Project

Mangukiya Jaldip Veljibhai

2023-04-18

```
# install and attach readr package
if(!require(readr)){install.packages("readr")}
library("readr")
```

Interpretation
Install and attach the ***readr*** package to work with the text dataset.

```
# Read a txt file, named "PROG8430-23W-Final-train.txt and change it to the dataframe"
data <- read.table("PROG8430-23W-Final-train.txt", sep = ",", header = TRUE)
data <- as.data.frame(data)
```

Interpretation
Read the "PROG8430_Assign04_23W.txt" file and store it in the data variable. Transform this variable into a dataframe for further processing.

```
# convert character variable to factor

data <- as.data.frame(unclass(data), stringsAsFactors = TRUE)
head(data, 5)
```

```
##   X    Inc Capital Hours Age Industry   Education       Marital   Occupation
## 1 1  >$50K       0    42  55  Private        Bach       Married       Trades
## 2 2 <=$50K       0     5  82  Private      Master Never Married        Admin
## 3 3 <=$50K       0    33  26  Private         Min Never Married Agricultural
## 4 4 <=$50K       0    12  79  Private High School       Married        Admin
## 5 5 <=$50K       0    56  19  Private High School       Married       Trades
```

Interpretation
Transform character variables to factor variable for further processing.

```
# install and attach lattice, pastecs package
if(!require(lattice)){install.packages("lattice")}
```

```
## Loading required package: lattice
```

```
library("lattice")

if(!require(pastecs)){install.packages("pastecs")}
```

```
## Loading required package: pastecs
```

```r
library("pastecs")
```

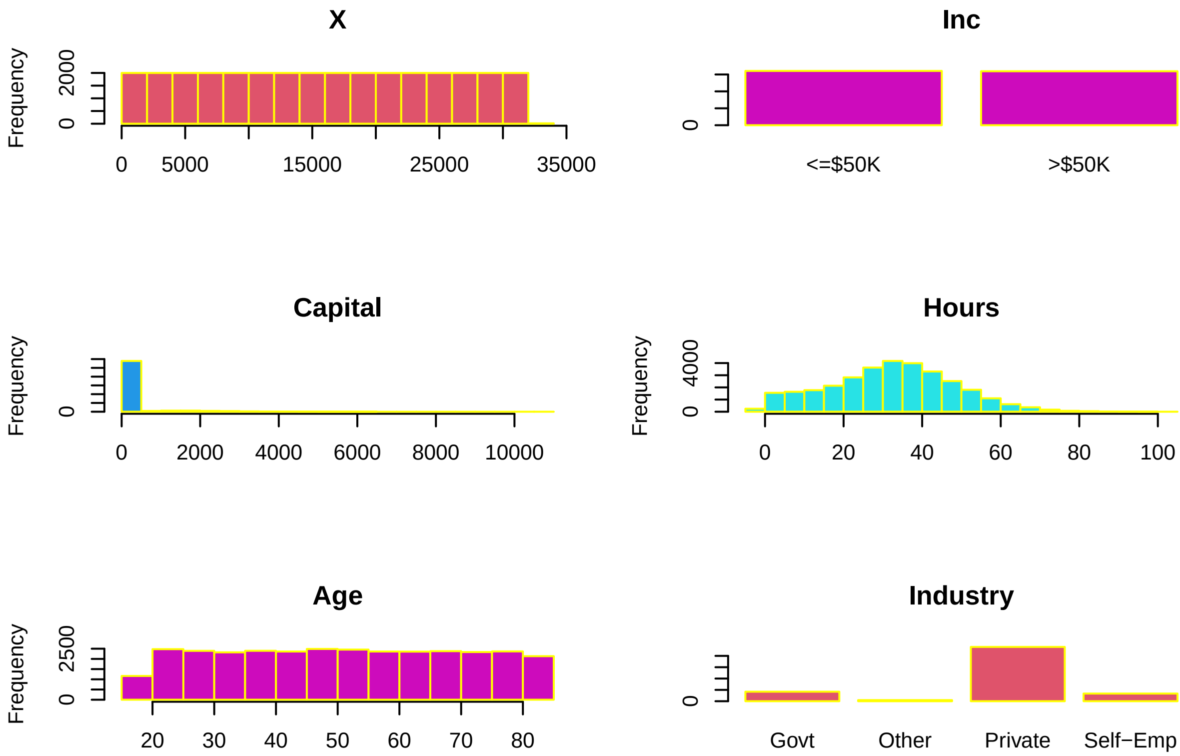
Interpretation
Install and attach required package for dataset exploration.

```r
# Explore statistics of each variable
round(stat.desc(data), 3)
```

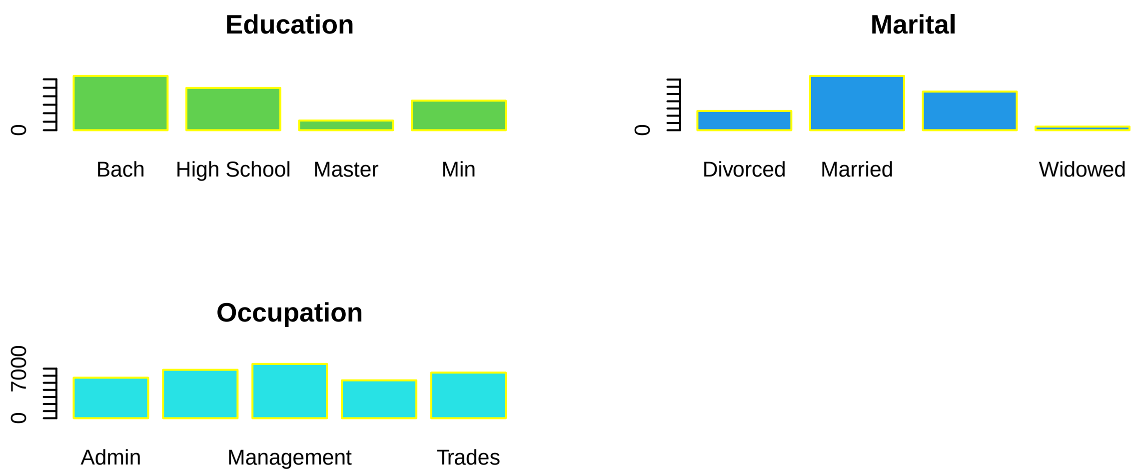
```
##                         X Inc     Capital       Hours         Age Industry
## nbr.val         32015.000  NA   32015.000   32015.000   32015.000       NA
## nbr.null            0.000  NA   28819.000     245.000       0.000       NA
## nbr.na              0.000  NA       0.000       0.000       0.000       NA
## min                 1.000  NA       0.000      -1.000      18.000       NA
## max             32015.000  NA   10532.000     103.000      85.000       NA
## range           32014.000  NA   10532.000     104.000      67.000       NA
## sum         512496120.000  NA 6628851.000 1046752.000 1645077.000       NA
## median          16008.000  NA       0.000      33.000      51.000       NA
## mean            16008.000  NA     207.055      32.696      51.385       NA
## SE.mean            51.653  NA       4.081       0.090       0.108       NA
## CI.mean.0.95      101.241  NA       7.999       0.176       0.212       NA
## var          85416020.000  NA  533249.476     259.521     373.116       NA
## std.dev          9242.079  NA     730.239      16.110      19.316       NA
## coef.var            0.577  NA       3.527       0.493       0.376       NA
##              Education Marital Occupation
## nbr.val             NA      NA         NA
## nbr.null            NA      NA         NA
## nbr.na              NA      NA         NA
## min                 NA      NA         NA
## max                 NA      NA         NA
## range               NA      NA         NA
## sum                 NA      NA         NA
## median              NA      NA         NA
## mean                NA      NA         NA
## SE.mean             NA      NA         NA
## CI.mean.0.95        NA      NA         NA
## var                 NA      NA         NA
## std.dev             NA      NA         NA
## coef.var            NA      NA         NA
```

```r
# make graphs of variables for exploring the data
par(mfrow=c(3,2))

for (i in 1:ncol(data)) {
  if (is.numeric(data[,i])) {
    hist(data[,i], main=names(data)[i], xlab="", col=i+1, border='yellow')
  } else if (is.factor(data[,i])) {
      cat_tbl <- table(data[i])
      barplot(cat_tbl, main=names(data)[i], col=i+4, border='yellow')
  }
}
```




```
par(mfrow=c(1,1))
```

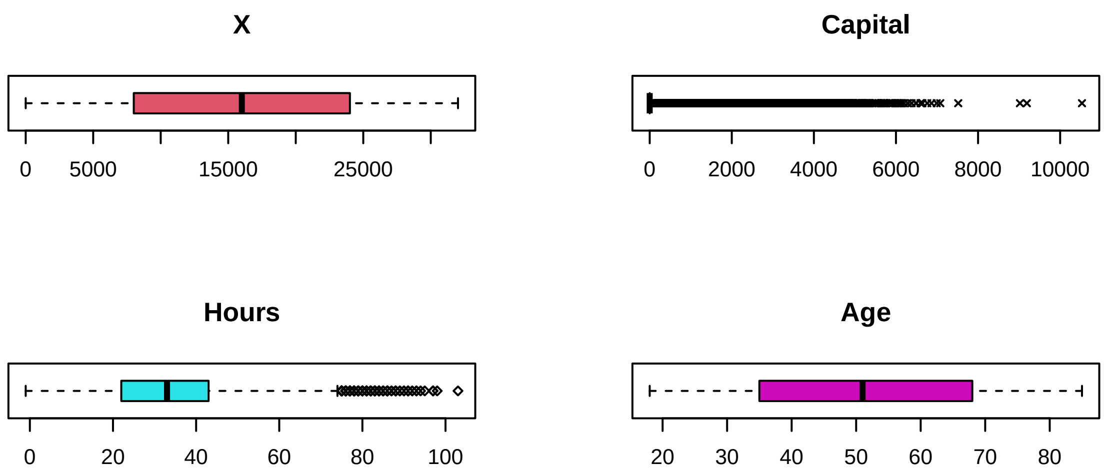


Interpretation
To acquire a broad perspective of the data, create a histogram of the numerical columns and a bar plot of the categorical variables.

```r
# make graphs for checking the outliers in the dataset
par(mfrow=c(3,2))

for (i in 1:ncol(data)) {
  if (is.numeric(data[,i])) {
    boxplot(data[,i], main=names(data)[i],xlab="", horizontal=TRUE,
            pch=i+1, col=i+1)
  }
}

par(mfrow=c(1,1))
```



Interpretation
After analysing the box plots, I noticed some of the variables have outliers such as Hours.

```
data <- data[, -which(names(data) == "X")]
head(data, 5)
```

```
##      Inc Capital Hours Age Industry   Education        Marital   Occupation
## 1  >$50K       0    42  55  Private        Bach        Married       Trades
## 2 <=$50K       0     5  82  Private      Master Never Married        Admin
## 3 <=$50K       0    33  26  Private         Min Never Married Agricultural
## 4 <=$50K       0    12  79  Private High School        Married        Admin
## 5 <=$50K       0    56  19  Private High School        Married       Trades
```

Interpretation:
I removed the “X” column, which does not provide any useful analytical information.

```
# remove outliers from the Hours (Hours worked in a typical week)

data <- data[!data$Hours < 0,]
dim(data)
```

```
## [1] 32014     8
```

Interpretation: I removed the record which has working hours less than 0, which is not feasible.

```r
# make a target variable(RES), remove Inc
data$RES <- as.factor(ifelse(data$Inc == ">$50K",1,0))
data <- data[, -which(names(data) == "Inc")]
head(data, 5)
```

```
##   Capital Hours Age Industry   Education       Marital   Occupation RES
## 1       0    42  55  Private        Bach       Married       Trades   1
## 2       0     5  82  Private      Master Never Married        Admin   0
## 3       0    33  26  Private         Min Never Married Agricultural   0
## 4       0    12  79  Private High School       Married        Admin   0
## 5       0    56  19  Private High School       Married       Trades   0
```

Interpretation
Make RES a new variable with the value 1 if Inc is greater than $50K and 0 otherwise. After that, remove the column Inc from the dataset; otherwise, Inc and RES show a significant correlation.

```r
#Load packages for correlations

if(!require(polycor)){install.packages("polycor")}
library("polycor")
```

Interpretation
Install and attach the polycor package to make correlation table between the dataset variables.

```r
# correlations between the variables
cor <- hetcor(data)
round(cor$correlations,2)
```

```
##            Capital Hours   Age Industry Education Marital Occupation   RES
## Capital       1.00  0.00  0.01    -0.01      0.00    0.00       0.00  0.10
## Hours         0.00  1.00 -0.60    -0.14     -0.04    0.03       0.01  0.79
## Age           0.01 -0.60  1.00     0.04      0.01   -0.01      -0.01 -0.17
## Industry     -0.01 -0.14  0.04     1.00      0.02    0.01      -0.02 -0.28
## Education     0.00 -0.04  0.01     0.02      1.00    0.00       0.01 -0.08
## Marital       0.00  0.03 -0.01     0.01      0.00    1.00       0.00  0.04
## Occupation    0.00  0.01 -0.01    -0.02      0.01    0.00       1.00  0.02
## RES           0.10  0.79 -0.17    -0.28     -0.08    0.04       0.02  1.00
```

Interpretation
Target variable RES is strong positive correlated with Hours, and week correlation with Capital and Occupation.

```r
# to generate same testing and training data every time
set.seed(9187)

# 80% of dataset as training set and remaining 20% as testing set
sample_data <- sample(c(TRUE, FALSE), nrow(data), replace=TRUE, prob=c(0.80,0.20))
train <- data[sample_data, ]
test <- data[!sample_data, ]
```

Interpretation
I utilised the seed function along with the last four digits of my student ID as the seed to consistently

generate similar training and testing datasets. Using the sample function, divide the dataset in half: 80% for training and 20% for testing. Filter the data using this variable, then put it in train and test.

```
# full logistic regression model
full_reg <- glm(RES ~ ., data=train, family="binomial", na.action=na.omit)
summary(full_reg)
```

```
##
## Call:
## glm(formula = RES ~ ., family = "binomial", data = train, na.action = na.omit)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.8587  -0.3422  -0.0165   0.3531   3.1471
##
## Coefficients:
##                           Estimate   Std. Error z value        Pr(>|z|)
## (Intercept)            -14.91855227   0.23013041 -64.827         < 2e-16 ***
## Capital                  0.00054155   0.00003047  17.775         < 2e-16 ***
## Hours                    0.28101430   0.00375666  74.804         < 2e-16 ***
## Age                      0.11331772   0.00203054  55.807         < 2e-16 ***
## IndustryOther           -0.75113220   0.19029928  -3.947 0.0000791002380 ***
## IndustryPrivate          0.22699572   0.06009803   3.777        0.000159 ***
## IndustrySelf-Emp        -3.23358142   0.14172645 -22.816         < 2e-16 ***
## EducationHigh School     0.22152519   0.05001848   4.429 0.0000094729306 ***
## EducationMaster         -0.14545045   0.08619835  -1.687        0.091528 .
## EducationMin            -0.38491403   0.05707619  -6.744 0.0000000000154 ***
## MaritalMarried           0.18498620   0.05973765   3.097        0.001957 **
## MaritalNever Married     0.17898871   0.06312303   2.836        0.004575 **
## MaritalWidowed           0.46790032   0.12926464   3.620        0.000295 ***
## OccupationAgricultural  -0.25613694   0.06755667  -3.791        0.000150 ***
## OccupationManagement     0.19195126   0.06589337   2.913        0.003579 **
## OccupationSpecialist     0.06372978   0.07235893   0.881        0.378456
## OccupationTrades         0.05247864   0.06862004   0.765        0.444408
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 35435  on 25561  degrees of freedom
## Residual deviance: 14410  on 25545  degrees of freedom
## AIC: 14444
##
## Number of Fisher Scoring iterations: 7
```

Interpretation
Utilizing the "generalized linear model(glm)" function, create a full logistic regression model with the train dataset. All other variables as independent and RES would be the target variable.

Evaluate Model

1. Fisher Scoring Iteration is 7, which is converged.
2. Residuals deviance is 14410

3. Residuals are symmetrical and median is near 0.
4. AIC is 14444

```r
# logistic regression model using step-wise selection
start_stp <- Sys.time()

step_reg <- step(full_reg)
```

```
## Start:  AIC=14443.7
## RES ~ Capital + Hours + Age + Industry + Education + Marital +
##     Occupation
##
##              Df Deviance   AIC
## <none>             14410 14444
## - Marital     3    14427 14455
## - Occupation  4    14464 14490
## - Education   3    14517 14545
## - Capital     1    14755 14787
## - Industry    3    15667 15695
## - Age         1    19530 19562
## - Hours       1    31369 31401
```

```r
summary(step_reg)
```

```
##
## Call:
## glm(formula = RES ~ Capital + Hours + Age + Industry + Education +
##     Marital + Occupation, family = "binomial", data = train,
##     na.action = na.omit)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.8587  -0.3422  -0.0165   0.3531   3.1471
##
## Coefficients:
##                          Estimate   Std. Error z value        Pr(>|z|)
## (Intercept)           -14.91855227   0.23013041 -64.827         < 2e-16 ***
## Capital                 0.00054155   0.00003047  17.775         < 2e-16 ***
## Hours                   0.28101430   0.00375666  74.804         < 2e-16 ***
## Age                     0.11331772   0.00203054  55.807         < 2e-16 ***
## IndustryOther          -0.75113220   0.19029928  -3.947 0.0000791002380 ***
## IndustryPrivate         0.22699572   0.06009803   3.777        0.000159 ***
## IndustrySelf-Emp       -3.23358142   0.14172645 -22.816         < 2e-16 ***
## EducationHigh School    0.22152519   0.05001848   4.429 0.0000094729306 ***
## EducationMaster        -0.14545045   0.08619835  -1.687        0.091528 .
## EducationMin           -0.38491403   0.05707619  -6.744 0.0000000000154 ***
## MaritalMarried          0.18498620   0.05973765   3.097        0.001957 **
## MaritalNever Married    0.17898871   0.06312303   2.836        0.004575 **
## MaritalWidowed          0.46790032   0.12926464   3.620        0.000295 ***
## OccupationAgricultural -0.25613694   0.06755667  -3.791        0.000150 ***
## OccupationManagement    0.19195126   0.06589337   2.913        0.003579 **
## OccupationSpecialist    0.06372978   0.07235893   0.881        0.378456
```

```
## OccupationTrades         0.05247864   0.06862004   0.765        0.444408
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 35435  on 25561  degrees of freedom
## Residual deviance: 14410  on 25545  degrees of freedom
## AIC: 14444
##
## Number of Fisher Scoring iterations: 7
```

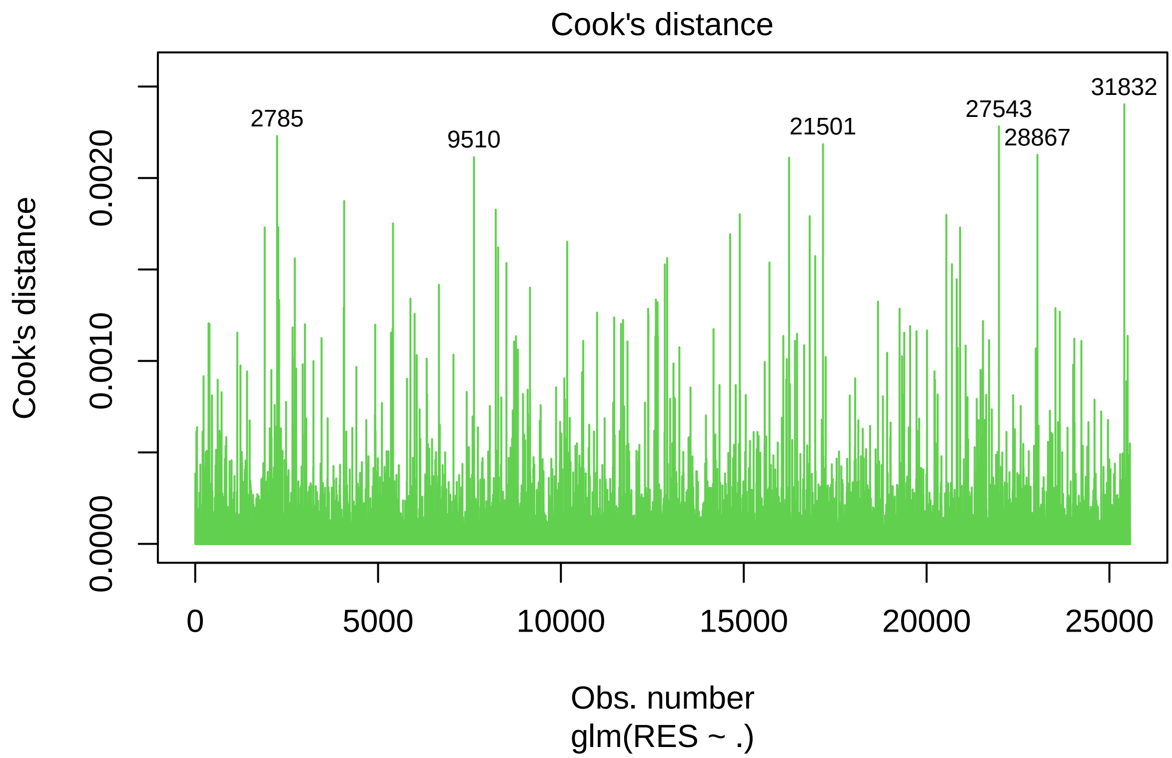
```
end_stp <- Sys.time()
```

Interpretation
create a stepwise logistic regression model with the train dataset. All other variables as independent and RES would be the target variable.

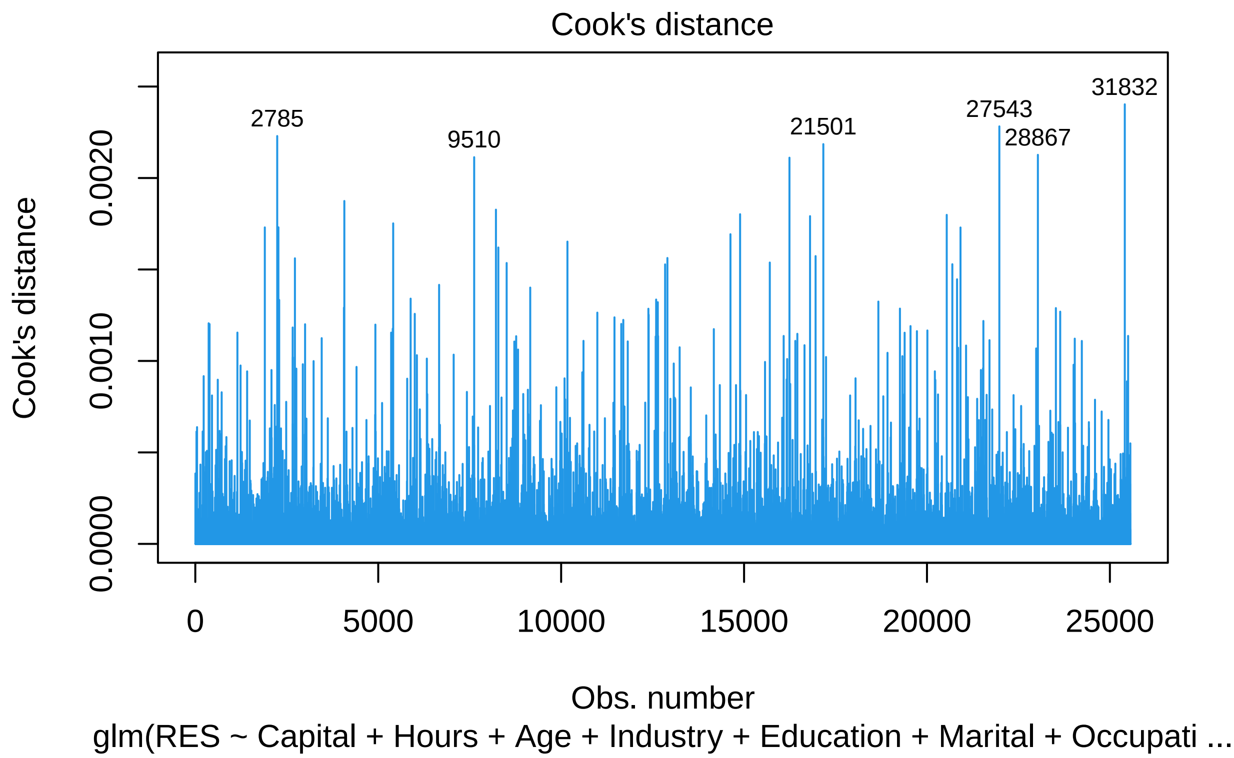
Evaluate Model

1. Fisher Scoring Iteration is 7, which is converged.
2. Residuals deviance is 14410
3. Residuals are symmetrical and median is near 0.
4. AIC is 14444

```
# plot of logistic regression model
plot(full_reg, which=4, id.n=6, col=3)
```




```r
plot(step_reg, which=4, id.n=6, col=4)
```



Interpretation
Make a scatter plot of full and step wise regression models. There are no data points over Cook's distance, neither the full model nor the backward selection model contain any data points that are particularly influential.

Test Accuracy, Precision and other parameters of step-wise logistic regression

```
# Confusion matrix of step-wise logistic regression and it's parameters
resp_stp_train <- predict(step_reg, newdata=train, type="response")
class_stp_train <- ifelse(resp_stp_train > 0.5,"1","0")
CF_stp_train <- table(train$RES, class_stp_train,
                dnn=list("Actual","Predicted"))

resp_stp_test <- predict(step_reg, newdata=test, type="response")
class_stp_test <- ifelse(resp_stp_test > 0.5,"1","0")
CF_stp_test <- table(test$RES, class_stp_test,
                dnn=list("Actual","Predicted"))

stp_reg_trn_acc <-
  (CF_stp_train[1,1] + CF_stp_train[2,2])/sum(CF_stp_train)
stp_reg_trn_prv <-
  (CF_stp_train[2,1] + CF_stp_train[2,2])/sum(CF_stp_train)
stp_reg_trn_miss_rate <-
  (CF_stp_train[2,1] + CF_stp_train[1,2])/sum(CF_stp_train)

stp_reg_tst_acc <-
  (CF_stp_test[1,1] + CF_stp_test[2,2])/sum(CF_stp_test)
```

```r
stp_reg_tst_prv <-
  (CF_stp_test[2,1] + CF_stp_test[2,2])/sum(CF_stp_test)
stp_reg_tst_miss_rate <-
  (CF_stp_test[2,1] + CF_stp_test[1,2])/sum(CF_stp_test)

print("Model: Step wise logistic regression model")
```

```
## [1] "Model: Step wise logistic regression model"
```

```r
print("Confusion matrix of test data")
```

```
## [1] "Confusion matrix of test data"
```

```r
CF_stp_train
```

```
##       Predicted
## Actual     0     1
##      0 11259  1616
##      1  1631 11056
```

```r
cat("Accuracy of the train dataset", round(stp_reg_trn_acc, 2), "\n")
```

```
## Accuracy of the train dataset 0.87
```

```r
cat("Prevalence of the train dataset", round(stp_reg_trn_prv, 2), "\n")
```

```
## Prevalence of the train dataset 0.5
```

```r
cat("Missclassification rate of the train dataset",
    round(stp_reg_trn_miss_rate, 2), "\n")
```

```
## Missclassification rate of the train dataset 0.13
```

```r
print("Confusion matrix of test data")
```

```
## [1] "Confusion matrix of test data"
```

```r
CF_stp_test
```

```
##       Predicted
## Actual    0    1
##      0 2815  365
##      1  385 2887
```

```r
cat("Accuracy of the test dataset", round(stp_reg_tst_acc, 2), "\n")
```

```
## Accuracy of the test dataset 0.88
```

```r
cat("Prevalence of the train dataset", round(stp_reg_tst_prv, 2), "\n")
```

```
## Prevalence of the train dataset 0.51
```

```r
cat("Missclassification rate of the train dataset",
    round(stp_reg_tst_miss_rate, 2))
```

```
## Missclassification rate of the train dataset 0.12
```

Table 1: Analysis of step-wise logistic regression confusion matrix

Parameters	Train Dataset	Test Dataset
Accuracy	87%	88%
Prevalence	50%	51%
Missclassification rate	13%	12%

```r
# calculate processing time of step wise logistic regression model
stp_reg_time <- end_stp - start_stp
stp_reg_time
```

```
## Time difference of 1.31615 secs
```

Interpretation

Time to run a step wise logistic regression is 1.25 seconds.

Surprisingly, the full and step-wise logistic regression models have the same AIC and Residual deviance. So, I will try another model for better model and accuracy.

```r
# install and attach packages for Naive Bayes, Recursive Partitioning,
# and Neural Network Classifications

if(!require(tinytex)){install.packages("tinytex")}
library("tinytex")

if(!require(pastecs)){install.packages("pastecs")}
library("pastecs")

if(!require(lattice)){install.packages("lattice")}
library("lattice")

if(!require(vcd)){install.packages("vcd")}
library("vcd")

if(!require(HSAUR)){install.packages("HSAUR")}
library("HSAUR")

if(!require(rmarkdown)){install.packages("rmarkdown")}
library("rmarkdown")
```

```r
if(!require(ggplot2)){install.packages("ggplot2")}
library("ggplot2")

if(!require(klaR)){install.packages("klaR")}
library("klaR")

if(!require(MASS)){install.packages("MASS")}
library("MASS")

if(!require(partykit)){install.packages("partykit")}
library("partykit")

if(!require(nnet)){install.packages("nnet")}
library("nnet")
```

Let's try Naive Bayesian Algorithm

```r
# Naive Bayes classification
start_naive <- Sys.time()

naive <- NaiveBayes(RES ~ . , data = train, na.action=na.omit)

end_naive <- Sys.time()
```

Interpretation
Utilizing the "NaiveBayes" function, create a Naive Bayesian classification model with the train dataset. All other variables as independent and RES would be the target variable. Calculate the processing time of the Naive Bayesian classification using "Sys.time" function.

2(2)

```r
# Confusion matrix of Naive Bayesian and it's parameters
pred_naive_train <- predict(naive, newdata=train)

CF_naive_trn <- table(Actual=train$RES, Predicted=pred_naive_train$class)

pred_naive_test <- predict(naive, newdata=test)

CF_naive_tst <- table(Actual=test$RES, Predicted=pred_naive_test$class)

nb_trn_acc <-
  (CF_naive_trn[1,1] + CF_naive_trn[2,2])/sum(CF_naive_trn)
nb_tst_acc <-
  (CF_naive_tst[1,1] + CF_naive_tst[2,2])/sum(CF_naive_tst)

nb_trn_prv <-
  (CF_naive_trn[2,1] + CF_naive_trn[2,2])/sum(CF_naive_trn)
nb_tst_prv <-
  (CF_naive_tst[2,1] + CF_naive_tst[2,2])/sum(CF_naive_tst)
```

```
nb_trn_miss_rate <-
  (CF_naive_trn[2,1] + CF_naive_trn[1,2])/sum(CF_naive_trn)
nb_tst_miss_rate <-
  (CF_naive_tst[2,1] + CF_naive_tst[1,2])/sum(CF_naive_tst)


print("Model: Naive Bayesian classification model")
```

```
## [1] "Model: Naive Bayesian classification model"
```

```
print("Confusion matrix of train data")
```

```
## [1] "Confusion matrix of train data"
```

```
CF_naive_trn
```

```
##       Predicted
## Actual     0     1
##      0 11070  1805
##      1  1518 11169
```

```
cat("Accuracy of the train dataset", round(nb_trn_acc, 2), "\n")
```

```
## Accuracy of the train dataset 0.87
```

```
cat("Prevalence of the train dataset", round(nb_trn_prv, 2), "\n")
```

```
## Prevalence of the train dataset 0.5
```

```
cat("Missclassification rate of the train dataset",
    round(nb_trn_miss_rate, 2), "\n")
```

```
## Missclassification rate of the train dataset 0.13
```

```
print("Confusion matrix of test data")
```

```
## [1] "Confusion matrix of test data"
```

```
CF_naive_tst
```

```
##       Predicted
## Actual    0    1
##      0 2734  446
##      1  368 2904
```

```
cat("Accuracy of the train dataset", round(nb_tst_acc, 2), "\n")
```

```
## Accuracy of the train dataset 0.87
```

```
cat("Prevalence of the train dataset", round(nb_tst_prv, 2), "\n")
```

```
## Prevalence of the train dataset 0.51
```

```
cat("Missclassification rate of the train dataset",
    round(nb_tst_miss_rate, 2), "\n")
```

```
## Missclassification rate of the train dataset 0.13
```

Interpretation
Create a prediction variable using the train dataset and the predict function, then create a corresponding variable for the test dataset. For making confusion matrix, set actual and predicted parameter in the table function for both the dataset. Using confusion matrix, calculate accuracy, prevalence, and missclassification rate.

Table 2: Analysis of Confusion Matrix of Naive Bayesian

Parameters	*Train dataset*	*Test dataset*
Accuracy	87%	87%
Prevalence	50%	51%
Missclassification rate	13%	13%

Therefore, our Naive Bayesian classification model is good and not overfitting or underfitting.

2(3)

```
# calculate processing time of Naive Bayesian classification model
naive_time <- end_naive - start_naive
naive_time
```

```
## Time difference of 0.02752614 secs
```

Interpretation
Calculate the computation time for running the Naive Bayesian classification model, which is 0.028 seconds.

Let's try Neural Network Algorithm, we may get better accuracy

```
# Neural Network model
start_nn <- Sys.time()

set.seed(9187)
nn <- nnet(RES ~ .,
          data=train,
          size=4,
```

```r
          rang=0.1,
          maxit=1500,
          trace=FALSE)

end_nn <- Sys.time()
```

Interpretation
RES would be the target variable, while all other variables would be considered independent variables. Set numerous parameters, such as size, the number of nodes in a single hidden layer of the model, maxit, the maximum number of optimisation iterations, and rang, the range of random weights provided to the connections between the input and hidden layers. Determine the Neural Network classification model's processing duration.

```r
# Confusion matrix of Neural Network and it's parameters
pred_nn_train <- predict(nn, newdata=train, type="class")
CF_nn_trn <- table(Actual=train$RES, Predicted=pred_nn_train)

pred_nn_test <- predict(nn, newdata=test, type="class")
CF_nn_tst <- table(Actual=test$RES, Predicted=pred_nn_test)

nn_trn_acc <-
  (CF_nn_trn[1,1] + CF_nn_trn[2,2])/sum(CF_nn_trn)
nn_trn_prv <-
  (CF_nn_trn[2,1] + CF_nn_trn[2,2])/sum(CF_nn_trn)
nn_trn_miss_rate <-
  (CF_nn_trn[2,1] + CF_nn_trn[1,2])/sum(CF_nn_trn)

nn_tst_acc <-
  (CF_nn_tst[1,1] + CF_nn_tst[2,2])/sum(CF_nn_tst)
nn_tst_prv <-
  (CF_nn_tst[2,1] + CF_nn_tst[2,2])/sum(CF_nn_tst)
nn_tst_miss_rate <-
  (CF_nn_tst[2,1] + CF_nn_tst[1,2])/sum(CF_nn_tst)


print("Model: Neural Network model")
```

```
## [1] "Model: Neural Network model"
```

```r
print("Confusion matrix of test data")
```

```
## [1] "Confusion matrix of test data"
```

```r
CF_nn_trn
```

```
##       Predicted
## Actual     0     1
##      0 12434   441
##      1    15 12672
```

```
cat("Accuracy of the train dataset", round(nn_trn_acc, 2), "\n")
```

```
## Accuracy of the train dataset 0.98
```

```
cat("Prevalence of the train dataset", round(nn_trn_prv, 2), "\n")
```

```
## Prevalence of the train dataset 0.5
```

```
cat("Missclassification rate of the train dataset",
    round(nn_trn_miss_rate, 2), "\n")
```

```
## Missclassification rate of the train dataset 0.02
```

```
print("Confusion matrix of test data")
```

```
## [1] "Confusion matrix of test data"
```

```
CF_nn_tst
```

```
##       Predicted
## Actual    0    1
##      0 3070  110
##      1    8 3264
```

```
cat("Accuracy of the train dataset", round(nn_tst_acc, 2), "\n")
```

```
## Accuracy of the train dataset 0.98
```

```
cat("Prevalence of the train dataset", round(nn_tst_prv, 2), "\n")
```

```
## Prevalence of the train dataset 0.51
```

```
cat("Missclassification rate of the train dataset",
    round(nn_tst_miss_rate, 2), "\n")
```

```
## Missclassification rate of the train dataset 0.02
```

Interpretation
Create a prediction variable using the train dataset and the predict function for neural network, then create a corresponding variable for the test dataset. For making a a confusion matrix, set actual and predicted parameter in the table function for both the dataset. Using confusion matrix, calculate accuracy, prevalence, and missclassification rate.

Table 3: Analysis of Neural Network Confusion Matrix

Parameters	Train Dataset	Test Dataset
Accuracy	98%	98%
Prevalence	50%	51%
Missclassification rate	2%	2%

```
# calculate processing time of Neural Network classification model
nn_time <- end_nn - start_nn
nn_time
```

```
## Time difference of 32.55671 secs
```

Interpretation Time to run a neural network model is 32.40 seconds.

Read out prediction

```
# Import test dataset to test the efficiency and accuracy of the model
test_data <- read.table("PROG8430-23W-Final-test.txt", header = TRUE, sep = ",")
test_data <- as.data.frame(test_data)
test_data <- test_data[, -which(names(test_data) == "X")]
head(test_data, 5)
```

```
##   Capital Hours Age Industry   Education       Marital   Occupation
## 1       0    43  29  Private         Min       Married   Specialist
## 2       0     2  83     Govt High School       Married       Trades
## 3       0    27  40  Private         Min      Divorced   Management
## 4       0    31  50  Private        Bach Never Married Agricultural
## 5       0    21  77  Private        Bach       Married       Trades
```

Interpretation
read test file "PROG8430-23W-Final-test", convert it into dataframe. Add initials to the column name. Remove X column because it does not make any analytical information.

```
# Convert character variable to factor
test_data <- as.data.frame(unclass(test_data), stringsAsFactors = TRUE)
```

Interpretation
Convert character variables to factor variabls.

```
# Prediction on test dataset and export it into txt dataset
pred <- predict(nn, newdata=test_data, type="class")
pred <- as.factor(ifelse(pred == 1, '>$50K','<=$50K'))
test_fin <- cbind(test_data, pred)
write.csv(test_fin, "PROG8430-23W-Final.txt")
```

Interpretation
Make prediction variables using the neural network model and combine them with the test file. Write an output file with predictions.