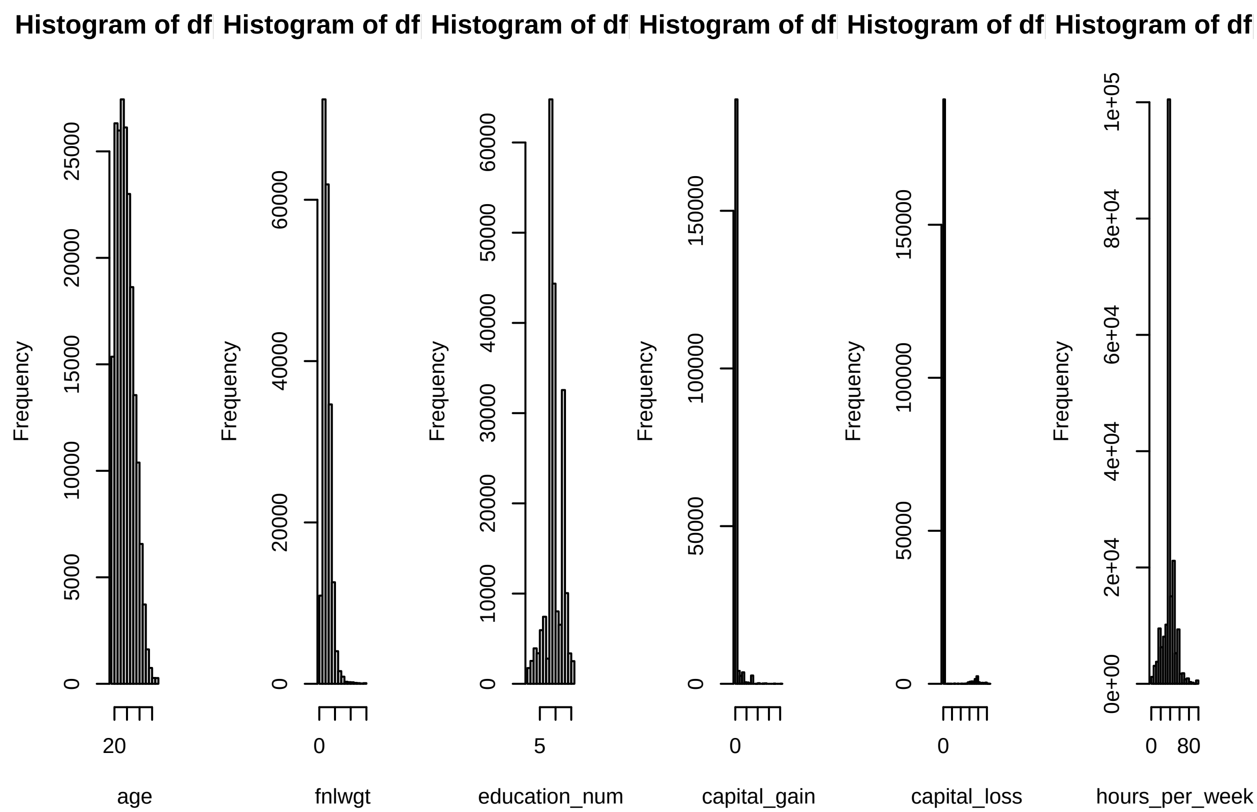
EDA, continued

```
numeric_cols = names(numeric_df)
```

```
par(mfrow = c(1, length(numeric_cols)))
for (i in numeric_cols){
  boxplot(
    df[i],
    xlab=c(i),
    coef = 1
    )}
```




```
numeric_cols = names(numeric_df)

par(mfrow = c(1, length(numeric_cols)))
for (i in numeric_cols){
  hist(
    df[[i]],
    xlab=c(i)
    )
}
```




```r
data.frame(type=sapply(df,class)) %>% filter(type== 'character')
```

```
##                        type
## workclass         character
## education         character
## marital_status    character
## occupation        character
## relationship      character
## race              character
## sex               character
## native_country    character
## salary            character
## married           character
## education_simple character
```

```r
#data.frame(table(df$education_simple))
```

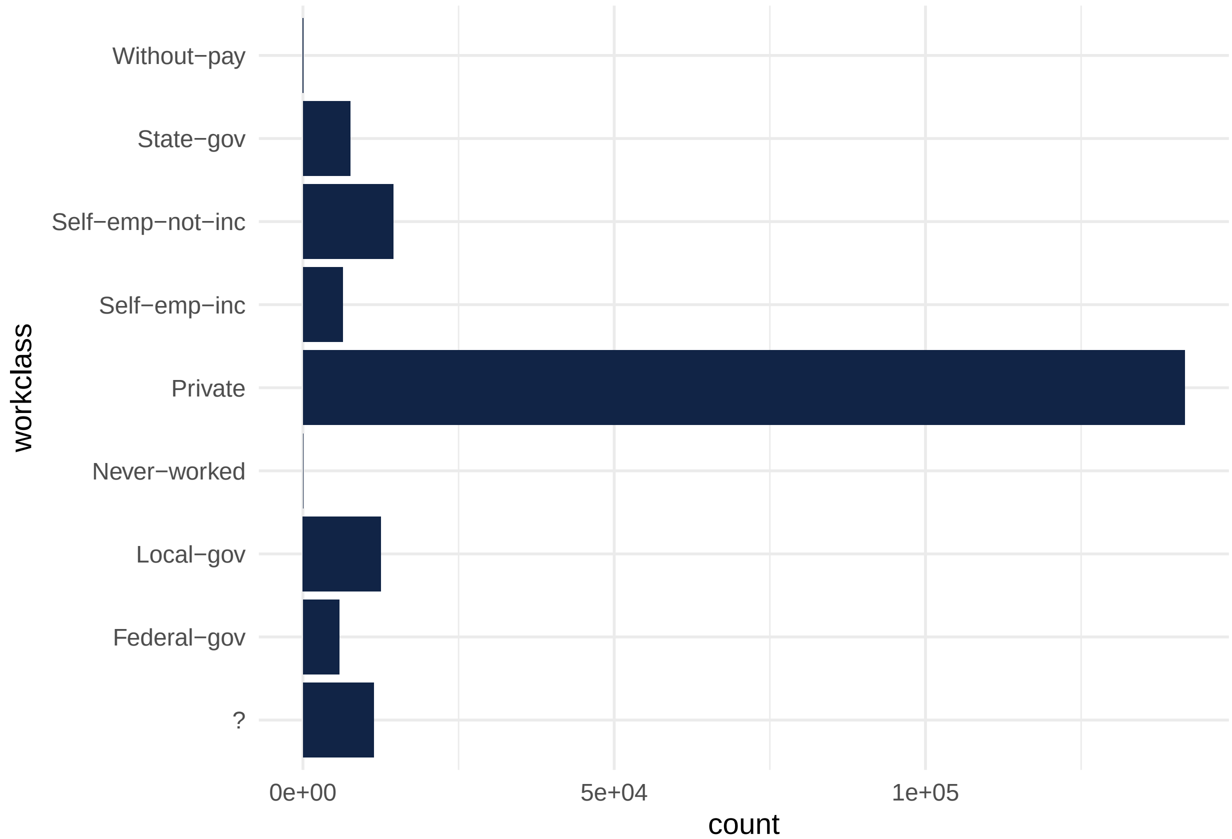
Convert All Categorical Columns to Factors

```r
#df$capital_gain = log( df$capital_gain )

df$workclass = factor(df$workclass)

df$education = factor(df$education, levels=
c(' Preschool' ,' 1st-4th' ,' 5th-6th' ,' 7th-8th' ,' 9th' ,' 10th' ,' 11th' ,' 12th' ,' HS-grad' ,' So

df$education_simple = factor(df$education_simple, levels=
```

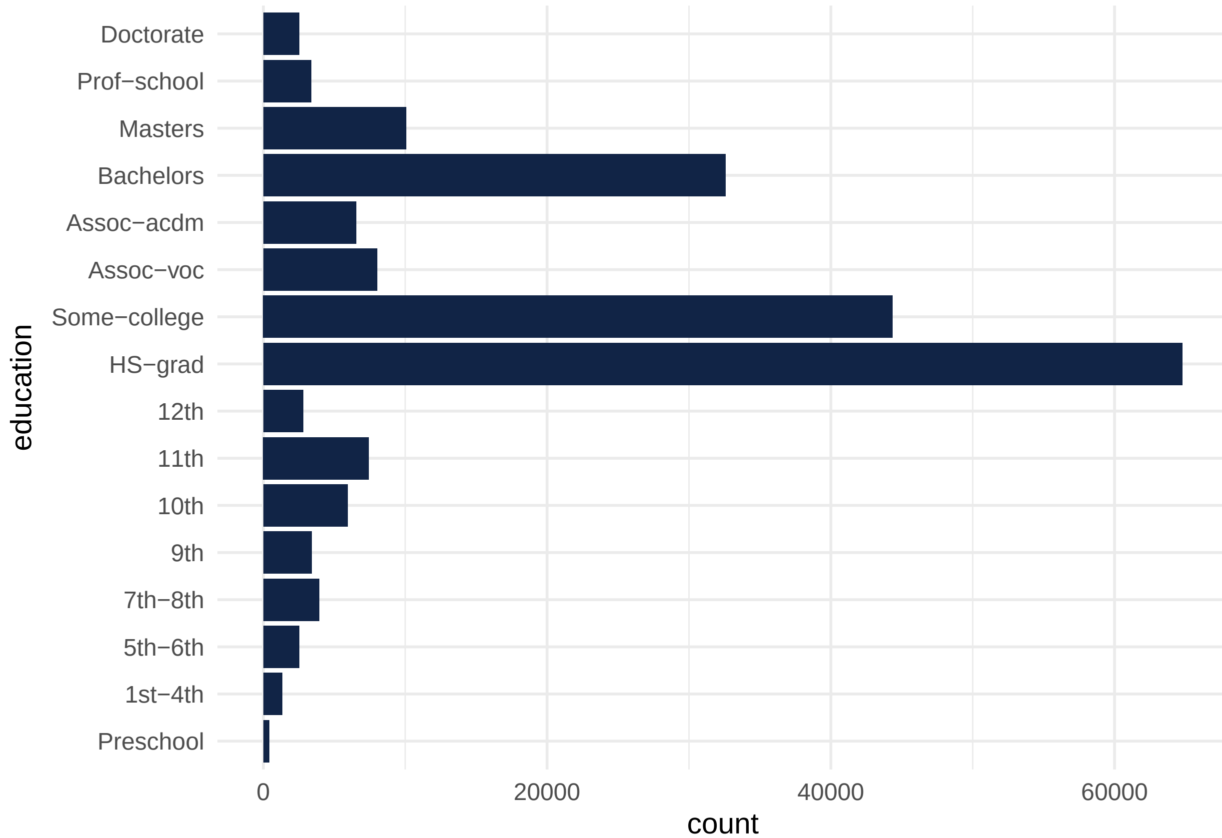
```r
c( 'Below High School','Some High School','High School','Some College','College','Masters or Above') )

df$marital_status = factor(df$marital_status)
df$occupation = factor(df$occupation)
df$relationship = factor(df$relationship)
df$race = factor(df$race)
df$sex = factor(df$sex)
df$native_country = factor(df$native_country)

df [df$salary==' >50K' , 'y'] = 1
df [df$salary==' <=50K' , 'y'] = 0
df$y = factor(df$y)

df$salary = factor(df$salary, levels = c(' <=50K',' >50K'))

df$married = factor(df$married)

#head(df$salary)
```

```r
ggplot(df) +
 aes_string(x = "workclass") +
 geom_bar(fill = "#112446") +
 coord_flip() +
 theme_minimal()
```

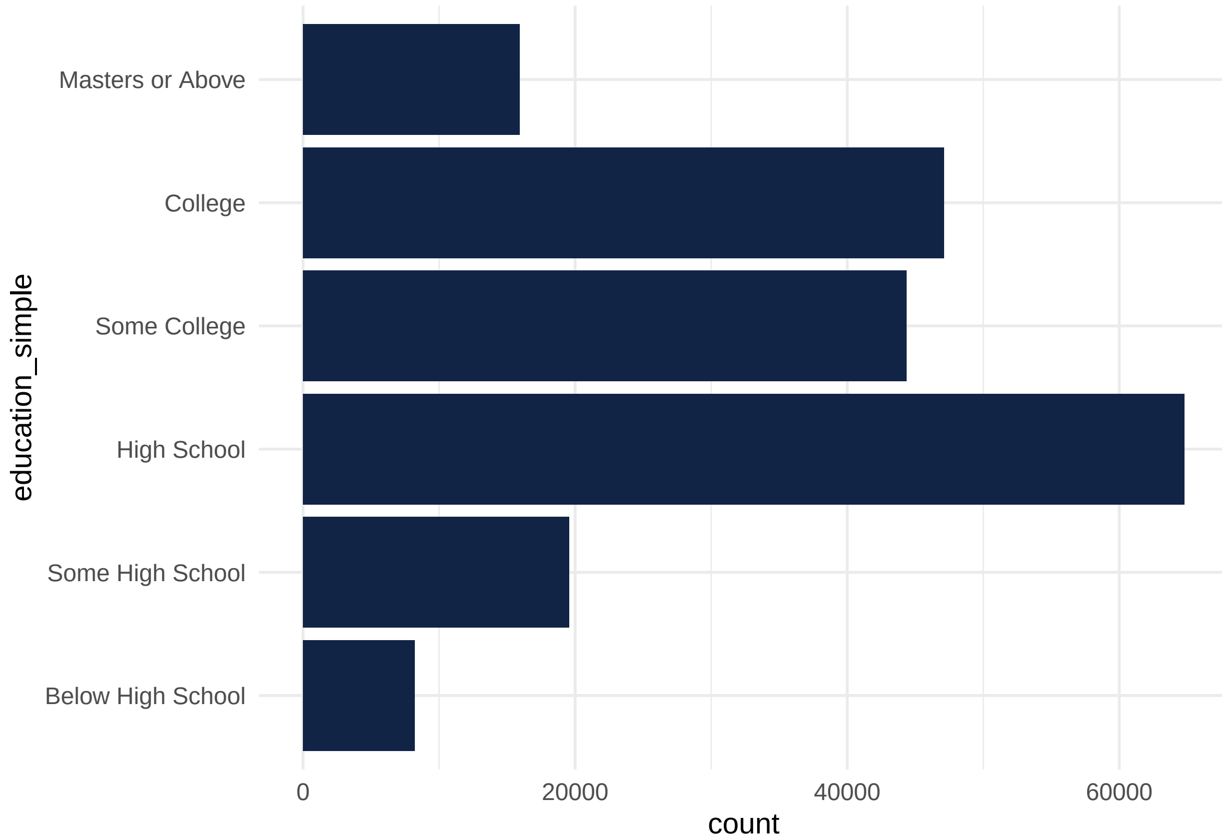
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



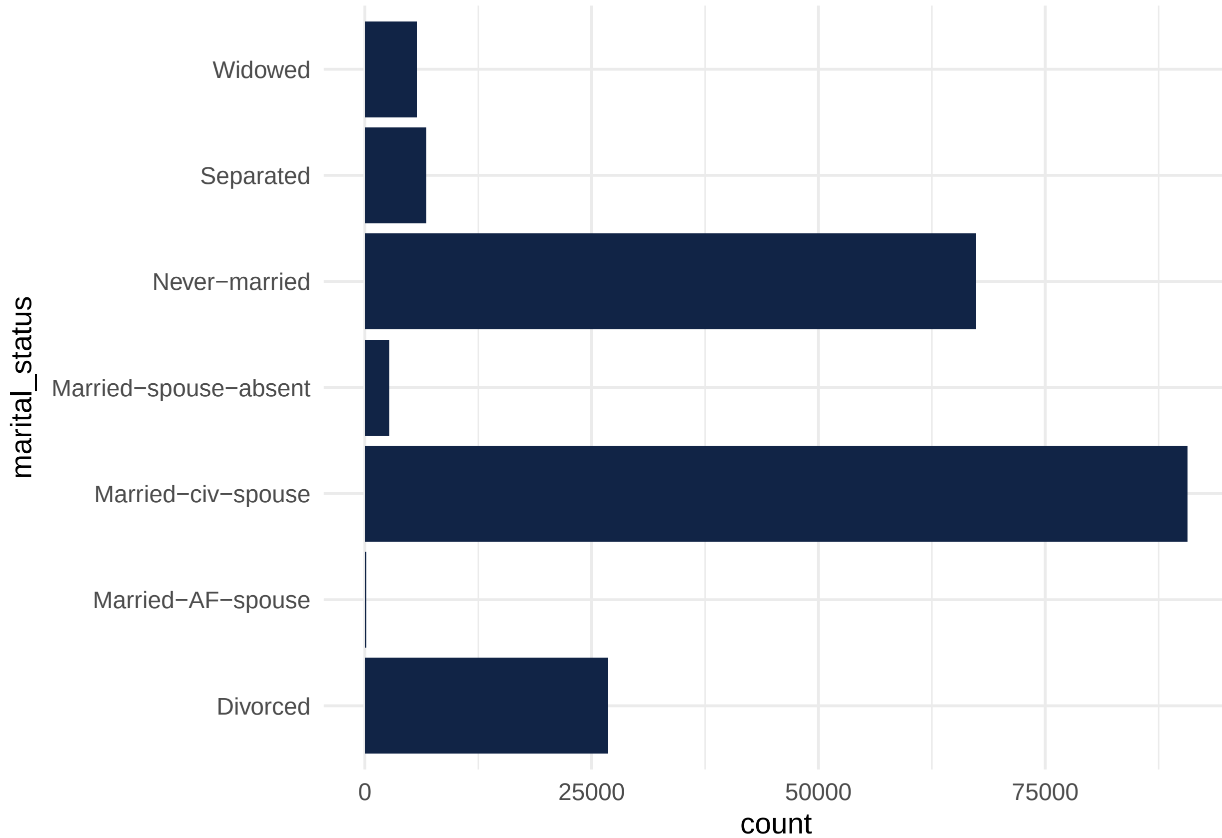

```r
ggplot(df) +
  aes_string(x = "education") +
  geom_bar(fill = "#112446") +
  coord_flip() +
  theme_minimal()
```



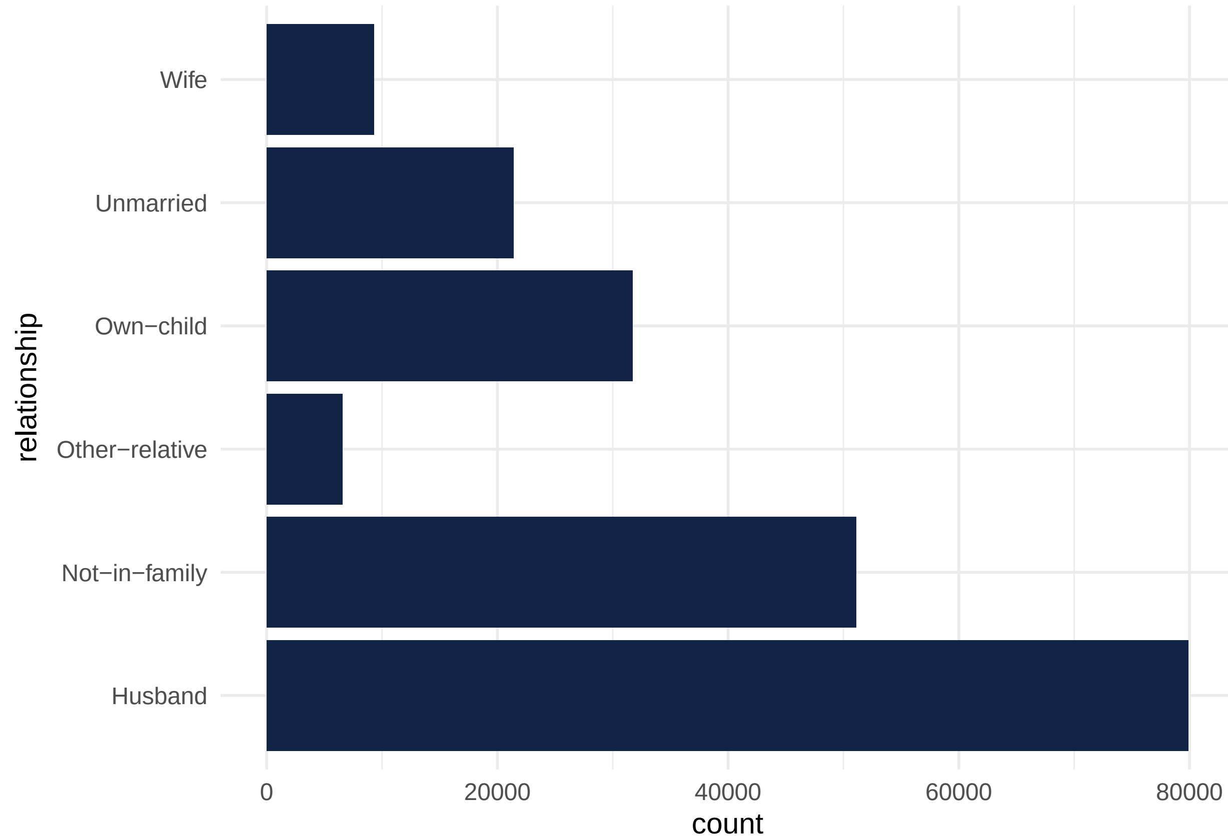

```
ggplot(df) +
 aes_string(x = "education_simple") +
 geom_bar(fill = "#112446") +
 coord_flip() +
 theme_minimal()
```



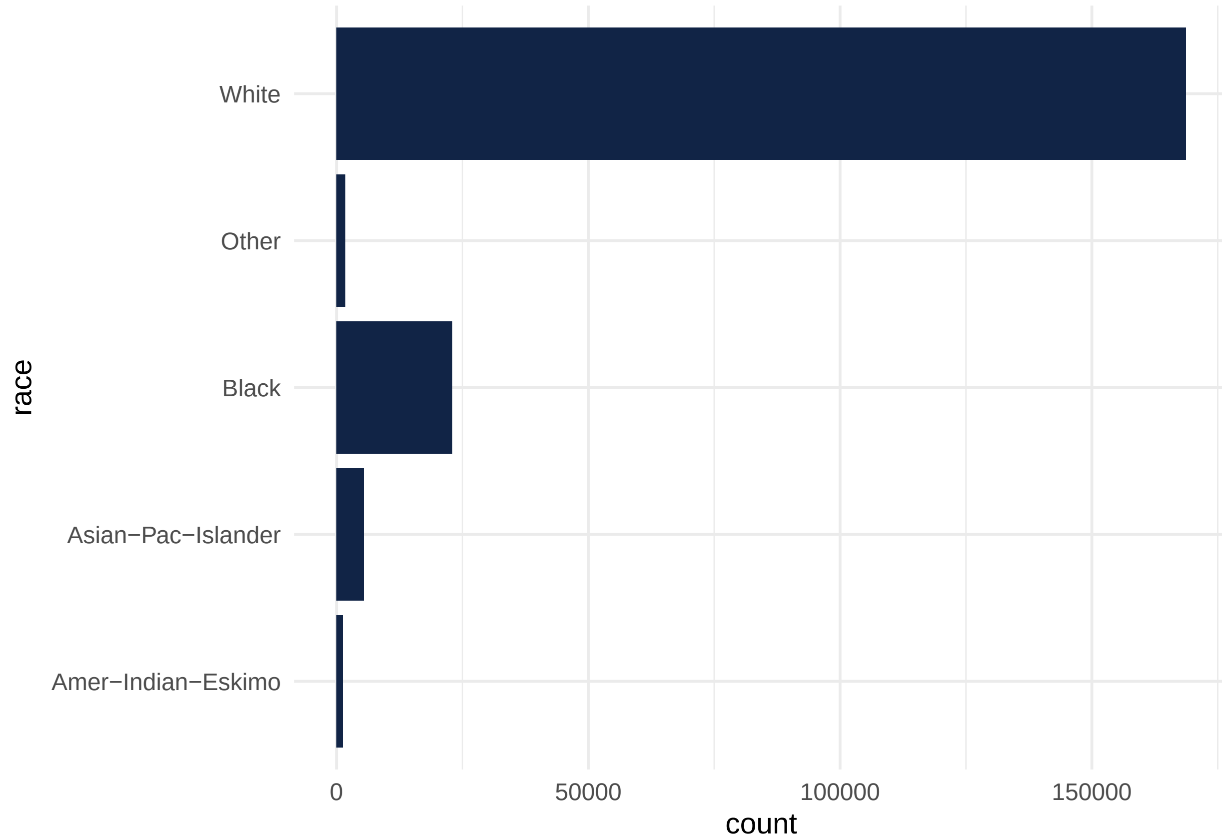

```
ggplot(df) +
 aes_string(x = "marital_status") +
 geom_bar(fill = "#112446") +
 coord_flip() +
 theme_minimal()
```



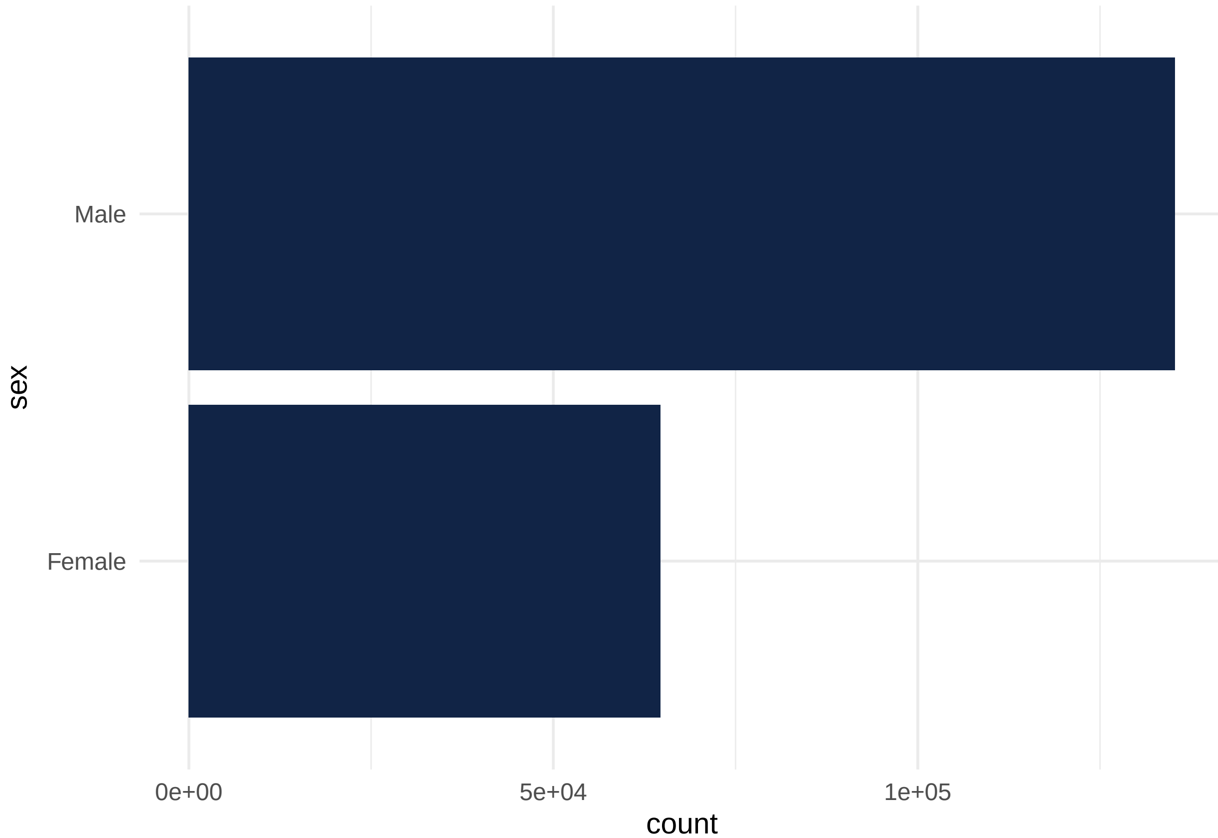

```r
ggplot(df) +
 aes_string(x = "relationship") +
 geom_bar(fill = "#112446") +
 coord_flip() +
 theme_minimal()
```



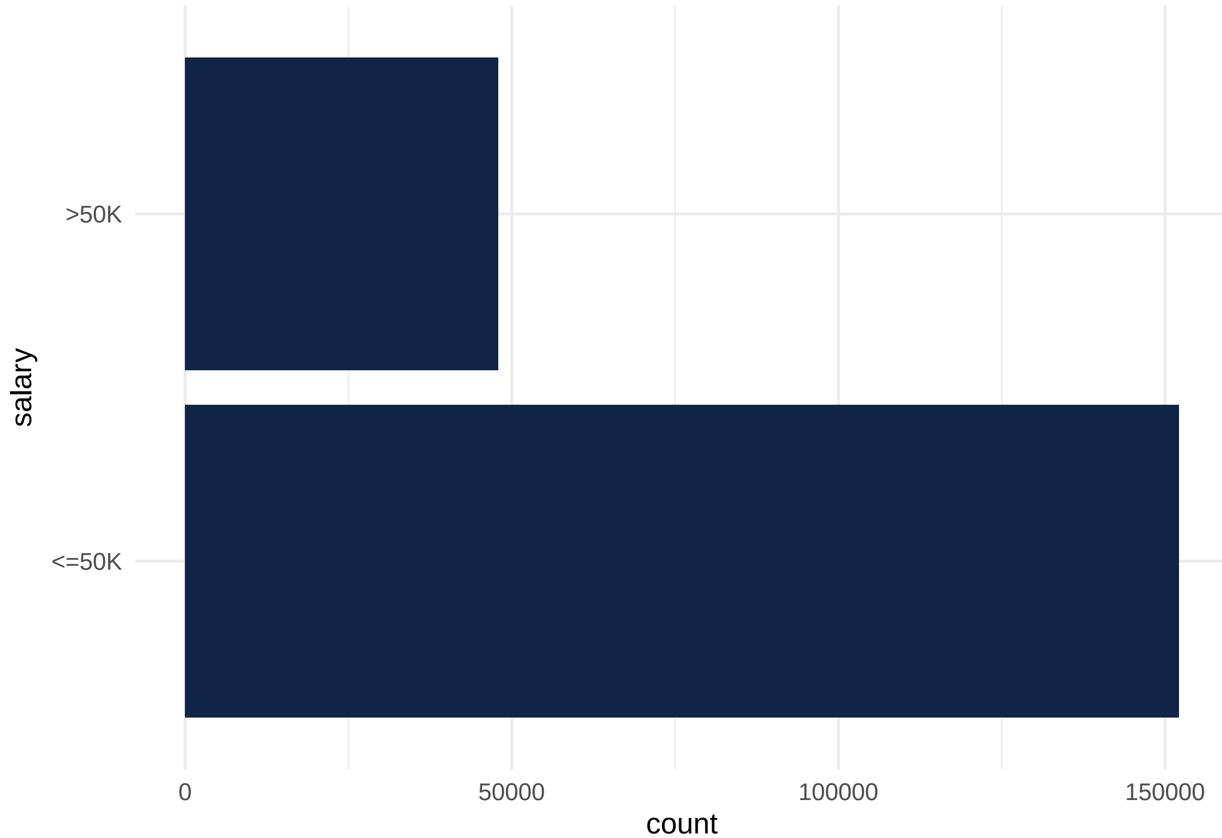

```r
ggplot(df) +
 aes_string(x = "race") +
 geom_bar(fill = "#112446") +
 coord_flip() +
 theme_minimal()
```



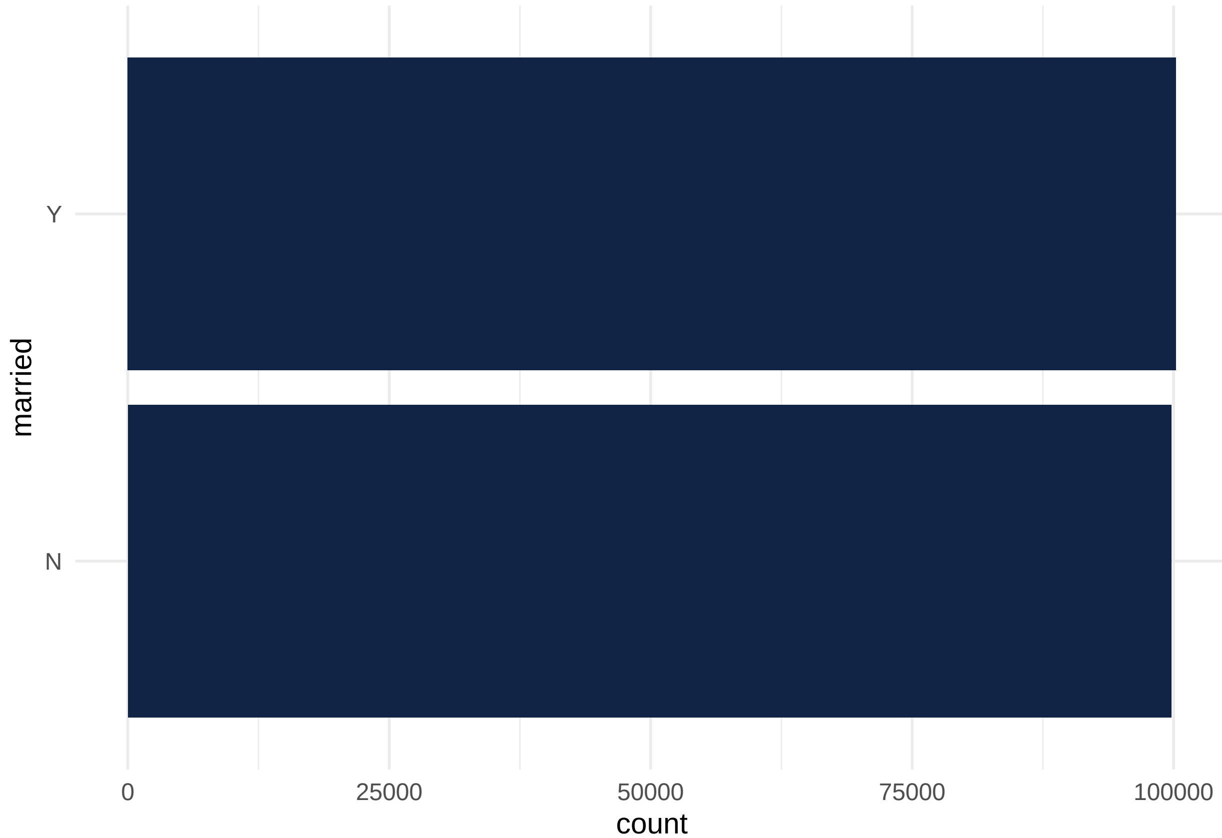

```
ggplot(df) +
 aes_string(x = "sex") +
 geom_bar(fill = "#112446") +
 coord_flip() +
 theme_minimal()
```




```r
ggplot(df) +
  aes_string(x = "salary") +
  geom_bar(fill = "#112446") +
  coord_flip() +
  theme_minimal()
```




```r
ggplot(df) +
 aes_string(x = "married") +
 geom_bar(fill = "#112446") +
 coord_flip() +
 theme_minimal()
```




```
# if you wish to do the model with another software, make sure to use this datat that has been formatte

#write.csv(df,"data/adult_sampled_formatted_pre_model.csv", row.names = FALSE)
```

Models

Split train and test data

```
training.rows <- sample(1:nrow(df),size=0.7*nrow(df))
train_data <- df[training.rows,]
test_data <- df[-training.rows,]


nrow(df)
```

```
## [1] 200000
```

```
print(nrow(train_data) )
```

```
## [1] 140000
```

```
print(nrow(test_data) )
```

```
## [1] 60000
```

```
#this is where we will store the metrics for each model
model_stats = data.frame(metric = c("Accuracy","True Positive Rate","False Positive Rate","Specificity"
```

```r
#our variables

#data.frame(type=sapply(df,class))
c(names(df))
```

```
##  [1] "age"              "workclass"        "fnlwgt"           "education"
##  [5] "education_num"    "marital_status"   "occupation"       "relationship"
##  [9] "race"             "sex"              "capital_gain"     "capital_loss"
## [13] "hours_per_week"   "native_country"   "salary"           "married"
## [17] "education_simple" "y"
```

Naïve Bayes Model

Which variables are important?

Let's start with:

age race native_country

married relationship

workclass occupation hours_per_week

education education_simple

capital_gain capital_loss

age+race+native_country+married+relationship+workclass+occupation+hours_per_week+education_simple+capital_gain-

```r
NBmodel <- naive_bayes(salary ~ age+race+married+relationship+workclass+occupation+hours_per_week+educa
                      data = train_data)
summary(NBmodel)
```

```
##
## ================================== Naive Bayes ==================================
##
## - Call: naive_bayes.formula(formula = salary ~ age + race + married +      relationship + workclass +
## - Laplace: 0
## - Classes: 2
## - Samples: 140000
## - Features: 10
## - Conditional distributions:
##     - Bernoulli: 1
##     - Categorical: 5
##     - Gaussian: 4
## - Prior probabilities:
##     -  <=50K: 0.761
##     -  >50K: 0.239
##
## ---------------------------------------------------------------------------------
```

Score the validation data (predict) using the model. Produce a confusion table and an ROC curve for the scored validation data.

```r
#classification matrix
predNB = predict(NBmodel, test_data, type="prob")[,2] #This is the probability that the score is a "goo

pred = predNB
pred[pred>=.5] = 1
pred[pred!=1] = 0
```

```r
classMatrix = table(pred,test_data$salary) #first variable is by row, the second is by column
print("Classification Matrix:")
```

```
## [1] "Classification Matrix:"
```

```r
print(classMatrix)
```

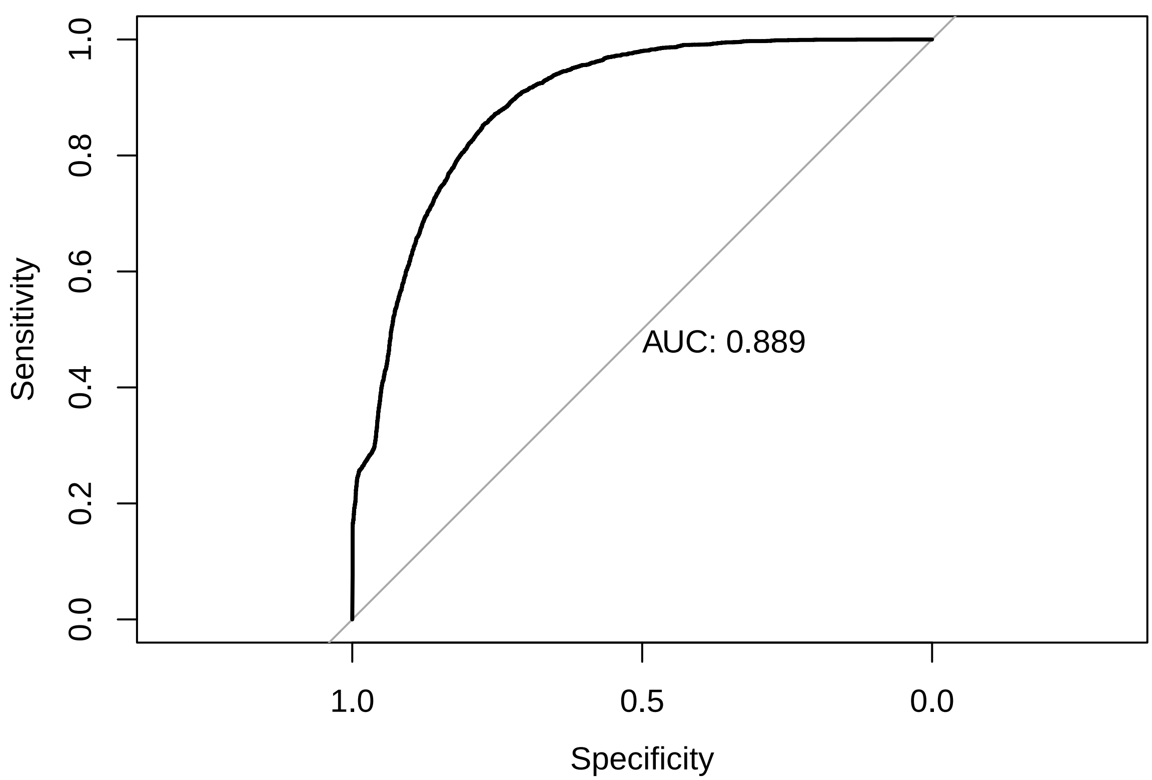
```
##
## pred  <=50K  >50K
##    0  42194  6791
##    1   3345  7670
```

ROC Curve

```r
roc_curve = roc(test_data$salary,predNB,plot = TRUE, print.auc = TRUE)
```

```
## Setting levels: control =  <=50K, case =  >50K
```

```
## Setting direction: controls < cases
```




```r
AUC<- auc(roc_curve)
```

Calculate: accuracy, misclassification rate, true positive rate, false positive rate, specificity, precision, and prevalence.

```r
#Accuracy score?
print("Accuracy:")
```

```
## [1] "Accuracy:"
```

```r
accuracy = sum(diag(classMatrix))/sum(classMatrix)
print(accuracy)
```

```
## [1] 0.8310667
```

```r
cat("\n")
```

```r
print("Misclassification Rate")
```

```
## [1] "Misclassification Rate"
```

```r
print(1-accuracy)
```

```
## [1] 0.1689333
```

```r
cat("\n")
```

```r
true_negative<-classMatrix[1,1]
false_positive<-classMatrix[2,1]
true_positive<-classMatrix[2,2]
false_negative<-classMatrix[1,2]


print("True Positive Rate")
```

```
## [1] "True Positive Rate"
```

```r
true_positive_rate = (true_positive)/ (true_positive + false_negative)
print(true_positive_rate)
```

```
## [1] 0.5303921
```

```r
cat("\n")
```

```r
print("False Positive Rate")
```

```
## [1] "False Positive Rate"
```

```r
false_positive_rate = (false_positive)/ (false_positive + true_negative)
print(false_positive_rate)
```

```
## [1] 0.07345352
```

```r
cat("\n")
```

```r
print("Specificity:")
```

```
## [1] "Specificity:"
```

```r
specificity<-true_negative/(true_negative+false_negative)
print(specificity)
```

```
## [1] 0.8613657
```

```r
cat("\n")
```

```r
print("Precision:")
```

```
## [1] "Precision:"
```

```r
precision<-true_positive/(true_positive+false_positive)
print(precision)
```

```
## [1] 0.6963232
```

```r
cat("\n")
```

```r
print("Prevalence:")
```

```
## [1] "Prevalence:"
```

```r
prevalence = (true_positive + false_negative)/(true_negative+false_positive+true_positive+false_negativ
print(prevalence)
```

```
## [1] 0.2410167
```

```r
#add to model_stats
model_stats$naiveBayes = c(accuracy, true_positive_rate, false_positive_rate, specificity, precision, 1
model_stats$naiveBayes = round(model_stats$naiveBayes,2)
```

Logit Model

```r
LRmodel = glm(salary~
            age+race+married+relationship+workclass+occupation+hours_per_week+education_simple+capital_g
            train_data, family = "binomial")

summary(LRmodel)
```

```
##
## Call:
## glm(formula = salary ~ age + race + married + relationship +
##     workclass + occupation + hours_per_week + education_simple +
##     capital_gain + capital_loss, family = "binomial", data = train_data)
##
## Coefficients: (1 not defined because of singularities)
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -7.413e+00  1.759e-01 -42.157  < 2e-16 ***
## age                             2.777e-02  7.681e-04  36.154  < 2e-16 ***
## race Asian-Pac-Islander         6.838e-01  1.419e-01   4.818 1.45e-06 ***
## race Black                      6.068e-01  1.361e-01   4.460 8.19e-06 ***
## race Other                      3.596e-01  1.781e-01   2.019 0.043466 *
## race White                      7.777e-01  1.330e-01   5.846 5.02e-09 ***
## marriedY                        6.441e-01  5.165e-02  12.471  < 2e-16 ***
## relationship Not-in-family     -1.590e+00  5.212e-02 -30.497  < 2e-16 ***
## relationship Other-relative    -1.951e+00  9.739e-02 -20.029  < 2e-16 ***
## relationship Own-child         -2.689e+00  7.958e-02 -33.789  < 2e-16 ***
## relationship Unmarried         -1.954e+00  5.751e-02 -33.975  < 2e-16 ***
## relationship Wife               5.534e-01  3.346e-02  16.538  < 2e-16 ***
## workclass Federal-gov           9.712e-01  7.384e-02  13.154  < 2e-16 ***
## workclass Local-gov             3.332e-01  6.697e-02   4.976 6.48e-07 ***
## workclass Never-worked         -9.380e+00  6.863e+01  -0.137 0.891288
## workclass Private               5.014e-01  5.995e-02   8.364  < 2e-16 ***
## workclass Self-emp-inc          7.359e-01  7.265e-02  10.128  < 2e-16 ***
## workclass Self-emp-not-inc      1.722e-01  6.586e-02   2.614 0.008944 **
## workclass State-gov             2.485e-01  7.258e-02   3.424 0.000617 ***
## workclass Without-pay          -1.097e+01  5.624e+01  -0.195 0.845288
## occupation Adm-clerical         8.640e-02  4.760e-02   1.815 0.069518 .
## occupation Armed-Forces        -2.531e-01  6.929e-01  -0.365 0.714929
## occupation Craft-repair         2.645e-01  4.149e-02   6.375 1.83e-10 ***
## occupation Exec-managerial      9.018e-01  4.247e-02  21.232  < 2e-16 ***
```

```
## occupation Farming-fishing       -1.270e+00  7.925e-02 -16.031  < 2e-16 ***
## occupation Handlers-cleaners     -5.548e-01  7.062e-02  -7.856 3.97e-15 ***
## occupation Machine-op-inspct     -1.521e-01  5.125e-02  -2.969 0.002992 **
## occupation Other-service         -7.205e-01  5.983e-02 -12.042  < 2e-16 ***
## occupation Priv-house-serv       -3.775e+00  8.854e-01  -4.264 2.01e-05 ***
## occupation Prof-specialty         7.945e-01  4.498e-02  17.662  < 2e-16 ***
## occupation Protective-serv        8.161e-01  6.256e-02  13.044  < 2e-16 ***
## occupation Sales                  5.142e-01  4.380e-02  11.740  < 2e-16 ***
## occupation Tech-support           8.583e-01  5.737e-02  14.962  < 2e-16 ***
## occupation Transport-moving              NA         NA      NA       NA
## hours_per_week                    3.487e-02  8.048e-04  43.331  < 2e-16 ***
## education_simpleSome High School  7.923e-01  8.424e-02   9.405  < 2e-16 ***
## education_simpleHigh School       1.622e+00  7.476e-02  21.693  < 2e-16 ***
## education_simpleSome College      2.023e+00  7.585e-02  26.674  < 2e-16 ***
## education_simpleCollege           2.565e+00  7.575e-02  33.860  < 2e-16 ***
## education_simpleMasters or Above  3.193e+00  7.967e-02  40.081  < 2e-16 ***
## capital_gain                      3.293e-04  5.024e-06  65.535  < 2e-16 ***
## capital_loss                      6.253e-04  1.890e-05  33.078  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 153970  on 139999  degrees of freedom
## Residual deviance:  88999  on 139959  degrees of freedom
## AIC: 89081
##
## Number of Fisher Scoring iterations: 12
```

Which variables can we reject the null hypothesis that their coefficients equal zero?

```r
varImp(LRmodel) %>% arrange(-Overall)
```

```
##                                     Overall
## capital_gain                     65.5345925
## hours_per_week                   43.3314474
## education_simpleMasters or Above 40.0812663
## age                              36.1536095
## relationship Unmarried           33.9748220
## education_simpleCollege          33.8595449
## relationship Own-child           33.7888527
## capital_loss                     33.0781061
## relationship Not-in-family       30.4968771
## education_simpleSome College     26.6738637
## education_simpleHigh School      21.6932622
## occupation Exec-managerial       21.2315053
## relationship Other-relative      20.0285406
## occupation Prof-specialty        17.6615577
## relationship Wife                16.5376407
## occupation Farming-fishing       16.0311257
## occupation Tech-support          14.9622017
## workclass Federal-gov            13.1537428
## occupation Protective-serv       13.0440601
## marriedY                         12.4713140
## occupation Other-service         12.0423781
```

```
## occupation Sales                    11.7397202
## workclass Self-emp-inc              10.1282295
## education_simpleSome High School  9.4051980
## workclass Private                    8.3638368
## occupation Handlers-cleaners         7.8558743
## occupation Craft-repair              6.3748255
## race White                           5.8463932
## workclass Local-gov                  4.9762693
## race Asian-Pac-Islander              4.8176246
## race Black                           4.4600695
## occupation Priv-house-serv           4.2638833
## workclass State-gov                  3.4239576
## occupation Machine-op-inspct         2.9685299
## workclass Self-emp-not-inc           2.6141860
## race Other                           2.0192016
## occupation Adm-clerical              1.8150413
## occupation Armed-Forces              0.3652439
## workclass Without-pay                0.1951341
## workclass Never-worked               0.1366748
```

Score the validation data (predict) using the logit model.

```
#classification matrix
predLR = predict(LRmodel,test_data,type="response") #This is the probability that the score is a "good
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
pred = predLR
pred[pred>=.5] = 1
pred[pred!=1] = 0

classMatrix = table(pred,test_data$salary) #first variable is by row, the second is by column
print("Classification Matrix:")
```

```
## [1] "Classification Matrix:"
```

```
print(classMatrix)
```

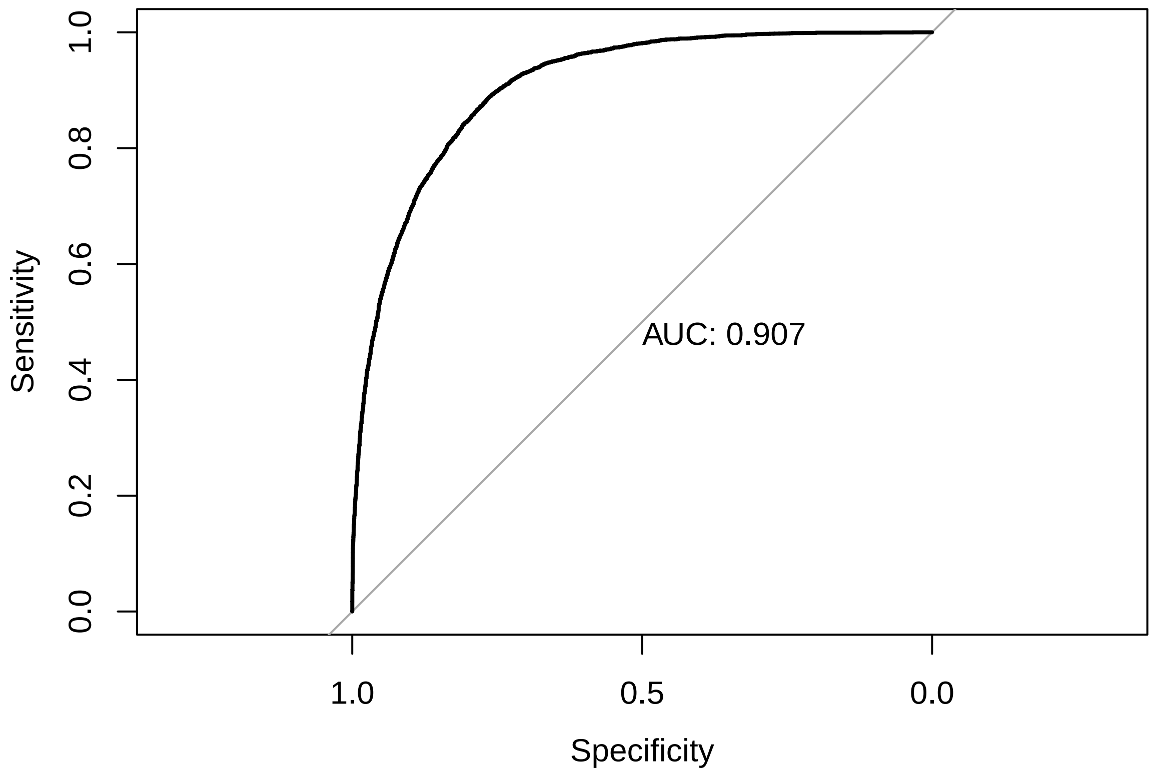
```
##
## pred  <=50K  >50K
##    0  42225  5512
##    1   3314  8949
```

ROC Curve

```
roc_curve = roc(test_data$salary,predLR,plot = TRUE, print.auc = TRUE)
```

```
## Setting levels: control =  <=50K, case =  >50K
```

```
## Setting direction: controls < cases
```




```
AUC<- auc(roc_curve)
```

Calculate: accuracy, misclassification rate, true positive rate, false positive rate, specificity, precision, and prevalence.

```
#Accuracy score?
print("Accuracy:")
```

```
## [1] "Accuracy:"
```

```
accuracy = sum(diag(classMatrix))/sum(classMatrix)
print(accuracy)
```

```
## [1] 0.8529
```

```
cat("\n")
```

```
print("Misclassification Rate")
```

```
## [1] "Misclassification Rate"
```

```
print(1-accuracy)
```

```
## [1] 0.1471
```

```
cat("\n")
```

```
true_negative<-classMatrix[1,1]
false_positive<-classMatrix[2,1]
true_positive<-classMatrix[2,2]
false_negative<-classMatrix[1,2]
```

```r
print("True Positive Rate")
```

```
## [1] "True Positive Rate"
```

```r
true_positive_rate = (true_positive)/ (true_positive + false_negative)
print(true_positive_rate)
```

```
## [1] 0.6188369
```

```r
cat("\n")
```

```r
print("False Positive Rate")
```

```
## [1] "False Positive Rate"
```

```r
false_positive_rate = (false_positive)/ (false_positive + true_negative)
print(false_positive_rate)
```

```
## [1] 0.07277279
```

```r
cat("\n")
```

```r
print("Specificity:")
```

```
## [1] "Specificity:"
```

```r
specificity<-true_negative/(true_negative+false_negative)
print(specificity)
```

```
## [1] 0.884534
```

```r
cat("\n")
```

```r
print("Precision:")
```

```
## [1] "Precision:"
```

```r
precision<-true_positive/(true_positive+false_positive)
print(precision)
```

```
## [1] 0.7297562
```

```r
cat("\n")
```

```r
print("Prevalence:")
```

```
## [1] "Prevalence:"
```

```r
prevalence = (true_positive + false_negative)/(true_negative+false_positive+true_positive+false_negativ
print(prevalence)
```

```
## [1] 0.2410167
```

```r
#add to model_stats
model_stats$Logistic = c(accuracy, true_positive_rate, false_positive_rate, specificity, precision, 1)
model_stats$Logistic = round(model_stats$Logistic,2)
```

Tree Model (CART)

```r
adultTree = rpart(y ~ age+race+married+relationship+workclass+occupation+hours_per_week+education_simpl
```

```r
#summary(adultTree)
```

Variable/feature importance

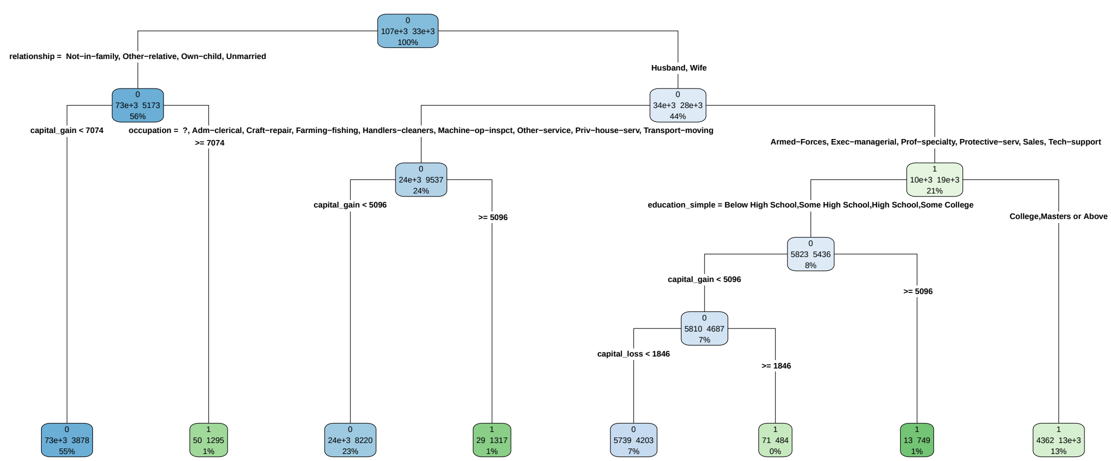
```r
varImp(adultTree) %>% arrange(-Overall)
```

```
##                       Overall
## capital_gain      13934.91666
## education_simple 11164.50423
## relationship     10419.02010
## occupation       10270.95033
## married           8558.07969
## capital_loss      2241.66083
## hours_per_week    1858.99375
## age               1600.10235
## workclass           85.70589
## race                 0.00000
```

```r
write.csv(varImp(adultTree) %>% arrange(-Overall), 'data/variable_importance.csv')
```

Plot of the decision tree.

```r
rpart.plot(adultTree, type=4, extra=101, fallen.leaves = TRUE)
```



Score the validation data (predict) using the CART model, produce a confusion table and an ROC curve.

```r
PredictCART = predict(adultTree, newdata= test_data,type='prob')[,2]

pred = PredictCART
```

```r
pred[pred>=.5] = 1
pred[pred!=1] = 0

classMatrix = table(pred,test_data$salary) #first variable is by row, the second is by column
print("Classification Matrix:")
```

```
## [1] "Classification Matrix:"
```

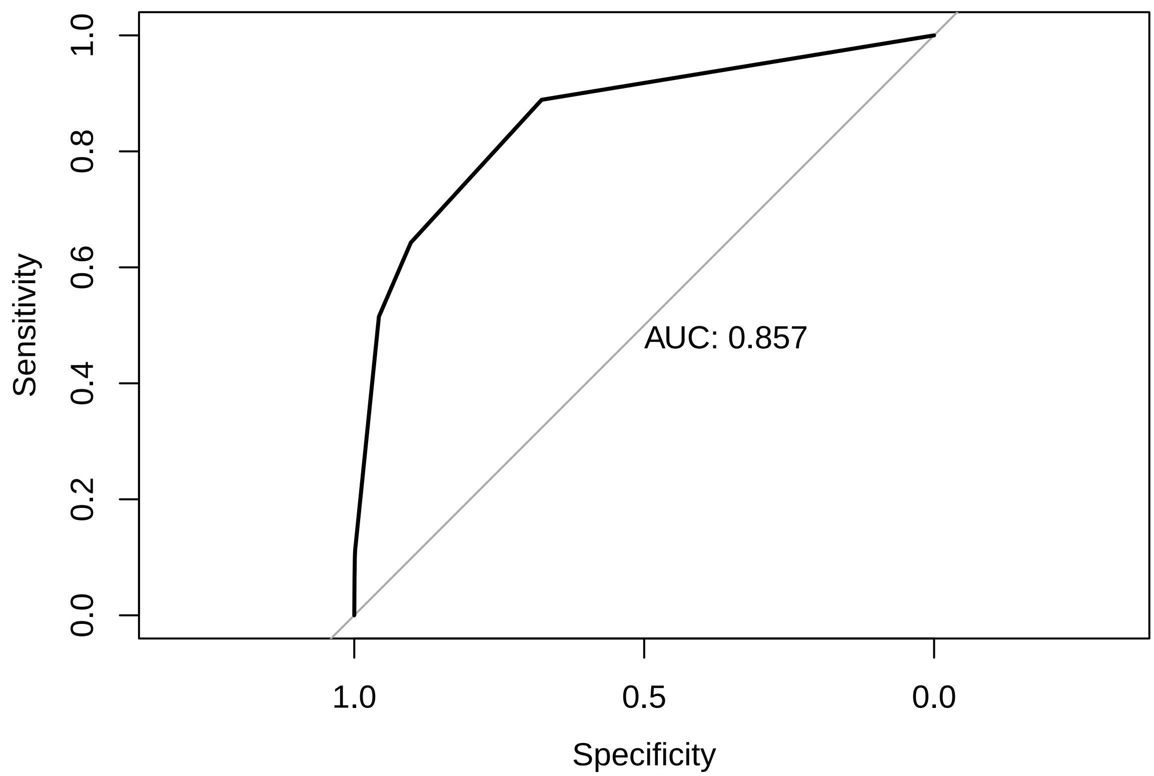
```r
print(classMatrix)
```

```
##
## pred  <=50K  >50K
##    0  43604  7013
##    1   1935  7448
```

```r
roc_curve = roc(test_data$salary,PredictCART,plot = TRUE, print.auc = TRUE)
```

```
## Setting levels: control =  <=50K, case =  >50K
```

```
## Setting direction: controls < cases
```




```r
AUC<- auc(roc_curve)
```

Calculate: accuracy, misclassification rate, true positive rate, false positive rate, specificity, precision, and prevalence.

```r
#Accuracy score?
print("Accuracy:")
```

```
## [1] "Accuracy:"
```

```
accuracy = sum(diag(classMatrix))/sum(classMatrix)
print(accuracy)
```

```
## [1] 0.8508667
```

```
cat("\n")
```

```
print("Misclassification Rate")
```

```
## [1] "Misclassification Rate"
```

```
print(1-accuracy)
```

```
## [1] 0.1491333
```

```
cat("\n")
```

```
true_negative<-classMatrix[1,1]
false_positive<-classMatrix[2,1]
true_positive<-classMatrix[2,2]
false_negative<-classMatrix[1,2]


print("True Positive Rate")
```

```
## [1] "True Positive Rate"
```

```
true_positive_rate = (true_positive)/ (true_positive + false_negative)
print(true_positive_rate)
```

```
## [1] 0.5150405
```

```
cat("\n")
```

```
print("False Positive Rate")
```

```
## [1] "False Positive Rate"
```

```
false_positive_rate = (false_positive)/ (false_positive + true_negative)
print(false_positive_rate)
```

```
## [1] 0.04249105
```

```
cat("\n")
```

```
print("Specificity:")
```

```
## [1] "Specificity:"
```

```
specificity<-true_negative/(true_negative+false_negative)
print(specificity)
```

```
## [1] 0.8614497
```

```
cat("\n")
```

```
print("Precision:")
```

```
## [1] "Precision:"
```

```r
precision<-true_positive/(true_positive+false_positive)
print(precision)
```

```
## [1] 0.793776
```

```r
cat("\n")
```

```r
print("Prevalence:")
```

```
## [1] "Prevalence:"
```

```r
prevalence = (true_positive + false_negative)/(true_negative+false_positive+true_positive+false_negativ
print(prevalence)
```

```
## [1] 0.2410167
```

```r
#add to model_stats
model_stats$CART = c(accuracy, true_positive_rate, false_positive_rate, specificity, precision, 1)
model_stats$CART = round(model_stats$CART,2)
```

```r
#Random Forest
modelRF <- randomForest(salary~age+race+married+relationship+workclass+occupation+hours_per_week+educati
```

```r
varImp(modelRF) %>% arrange(-Overall)
```

```
##                     Overall
## relationship      6446.5860
## age               5773.4336
## capital_gain      5517.8293
## occupation        5031.0385
## education_simple  4191.8906
## married           3541.9132
## hours_per_week    3472.9802
## workclass         1732.7218
## capital_loss      1682.3496
## race               837.3012
```

```r
PredRF = predict(modelRF, newdata= test_data,type='prob')[,2]
pred = PredRF

pred[pred>=.5] = 1
pred[pred!=1] = 0

classMatrix = table(pred,test_data$salary) #first variable is by row, the second is by column
print("Classification Matrix:")
```

```
## [1] "Classification Matrix:"
```

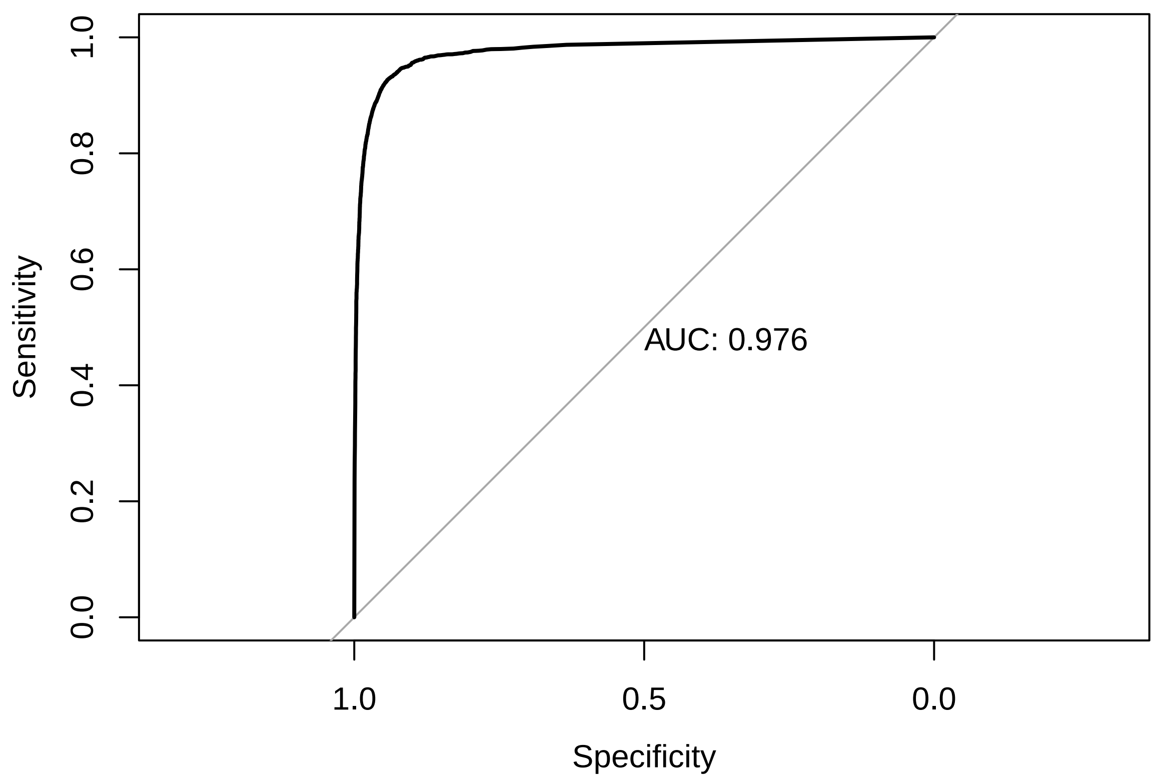
```r
print(classMatrix)
```

```
##
## pred  <=50K  >50K
##    0  44329  2109
##    1   1210 12352
```

```r
roc_curve = roc(test_data$salary,PredRF,plot = TRUE, print.auc = TRUE)
```

```
## Setting levels: control =  <=50K, case =  >50K
```

```
## Setting direction: controls < cases
```




```
AUC<- auc(roc_curve)
```

Calculate: accuracy, misclassification rate, true positive rate, false positive rate, specificity, precision, and prevalence.

```
#Accuracy score?
print("Accuracy:")
```

```
## [1] "Accuracy:"
```

```
accuracy = sum(diag(classMatrix))/sum(classMatrix)
print(accuracy)
```

```
## [1] 0.9446833
```

```
cat("\n")
```

```
print("Misclassification Rate")
```

```
## [1] "Misclassification Rate"
```

```
print(1-accuracy)
```

```
## [1] 0.05531667
```

```
cat("\n")
```

```
true_negative<-classMatrix[1,1]
false_positive<-classMatrix[2,1]
true_positive<-classMatrix[2,2]
false_negative<-classMatrix[1,2]
```

```
print("True Positive Rate")
```

```
## [1] "True Positive Rate"
```

```
true_positive_rate = (true_positive)/ (true_positive + false_negative)
print(true_positive_rate)
```

```
## [1] 0.8541595
```

```
cat("\n")
```

```
print("False Positive Rate")
```

```
## [1] "False Positive Rate"
```

```
false_positive_rate = (false_positive)/ (false_positive + true_negative)
print(false_positive_rate)
```

```
## [1] 0.02657063
```

```
cat("\n")
```

```
print("Specificity:")
```

```
## [1] "Specificity:"
```

```
specificity<-true_negative/(true_negative+false_negative)
print(specificity)
```

```
## [1] 0.9545846
```

```
cat("\n")
```

```
print("Precision:")
```

```
## [1] "Precision:"
```

```
precision<-true_positive/(true_positive+false_positive)
print(precision)
```

```
## [1] 0.9107801
```

```
cat("\n")
```

```
print("Prevalence:")
```

```
## [1] "Prevalence:"
```

```
prevalence = (true_positive + false_negative)/(true_negative+false_positive+true_positive+false_negativ
print(prevalence)
```

```
## [1] 0.2410167
```

```
#add to model_stats
model_stats$randomForest = c(accuracy, true_positive_rate, false_positive_rate, specificity, precision,
model_stats$randomForest = round(model_stats$randomForest,2)
```

Compare these metrics between all three models.

```
write.csv(model_stats, file='data/model_stats.csv')
model_stats
```

```
##                metric naiveBayes Logistic CART randomForest
## 1            Accuracy       0.83     0.85 0.85          0.94
## 2  True Positive Rate       0.53     0.62 0.52          0.85
## 3 False Positive Rate       0.07     0.07 0.04          0.03
## 4         Specificity       0.86     0.88 0.86          0.95
## 5           Precision       0.70     0.73 0.79          0.91
## 6          Prevalence       1.00     1.00 1.00          1.00
```