CSCI E-106 : Spring 2025 Final Exam

Instructions

1-) As always, you are required to follow Harvard University academic integrity and honor code.

2-) Open book and open notes exam (textbooks (print or pdf), lecture slides, notes, practice exam, homework solutions, and TA slides, including all Rmd's*).

3-) You are allowed to use RStudio Desktop or RStudio Cloud (https://rstudio.cloud.) , Microsoft Word, Power Point and PDF reader, and canvas on your laptop.

4-) Proctorio is required to start this exam. If you are prompted for an access code, you must Install and Configure Proctorio on your machine.

5-) Review the Proctorio Support for Test Takers page, Online Exam Checklist and Proctorio Checklist to help you avoid common errors and who to contact if you run into any issues.

6-) Practice Setup Quiz is available under Quizzes to test your connection and to find out if you are ok with the Proctorio.

7-) Please read the list of recording and restrictions provided by Proctorio carefully before taking the exam.

8-) The final exam will be available from Monday, May 12th at 12 pm EST through Tuesday, May 13th at 12:00 pm EST. You must submit your midterm by Tuesday, May 13th at 12:00 pm EST.

9-) Once you start the exam, you must complete it in 3 hours or by Tuesday, May 13th, at 12:00 p.m. EST.

10-) In order to receive full credit, please provide full explanations and calculations for each questions.

11-) Make sure that you are familiar with the procedures for troubleshooting exam issues. Preview the document Download the document and follow the protocol if there are any issues!

12-) Make sure you submit both .Rmd and (knitted) pdf or html files.

13-) You need to have a camera on your laptop.

14-) Please reach out to DCE Online Support by using the information below
DCE Online Support
(617) 998-8571
(Mon-Thurs 10am-11pm, Fri-Sun 10am-8pm EST)
AcademicTechnology@dce.harvard.edu

15-) Our emails:
hakangogtas@yahoo.com
rafael_gomeztagle@g.harvard.edu
andreahatch10@gmail.com
sezer@yahoo.com
srg3924@gmail.com

Problem 1

Refer to the Hitters data set. Major League Baseball Data from the 1986 and 1987 seasons. The data set can be downloaded from the attached excel file or directly from ISL2 library in R by copying and pasting the following command into R console:library(ISLR2);data("Hitters"). (55 Points)

Description of the data is below and there are 322 observations and 20 variables, including 3 categorical variables (League, Division and NewLeague).

X1=AtBat=Number of times at bat in 1986

X2=Hits=Number of hits in 1986

X3=HmRun=Number of home runs in 1986

X4=Runs=Number of runs in 1986

RBI=Number of runs batted in in 1986

Walks=Number of walks in 1986

Years=Number of years in the major leagues

CAtBat=Number of times at bat during his career

CHits=Number of hits during his career

CHmRun=Number of home runs during his career

CRuns=Number of runs during his career

CRBI=Number of runs batted in during his career

CWalks=Number of walks during his career

League=A factor with levels A and N indicating player's league at the end of 1986

Division=A factor with levels E and W indicating player's division at the end of 1986

PutOuts=Number of put outs in 1986

Assists=Number of assists in 1986

Errors=Number of errors in 1986

Salary=1987 annual salary on opening day in thousands of dollars

NewLeague=A factor with levels A and N indicating player's league at the beginning of 1987

a-) Remove the missing data by using na.omit comment (e.g. Hitters<-na.omit(Hitters)). Create dummy variables for the categorical variables League, Division and NewLeague. Drop the categorical variables after creating the dummy variables from the data set. Perform initial data analyses to identify outliers, missing data, and variables that show high correlation with the salary and with each other. Document your findings (10 points)

CAtBat,CHits,CHmRun and CRBI are moderately correlated with Salary with the highest correlation coefficients between 0.52 to 0.57. Missing data was excluded. Boxplots for Salary indicate that there are outliers and the distribution of salary is right skewed. Some of the independent variables are highly correlated with each other.

```
library(ISLR2);data("Hitters")
```

```
##
## Attaching package: 'ISLR2'
```

```
## The following object is masked from 'package:MASS':
##
##     Boston
```

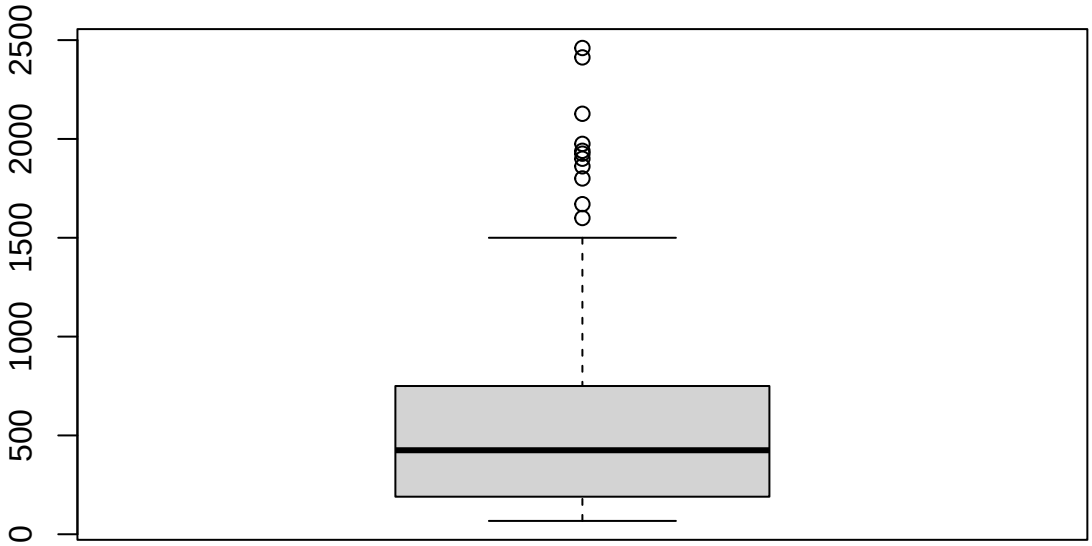
```
#adding the dummy variables
Hitters$League_1<- I(Hitters$League=="A")*1
Hitters$Division_1<- I(Hitters$Division=="E")*1
Hitters$NewLeague_1<- I(Hitters$NewLeague=="A")*1

#dropping the categorical variables
Hitters1<-na.omit(Hitters[,-c(14,15,20)])
round(cor(Hitters1),2)
```

```
##             AtBat Hits HmRun  Runs  RBI Walks Years CAtBat CHits CHmRun CRuns
## AtBat        1.00 0.96  0.56  0.90 0.80  0.62  0.01   0.21  0.23   0.21  0.24
## Hits         0.96 1.00  0.53  0.91 0.79  0.59  0.02   0.21  0.24   0.19  0.24
## HmRun        0.56 0.53  1.00  0.63 0.85  0.44  0.11   0.22  0.22   0.49  0.26
## Runs         0.90 0.91  0.63  1.00 0.78  0.70 -0.01   0.17  0.19   0.23  0.24
## RBI          0.80 0.79  0.85  0.78 1.00  0.57  0.13   0.28  0.29   0.44  0.31
## Walks        0.62 0.59  0.44  0.70 0.57  1.00  0.13   0.27  0.27   0.35  0.33
## Years        0.01 0.02  0.11 -0.01 0.13  0.13  1.00   0.92  0.90   0.72  0.88
## CAtBat       0.21 0.21  0.22  0.17 0.28  0.27  0.92   1.00  1.00   0.80  0.98
## CHits        0.23 0.24  0.22  0.19 0.29  0.27  0.90   1.00  1.00   0.79  0.98
## CHmRun       0.21 0.19  0.49  0.23 0.44  0.35  0.72   0.80  0.79   1.00  0.83
## CRuns        0.24 0.24  0.26  0.24 0.31  0.33  0.88   0.98  0.98   0.83  1.00
## CRBI         0.22 0.22  0.35  0.20 0.39  0.31  0.86   0.95  0.95   0.93  0.95
## CWalks       0.13 0.12  0.23  0.16 0.23  0.43  0.84   0.91  0.89   0.81  0.93
## PutOuts      0.31 0.30  0.25  0.27 0.31  0.28 -0.02   0.05  0.07   0.09  0.06
## Assists      0.34 0.30 -0.16  0.18 0.06  0.10 -0.09  -0.01 -0.01  -0.19 -0.04
## Errors       0.33 0.28 -0.01  0.19 0.15  0.08 -0.16  -0.07 -0.07  -0.17 -0.09
## Salary       0.39 0.44  0.34  0.42 0.45  0.44  0.40   0.53  0.55   0.52  0.56
## League_1     0.15 0.15  0.22  0.21 0.19  0.07  0.03   0.02  0.02   0.11  0.05
## Division_1   0.06 0.08  0.03  0.11 0.09  0.07  0.02   0.02  0.02   0.03  0.05
## NewLeague_1  0.09 0.09  0.20  0.15 0.14  0.03  0.02   0.00  0.00   0.10  0.03
##              CRBI CWalks PutOuts Assists Errors Salary League_1 Division_1
```

```
## AtBat         0.22   0.13    0.31    0.34   0.33   0.39     0.15        0.06
## Hits          0.22   0.12    0.30    0.30   0.28   0.44     0.15        0.08
## HmRun         0.35   0.23    0.25   -0.16  -0.01   0.34     0.22        0.03
## Runs          0.20   0.16    0.27    0.18   0.19   0.42     0.21        0.11
## RBI           0.39   0.23    0.31    0.06   0.15   0.45     0.19        0.09
## Walks         0.31   0.43    0.28    0.10   0.08   0.44     0.07        0.07
## Years         0.86   0.84   -0.02   -0.09  -0.16   0.40     0.03        0.02
## CAtBat        0.95   0.91    0.05   -0.01  -0.07   0.53     0.02        0.02
## CHits         0.95   0.89    0.07   -0.01  -0.07   0.55     0.02        0.02
## CHmRun        0.93   0.81    0.09   -0.19  -0.17   0.52     0.11        0.03
## CRuns         0.95   0.93    0.06   -0.04  -0.09   0.56     0.05        0.05
## CRBI          1.00   0.89    0.10   -0.10  -0.12   0.57     0.05        0.02
## CWalks        0.89   1.00    0.06   -0.07  -0.13   0.49     0.03        0.05
## PutOuts       0.10   0.06    1.00   -0.04   0.08   0.30    -0.04        0.03
## Assists      -0.10  -0.07   -0.04    1.00   0.70   0.03    -0.05        0.02
## Errors       -0.12  -0.13    0.08    0.70   1.00  -0.01    -0.09        0.00
## Salary        0.57   0.49    0.30    0.03  -0.01   1.00     0.01        0.19
## League_1      0.05   0.03   -0.04   -0.05  -0.09   0.01     1.00        0.00
## Division_1    0.02   0.05    0.03    0.02   0.00   0.19     0.00        1.00
## NewLeague_1   0.04   0.03   -0.06   -0.04  -0.06   0.00     0.86        0.00
##              NewLeague_1
## AtBat               0.09
## Hits                0.09
## HmRun               0.20
## Runs                0.15
## RBI                 0.14
## Walks               0.03
## Years               0.02
## CAtBat              0.00
## CHits               0.00
## CHmRun              0.10
## CRuns               0.03
## CRBI                0.04
## CWalks              0.03
## PutOuts            -0.06
## Assists            -0.04
## Errors             -0.06
## Salary              0.00
## League_1            0.86
## Division_1          0.00
## NewLeague_1         1.00
```

```
boxplot(Hitters1$Salary)
```



b-) Create train and test data sets: select a random sample of 70% observations from the data set for the train data set and remaining cases for the test data set. (use set.seed(994) before selecting the sample and running Neuron Network and Regression Tree) (5 points)

```
set.seed(994)

n1 <- nrow(Hitters1)
q1_sample <- sample(seq_len(n1), size = floor(0.7 * n1))

q1_train_data <- Hitters1[q1_sample, ]
q1_test_data  <- Hitters1[-q1_sample, ]
```

c-) Use stepwise (both ways) model selection to select the best model for predicting the Salary on the train data set. **Ensure that all variables are significant, use** $\alpha = 0.05$. Justify your choice of model. Check all regression model assumptions visually with appropriate graphs and conduct the Breusch-Pagan Test to determine whether or not the error variances are constant. Please discuss your findings. (10 points)

Variables **CRuns, Walks, Years, PutOuts, and Division_1** are significant at $\alpha = 0.05$ in the selected model. The training-set R^2 is 0.5366 (about 54%), and the Q–Q plot suggests some outliers/heavier-than-normal tails; the residual plots also suggest non-constant variance.

Breusch-Pagan Test:

Ho: Error variances are constant
Ha: Error variances are not constant

The Breusch–Pagan test p-value is $0.0000000766 < 0.05$, so we reject Ho and conclude there is evidence of heteroskedasticity (error variance is not constant).

VIFs range from about 1.02 to 5.55 (all < 10), so there is no evidence of severe multicollinearity.

On the test set, performance decreases: R^2 drops from 0.5366 to 0.2982 (about 30%), indicating weaker generalization to unseen data.

```
f1<-lm(Salary~.,data=q1_train_data)
summary(f1)
```

```
##
## Call:
## lm(formula = Salary ~ ., data = q1_train_data)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -863.60 -176.44  -30.57  140.39 1900.81
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 115.38733  106.93158   1.079  0.28214
## AtBat        -2.29829    0.75705  -3.036  0.00279 **
## Hits          9.18831    2.96786   3.096  0.00231 **
## HmRun         1.24458    7.36854   0.169  0.86608
## Runs         -3.94566    3.69054  -1.069  0.28658
## RBI          -1.26084    3.24901  -0.388  0.69847
## Walks         7.72437    2.34601   3.293  0.00122 **
## Years        -7.25782   16.20029  -0.448  0.65474
## CAtBat       -0.08741    0.17228  -0.507  0.61258
## CHits         0.20401    0.89207   0.229  0.81939
## CHmRun        3.04194    2.05851   1.478  0.14140
## CRuns         1.47942    0.88735   1.667  0.09738 .
## CRBI         -0.49598    0.95222  -0.521  0.60316
## CWalks       -0.72749    0.41752  -1.742  0.08331 .
## PutOuts       0.20187    0.09230   2.187  0.03015 *
## Assists       0.17798    0.25347   0.702  0.48357
## Errors        0.63993    5.14745   0.124  0.90121
## League_1    -39.43829   93.77059  -0.421  0.67461
## Division_1   96.86162   49.32998   1.964  0.05127 .
## NewLeague_1 -44.41571   93.77660  -0.474  0.63639
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 317.1 on 164 degrees of freedom
## Multiple R-squared:  0.6067, Adjusted R-squared:  0.5611
## F-statistic: 13.31 on 19 and 164 DF,  p-value: < 2.2e-16
```

```
k1<-ols_step_both_p(f1, p_remove =0.05,details=FALSE)
k1$model
```

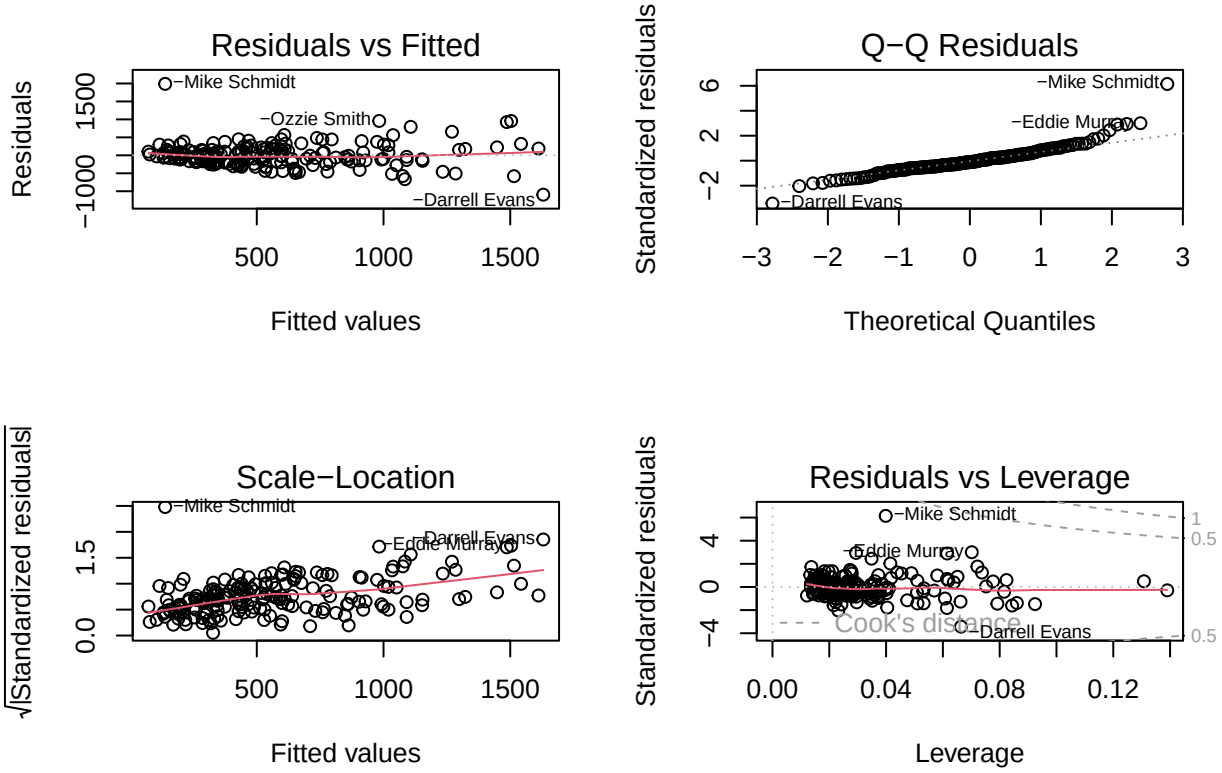
```
##
## Call:
## lm(formula = paste(response, "~", paste(preds, collapse = " + ")),
##     data = l)
##
## Coefficients:
## (Intercept)        CRuns        Walks        Years      PutOuts   Division_1
##     65.7659       1.2787       3.6514     -30.4516       0.2082     109.7112
```

```
f2<-lm(Salary~ CRuns+Walks+Years+PutOuts+Division_1,data=q1_train_data)
summary(f2)
```

```
##
## Call:
## lm(formula = Salary ~ CRuns + Walks + Years + PutOuts + Division_1,
##     data = q1_train_data)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -1095.34  -174.71   -37.98   149.64  1988.85
##
## Coefficients:
##              Estimate Std. Error t value       Pr(>|t|)
## (Intercept)  65.76589   76.39568   0.861        0.39047
## CRuns         1.27874    0.18738   6.824 0.000000000134 ***
## Walks         3.65142    1.36874   2.668        0.00834 **
## Years       -30.45162   11.67547  -2.608        0.00988 **
## PutOuts       0.20815    0.09011   2.310        0.02204 *
## Division_1  109.71118   49.23182   2.228        0.02710 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 330.4 on 178 degrees of freedom
## Multiple R-squared:  0.5366, Adjusted R-squared:  0.5236
## F-statistic: 41.22 on 5 and 178 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(f2)
```




```
ols_test_breusch_pagan(f2)
```

```
##
##  Breusch Pagan Test for Heteroskedasticity
##  -----------------------------------------
##  Ho: the variance is constant
##  Ha: the variance is not constant
##
##                Data
```

```
##  ---------------------------------
##  Response : Salary
##  Variables: fitted values of Salary
##
##            Test Summary
##  ---------------------------------
##  DF            =    1
##  Chi2          =    28.88988
##  Prob > Chi2   =    0.00000007661214
```

```
vif(f2)
```

```
##      CRuns      Walks      Years    PutOuts Division_1
##   5.545398   1.507364   4.881499   1.108803   1.021099
```

```
#performance on train and test data set

PredictedTest<-predict(f2,q1_train_data)
ModelTest1<-data.frame(obs = q1_train_data$Salary, pred=PredictedTest)
reg.train<-defaultSummary(ModelTest1)
reg.train
```

```
##        RMSE    Rsquared         MAE
## 324.9296606   0.5366007 228.5574162
```

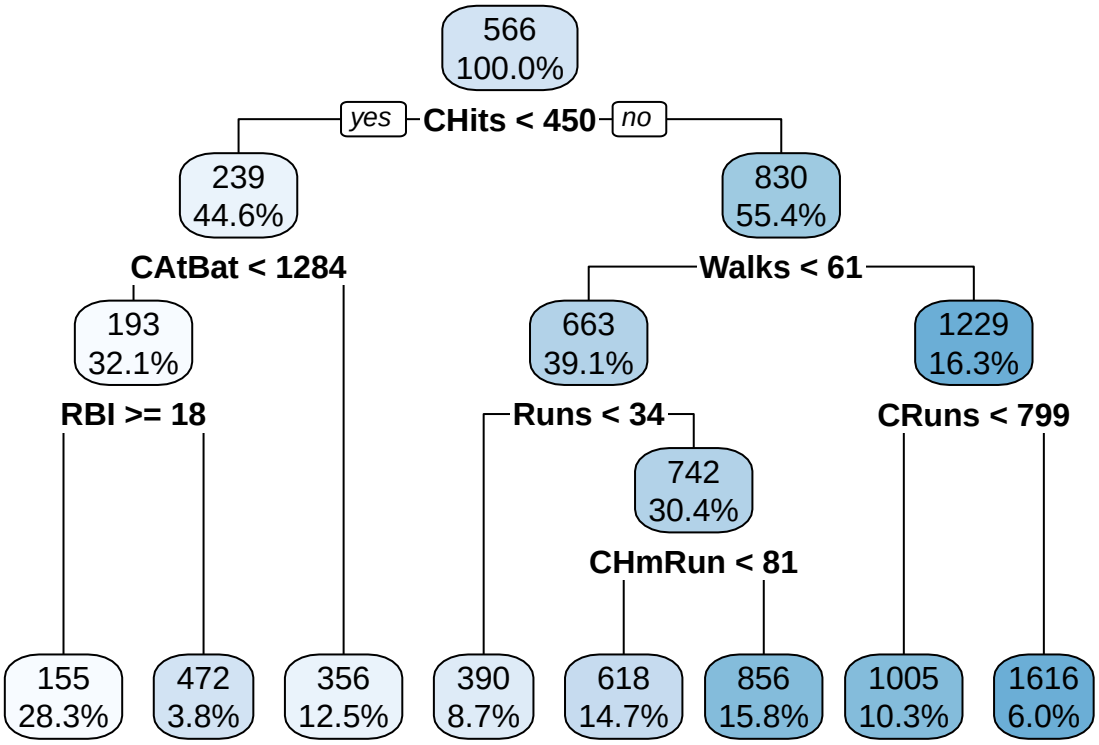
```
PredictedTest<-predict(f2,q1_test_data)
ModelTest2<-data.frame(obs = q1_test_data$Salary, pred=PredictedTest)
reg.test<-defaultSummary(ModelTest2)
```

d-) Use Regression tree to predict the Salary on the train data set and evaluate the performance on the train and test data sets. (10 points)

The performance of tree model is deteriorated on the test data set. R^2 is decreased from %68 to %36.

```
set.seed(994)
# Fitting the tree on train data set
par(mfrow=c(1,1))

q1.tree <- rpart(Salary ~ ., data = q1_train_data)
rpart.plot(q1.tree, digits = 3)
```




```
#regression tree on the test and train data
PredictedTest<-predict(q1.tree,q1_train_data)
ModelTest2<-data.frame(obs = q1_train_data$Salary, pred=PredictedTest)
tree.train<-defaultSummary(ModelTest2)
tree.train
```

```
##       RMSE   Rsquared        MAE
## 268.657395   0.683208 169.172399
```

```
PredictedTest<-predict(q1.tree,q1_test_data)
ModelTest2<-data.frame(obs = q1_test_data$Salary, pred=PredictedTest)
tree.test<-defaultSummary(ModelTest2)
tree.test
```

```
##        RMSE    Rsquared         MAE
## 346.1120076   0.3656705 239.8415325
```

e-) Use Neural Network (NN) with four hidden layers with 9,7,5, and 3 neurons respectively (hidden=c(9,7,5,3)) with softplus activation function (softplus <- function(x) { log(1 + exp(x)) }) to predict the Salary on the train data set and evaluate the performance of NN on the train and test data sets.(10 points)

The neural network fits the training data extremely well ($R^2 = 0.9838$), but test-set performance is lower ($R^2 = 0.4406$), indicating some overfitting (though it generalizes better than the regression and tree in terms of test R^2).

```
dat <- Hitters1
softplus <- function(x) { log(1 + exp(x)) }

# IMPORTANT: compute min/max from TRAIN only (avoid leakage)
train_mins <- sapply(q1_train_data, min)
train_maxs <- sapply(q1_train_data, max)

normalize_df <- function(df, mins, maxs) {
```

```
  out <- df
  for (nm in names(df)) {
    denom <- (maxs[[nm]] - mins[[nm]])
    out[[nm]] <- (df[[nm]] - mins[[nm]]) / denom
  }
  out
}

q1_norm_train <- normalize_df(q1_train_data, train_mins, train_maxs)
q1_norm_test  <- normalize_df(q1_test_data,  train_mins, train_maxs)

set.seed(994)
q1_nn <- neuralnet(Salary ~ ., data = q1_norm_train, hidden = c(9,7,5,3), act.fct = softplus)
plot(q1_nn)

# Correct inverse transform for Salary (must match train scaling)
salary_min <- train_mins[["Salary"]]
salary_max <- train_maxs[["Salary"]]
unnormalize <- function(x) { x * (salary_max - salary_min) + salary_min }

# performance on train
x_train <- q1_norm_train[, setdiff(names(q1_norm_train), "Salary")]
predicted_y <- compute(q1_nn, x_train)$net.result
pred_new <- unnormalize(predicted_y)

ModelTest1 <- data.frame(obs = q1_train_data$Salary, pred = as.vector(pred_new))
nn.train <- defaultSummary(ModelTest1)
nn.train
```

```
##       RMSE   Rsquared        MAE
## 60.8152904  0.9837674 40.9262350
```

```
# performance on test
x_test <- q1_norm_test[, setdiff(names(q1_norm_test), "Salary")]
predicted_y <- compute(q1_nn, x_test)$net.result
pred_new <- unnormalize(predicted_y)

ModelTest1 <- data.frame(obs = q1_test_data$Salary, pred = as.vector(pred_new))
nn.test <- defaultSummary(ModelTest1)
nn.test
```

```
##        RMSE    Rsquared         MAE
## 352.7021088   0.4405791 227.1888738
```

f-) Evaluate the performances of models built in part c through part e on test data set. Compare the R^2 and select the best model. (5 points)

Comparing test-set R^2 values: Regression ($R^2 = 0.298$), Tree ($R^2 = 0.366$), and Neural Network ($R^2 = 0.441$). The neural network has the highest test-set R^2 (and the lowest test MAE), so it is selected as the best model based on test performance.

```
out.train<-data.frame(rbind(reg.train,tree.train,nn.train))
dimnames(out.train)[[1]]<-c("Regression","Tree","Neuron Network")
out.train
```

```
##                     RMSE  Rsquared       MAE
## Regression     324.92966 0.5366007 228.55742
```

```
## Tree           268.65740 0.6832080 169.17240
## Neuron Network  60.81529 0.9837674  40.92623
```

```
out.test<-data.frame(rbind(reg.test,tree.test,nn.test))
dimnames(out.test)[[1]]<-c("Regression","Tree","Neuron Network")
out.test
```

```
##                    RMSE  Rsquared      MAE
## Regression     381.4991 0.2982278 264.1315
## Tree           346.1120 0.3656705 239.8415
## Neuron Network 352.7021 0.4405791 227.1889
```

Problem 2

Refer to the attached file for the SENIC data set.

Length.of.stay = Average length of stay of all patients in hospital (in days)
Age =Average age of patients (in years)
Infection.risk=Average estimated probability of acquiring infection in hospital (in percent)
Routine.culturing.ratio =Ratio of number of cultures performed to number of patients without signs or symptoms of hospital-acquired infection, times 100
Routine.chest.X.ray.ratio =Ratio of number of X-rays performed to number of patients, without signs or symptoms of pneumonia, times 100
Number.of.beds=Average number of beds in hospital during study period
Medical.school.affiliation =Medical school affiliation, where 1=Yes, 2=No
Region=Geographic region, where: 1 =NE, 2=NC, 3=S, 4=W
Average.daily.census =Average number of patients in hospital per day during study period
Number.of.nurses=Average number of full-time equivalent registered and licensed practical nurses during study period (number full-time plus one half the number part time)
Available.facilities.and.service=Percent of 35 potential facilities and services that are provided by the hospital

Medical school affiliation is the response variable to be coded Y = 1 If Medical school affiliation and Y = 0 if no Medical school affiliation. The pool of potential predictor variables are all variables except Geographic region. Exclude Geographic region from your data set.

a-) Medical school affiliation (Y) is the response variable to be coded Y = 1 If Medical school affiliation and Y = 0 if no Medical school affiliation. The pool of potential predictor variables are all variables except Geographic region. Exclude Geographic region from your data set. (5 points)

See below

```
SENIC <- read.csv("SENIC.csv")
#creating medical acciliation variable
SENIC$Medical.school.affiliation<-I(SENIC$Medical.school.affiliation==1)*1
#dropping geographic region
SENIC1<-SENIC[-8]
```

b-) Build a regression model to predict Medical school affiliation by using all variables on the full data set and comment on your model and perform variable selection (Hint: Use the step function.) Comment on your final model **Ensure that all variables are significant, use** $\alpha = 0.05$. Conduct the Hosmer-Lemeshow goodness of fit test for the appropriateness of the logistic regression function by forming four groups. State the alternatives, decision rule, and conclusion. (10 points)

On the model built on all, Variables Age of person, Number of beds,average daily census, and available facilities and services are significant.

For the model based the variable selection, Age of person, Routine culturing ratio, Number of beds,average daily census, and available facilities and services are significant.

Hosmer-Lemeshow goodness of fit test:

Ho: Fit is good
Ha: Fit is not good

Fail to reject H_0, the fit is good. P value is 0.8186. The fit is good.

```
g<-glm(Medical.school.affiliation~.,data=SENIC1,family=binomial)
summary(g)
```

```
##
## Call:
## glm(formula = Medical.school.affiliation ~ ., family = binomial,
##     data = SENIC1)
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -6.464922   6.812796  -0.949   0.3427
## Length.of.stay                  0.144200   0.325576   0.443   0.6578
## Age                            -0.439280   0.192240  -2.285   0.0223 *
## Infection.risk                  0.036813   0.740616   0.050   0.9604
## Routine.culturing.ratio         0.117773   0.072452   1.626   0.1040
## Routine.chest.X.ray.ratio       0.025102   0.026588   0.944   0.3451
## Number.of.beds                 -0.052865   0.023420  -2.257   0.0240 *
## Average.daily.census            0.082985   0.036865   2.251   0.0244 *
## Number.of.nurses               -0.003326   0.007028  -0.473   0.6361
## Available.facilities.and.services  0.326678   0.129706   2.519   0.0118 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 95.706  on 112  degrees of freedom
## Residual deviance: 29.831  on 103  degrees of freedom
## AIC: 49.831
##
## Number of Fisher Scoring iterations: 9
```

```
g1<-step(g,trace=0)
g1$formula
```

```
## Medical.school.affiliation ~ Age + Routine.culturing.ratio +
##     Number.of.beds + Average.daily.census + Available.facilities.and.services
```

```
g1<-glm(Medical.school.affiliation ~ Age + Routine.culturing.ratio +
    Number.of.beds + Average.daily.census + Available.facilities.and.services,data=SENIC1,family=binomia
summary(g1)
```

```
##
## Call:
## glm(formula = Medical.school.affiliation ~ Age + Routine.culturing.ratio +
##     Number.of.beds + Average.daily.census + Available.facilities.and.services,
##     family = binomial, data = SENIC1)
##
```

```
## Coefficients:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -6.81053    6.77130  -1.006   0.3145
## Age                              -0.31228    0.14280  -2.187   0.0287 *
## Routine.culturing.ratio           0.12125    0.05097   2.379   0.0174 *
## Number.of.beds                   -0.04961    0.02051  -2.419   0.0155 *
## Average.daily.census              0.07350    0.02928   2.510   0.0121 *
## Available.facilities.and.services 0.29629    0.12132   2.442   0.0146 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 95.706  on 112  degrees of freedom
## Residual deviance: 31.812  on 107  degrees of freedom
## AIC: 43.812
##
## Number of Fisher Scoring iterations: 9
```

```
library(ResourceSelection)
hoslem.test(g1$y,fitted(g1),g=4)
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  g1$y, fitted(g1)
## X-squared = 0.40043, df = 2, p-value = 0.8186
```

c-) Suppose that a subject is classified as with disease if p $\geq$ 0.40 and not with disease if p $<$ 0.40. Compute the confusion matrix and comment on model performance.(10 points)

With a 0.4 cutoff, logistic regression achieves Precision = 0.7000, Recall = 0.8235, and $F_1 = 0.7568$ for the positive class (1). Overall accuracy is 0.9204 (about 92%).

```
true_labels <- SENIC1$Medical.school.affiliation
predicted_probabilities <- predict(g1, SENIC1, type = "response")

# use 0.40 cutoff as required
predictions <- ifelse(predicted_probabilities >= 0.40, 1, 0)

# ensure consistent factor levels (0, 1) for both
pred_factor <- factor(predictions, levels = c(0, 1))
true_factor <- factor(true_labels,  levels = c(0, 1))

confusion_matrix <- confusionMatrix(
  pred_factor, true_factor,
  mode = "prec_recall",
  positive = "1"
)
confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 90  3
```

```
##          1  6 14
##
##                Accuracy : 0.9204
##                  95% CI : (0.8542, 0.9629)
##     No Information Rate : 0.8496
##     P-Value [Acc > NIR] : 0.01833
##
##                   Kappa : 0.7095
##
##  Mcnemar's Test P-Value : 0.50499
##
##               Precision : 0.7000
##                  Recall : 0.8235
##                      F1 : 0.7568
##              Prevalence : 0.1504
##          Detection Rate : 0.1239
##    Detection Prevalence : 0.1770
##       Balanced Accuracy : 0.8805
##
##        'Positive' Class : 1
##
```

```
a <- confusion_matrix
Logistic <- a$byClass
```

d-) Use a decision tree to predict the Medical school affiliation and calculate the confusion matrix to evaluate the model performance.

The C5.0 tree achieves Recall = 1.0000 (no false negatives for class 1), Precision = 0.7727, and $F_1 = 0.8718$, with accuracy 0.9558 (about 96%). This comes with 5 false positives (predicting affiliation when the true class is 0).

```
library(C50)
set.seed(304)
g2 <- C5.0(SENIC1[-7], as.factor(SENIC1$Medical.school.affiliation))

# predict using predictors only (exclude the response column)
predicted_class <- predict(g2, SENIC1[-7])

confusion_matrix <- confusionMatrix(
  as.factor(predicted_class),
  as.factor(SENIC1$Medical.school.affiliation),
  mode = "prec_recall",
  positive = "1"
)
confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 91  0
##          1  5 17
##
##                Accuracy : 0.9558
##                  95% CI : (0.8998, 0.9855)
```

```
##     No Information Rate : 0.8496
##     P-Value [Acc > NIR] : 0.000322
##
##                   Kappa : 0.8456
##
##  Mcnemar's Test P-Value : 0.073638
##
##               Precision : 0.7727
##                  Recall : 1.0000
##                      F1 : 0.8718
##              Prevalence : 0.1504
##          Detection Rate : 0.1504
##    Detection Prevalence : 0.1947
##       Balanced Accuracy : 0.9740
##
##        'Positive' Class : 1
##
```

```
a <- confusion_matrix
Tree <- a$byClass
```

e-) Use Neuron Network (NN) with five hidden layers with 8,6,5,3, and 2 neurons respectively (hidden=c(8,6,5,3,2)) with relu activation function (relu <- function(x) {x * (x>=0)}) to predict the Medical school affiliation.Suppose that a subject is classified as with disease if p $\geq$ 0.40 and not with disease if p $<$ 0.40 and compute the confusion matrix evaluate the performance of NN.(10 points)

Note: The ReLU function provided in the question (`relu <- function(x) {x * (x>=0)}`) will cause an error when used with the `neuralnet` package. This is because `neuralnet` requires activation functions that can be differentiated symbolically, and the `>=` operator in the standard ReLU formula cannot be processed by R's symbolic differentiation tools.

To work around this, we use a "smooth ReLU" approximation that behaves nearly identically to ReLU but is fully differentiable:

```
relu <- function(x) { (x + sqrt(x^2 + 1e-8)) / 2 }
```

This is a known limitation of the `neuralnet` package.

The neural network predicts every observation as class 0, so Recall (Sensitivity) for class 1 is 0. Precision and F1 are undefined (NA) because there are no predicted positives. Accuracy is 0.8496 due to class imbalance rather than good classification, and balanced accuracy is 0.5.

```
normalize <- function(x) { return((x - min(x)) / (max(x) - min(x))) }

SENIC_norm <- as.data.frame(lapply(SENIC1, normalize))

# NOTE: The neuralnet package requires differentiable activation functions
# for backpropagation (it uses symbolic differentiation via the Deriv package).
# The standard ReLU: relu <- function(x) { x * (x >= 0) } is not differentiable
# at x = 0 and contains a logical operator that Deriv cannot handle.
#
# We use a smooth approximation: (x + sqrt(x^2 + epsilon)) / 2, which approaches
# ReLU as epsilon → 0 but remains differentiable everywhere.
#
# Alternative: Use softplus: log(1 + exp(x)), which is the natural smooth
# approximation to ReLU, ot leaky_relu <- function(x) { ifelse(x > 0, x, 0.01 * x) }
# which is differentiable everywhere (though not smooth at 0).
```

```r
relu <- function(x) { (x + sqrt(x^2 + 1e-8)) / 2 }

set.seed(304)
c3_nn <- neuralnet(
  Medical.school.affiliation ~ .,
  data = SENIC_norm,
  hidden = c(8,6,5,3,2),
  act.fct = relu,
  linear.output = FALSE
)
plot(c3_nn)

true_labels <- SENIC1$Medical.school.affiliation

# predicted probabilities/scores (use compute, not predict)
x_mat <- SENIC_norm[, setdiff(names(SENIC_norm), "Medical.school.affiliation")]
predicted_probabilities <- compute(c3_nn, x_mat)$net.result

# use 0.40 cutoff (as required)
predictions <- ifelse(predicted_probabilities >= 0.40, 1, 0)

pred_factor <- factor(predictions, levels = c(0, 1))
true_factor <- factor(true_labels, levels = c(0, 1))

confusion_matrix <- confusionMatrix(
  pred_factor, true_factor,
  mode = "prec_recall",
  positive = "1"
)
confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 96 17
##          1  0  0
##
##                Accuracy : 0.8496
##                  95% CI : (0.7701, 0.9099)
##     No Information Rate : 0.8496
##     P-Value [Acc > NIR] : 0.5642363
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.0001042
##
##               Precision :     NA
##                  Recall : 0.0000
##                      F1 :     NA
##              Prevalence : 0.1504
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
```

```
##
##        'Positive' Class : 1
##
```

```
a <- confusion_matrix
NN <- a$byClass
```

f-) which model would you choose?

Based on the confusion-matrix metrics, the C5.0 tree is the best model: it has the highest F_1 (0.8718) and balanced accuracy (0.9739), and it achieves Recall = 1.0000 while maintaining Precision = 0.7727. Logistic regression is weaker (F1 = 0.7568), and the neural network fails to identify any positives (Precision and F1 are NA because it predicts no positives).

```
data.frame(rbind(Logistic,Tree, NN))
```

```
##          Sensitivity Specificity Pos.Pred.Value Neg.Pred.Value Precision
## Logistic   0.8235294   0.9375000      0.7000000      0.9677419 0.7000000
## Tree       1.0000000   0.9479167      0.7727273      1.0000000 0.7727273
## NN         0.0000000   1.0000000            NaN      0.8495575        NA
##             Recall        F1 Prevalence Detection.Rate Detection.Prevalence
## Logistic 0.8235294 0.7567568  0.1504425      0.1238938            0.1769912
## Tree     1.0000000 0.8717949  0.1504425      0.1504425            0.1946903
## NN       0.0000000        NA  0.1504425      0.0000000            0.0000000
##          Balanced.Accuracy
## Logistic         0.8805147
## Tree             0.9739583
## NN               0.5000000
```