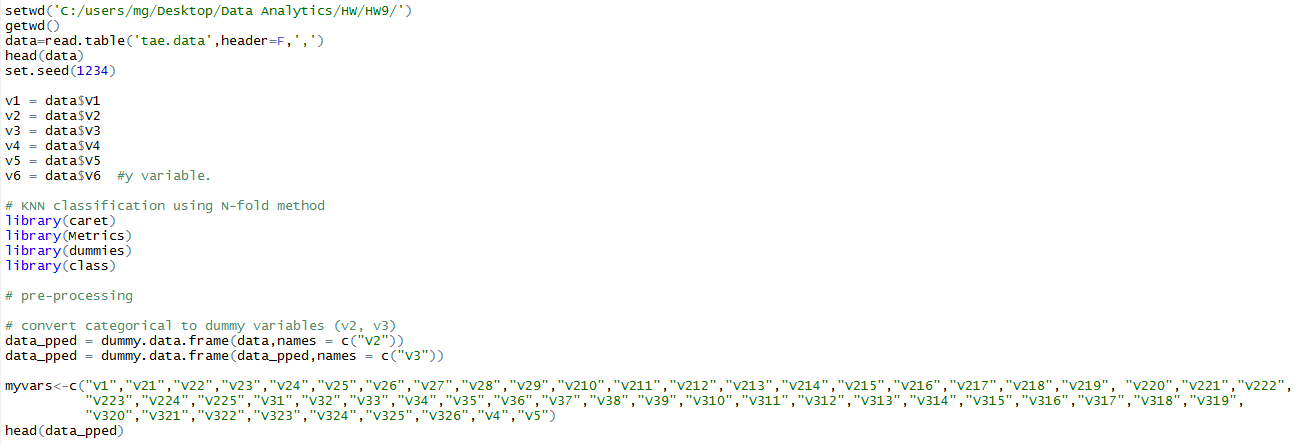
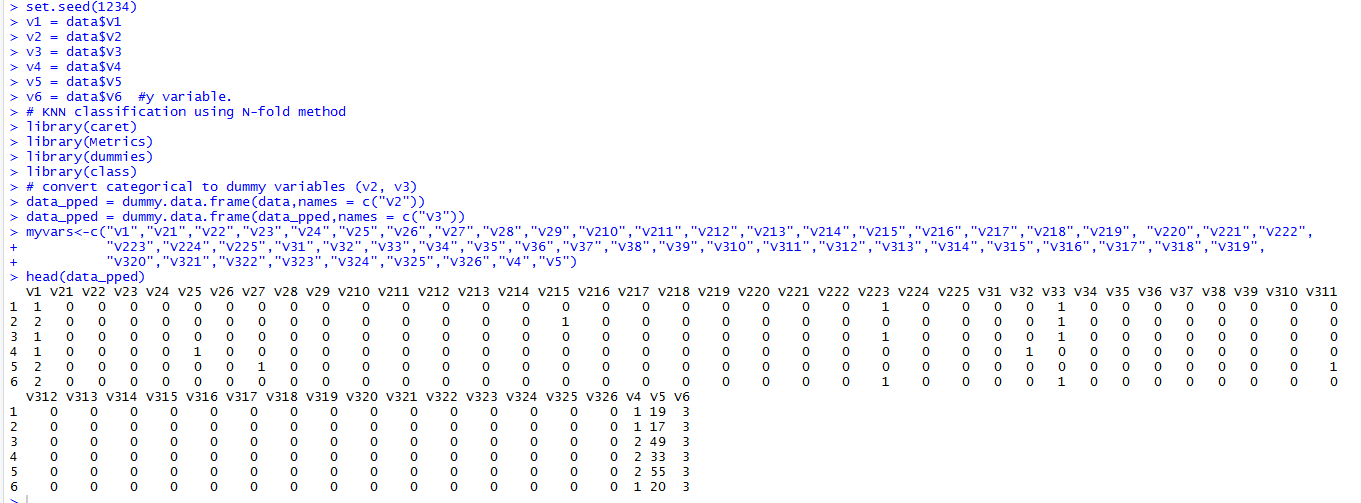
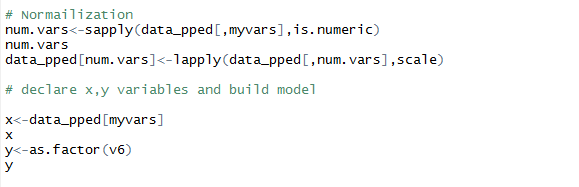
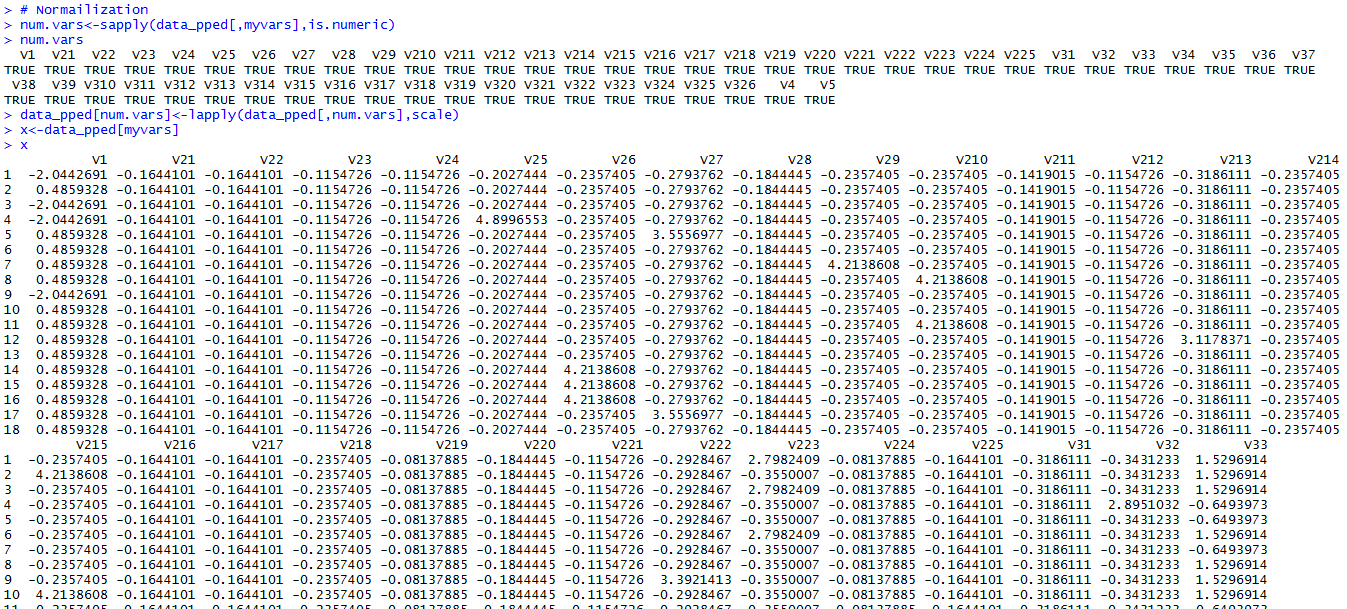
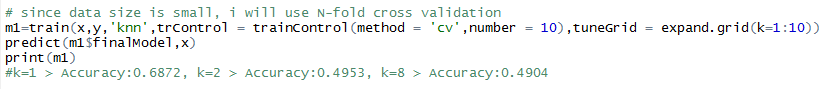
**Using KNN**

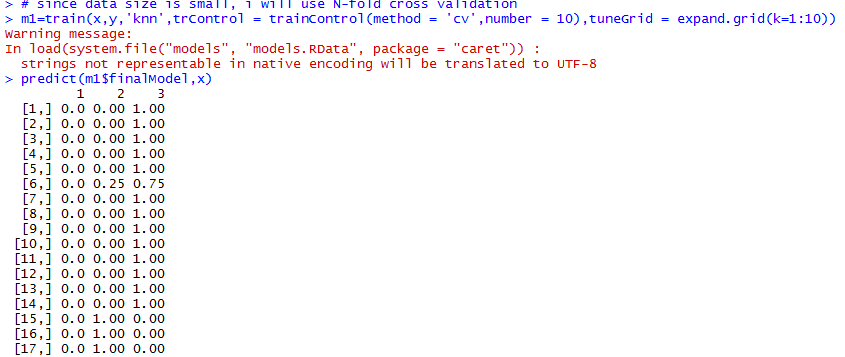


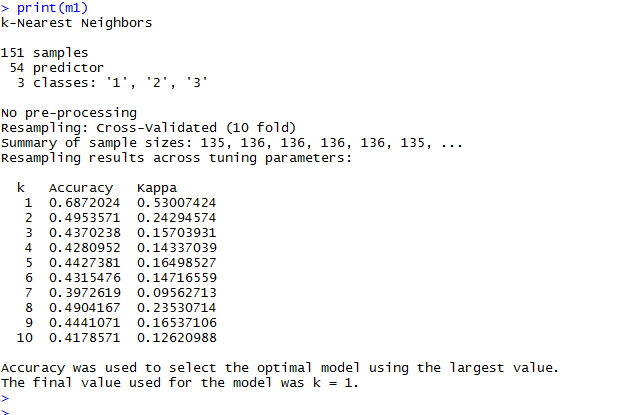












**KNN code**

"""

Objective:

This is the data on the teaching performance of TA (151 students) during the summer / general semester.

So, can we predict the y variable (teaching performance of TA) with x variables?

"""

"""

Attribute Information:

1. Whether of not the TA is a native English speaker (binary); 1=English speaker, 2=non-English speaker

2. Course instructor (categorical, 25 categories)

3. Course (categorical, 26 categories)

4. Summer or regular semester (binary) 1=Summer, 2=Regular

5. Class size (numerical)

6. Class attribute (categorical) 1=Low, 2=Medium, 3=High

Number of Instances: 151

Number of Attributes: 6 (including the class attribute)

Missing Attribute Values: None

"""

setwd('C:/users/mg/Desktop/Data Analytics/HW/HW9/')

getwd()

data=read.table('tae.data',header=F,',')

head(data)

set.seed(1234)

v1 = data$V1

v2 = data$V2

v3 = data$V3

v4 = data$V4

v5 = data$V5

v6 = data$V6 #y variable.

# KNN classification using N-fold method

library(caret)

library(Metrics)

library(dummies)

library(class)

# pre-processing

# convert categorical to dummy variables (v2, v3)

data\_pped = dummy.data.frame(data,names = c("V2"))

data\_pped = dummy.data.frame(data\_pped,names = c("V3"))

myvars<-c("V1","V21","V22","V23","V24","V25","V26","V27","V28","V29","V210","V211","V212","V213","V214","V215","V216","V217","V218","V219", "V220","V221","V222",

"V223","V224","V225","V31","V32","V33","V34","V35","V36","V37","V38","V39","V310","V311","V312","V313","V314","V315","V316","V317","V318","V319",

"V320","V321","V322","V323","V324","V325","V326","V4","V5")

head(data\_pped)

# Normailization

num.vars<-sapply(data\_pped[,myvars],is.numeric)

num.vars

data\_pped[num.vars]<-lapply(data\_pped[,num.vars],scale)

# declare x,y variables and build model

x<-data\_pped[myvars]

x

y<-as.factor(v6)

y

# since data size is small, i will use N-fold cross validation

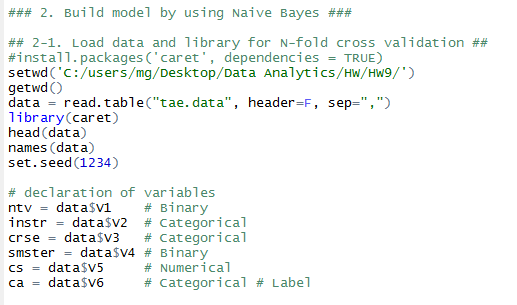
m1=train(x,y,'knn',trControl = trainControl(method = 'cv',number = 10),tuneGrid = expand.grid(k=1:10))

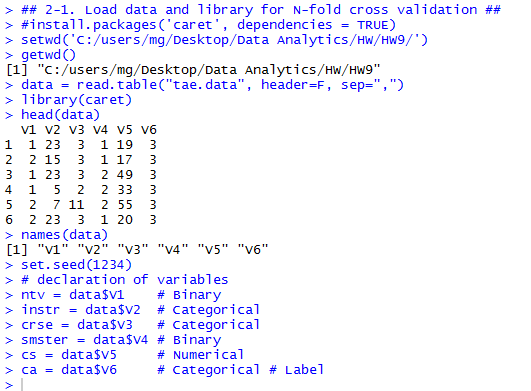
predict(m1$finalModel,x)

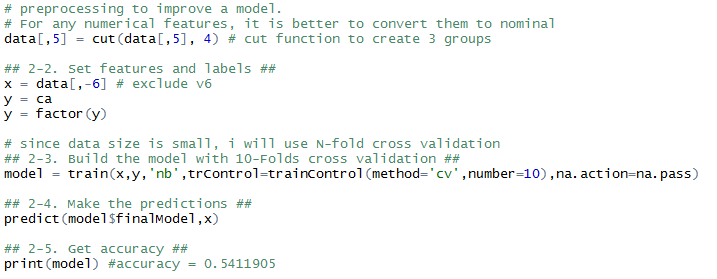
print(m1)

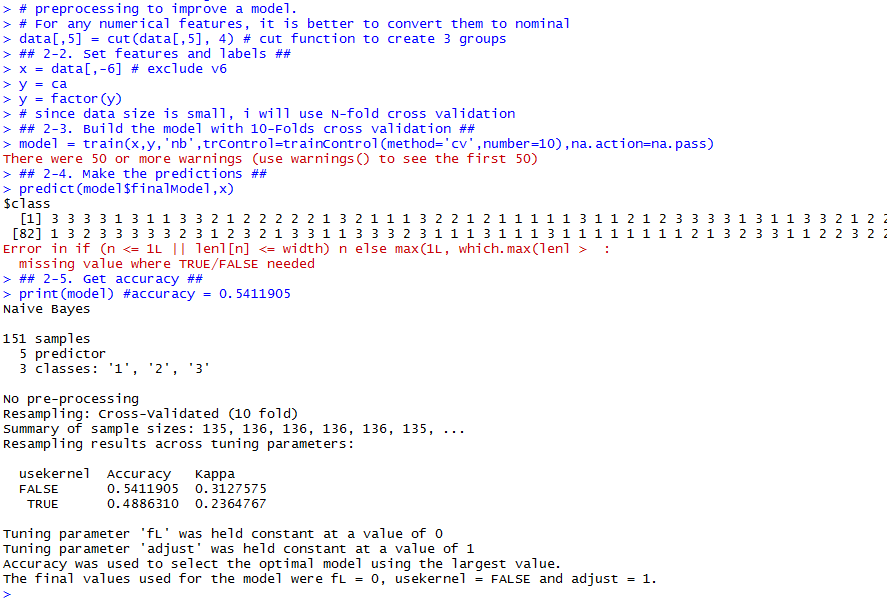
#k=1 > Accuracy:0.6872, k=2 > Accuracy:0.4953, k=8 > Accuracy:0.4904

**Using Naïve Bayes**









**NB code**

"""

Objective:

This is the data on the teaching performance of TA (151 students) during the summer / general semester.

So, can we predict the y variable (teaching performance of TA) with x variables?

"""

"""

Attribute Information:

1. Whether of not the TA is a native English speaker (binary); 1=English speaker, 2=non-English speaker

2. Course instructor (categorical, 25 categories)

3. Course (categorical, 26 categories)

4. Summer or regular semester (binary) 1=Summer, 2=Regular

5. Class size (numerical)

6. Class attribute (categorical) 1=Low, 2=Medium, 3=High

Number of Instances: 151

Number of Attributes: 6 (including the class attribute)

Missing Attribute Values: None

"""

### 2. Build model by using Naive Bayes ###

## 2-1. Load data and library for N-fold cross validation ##

#install.packages('caret', dependencies = TRUE)

setwd('C:/users/mg/Desktop/Data Analytics/HW/HW9/')

getwd()

data = read.table("tae.data", header=F, sep=",")

library(caret)

head(data)

names(data)

set.seed(1234)

# declaration of variables

ntv = data$V1 # Binary

instr = data$V2 # Categorical

crse = data$V3 # Categorical

smster = data$V4 # Binary

cs = data$V5 # Numerical

ca = data$V6 # Categorical # Label

# preprocessing to improve a model.

# For any numerical features, it is better to convert them to nominal

data[,5] = cut(data[,5], 4) # cut function to create 3 groups

## 2-2. Set features and labels ##

x = data[,-6] # exclude v6

y = ca

y = factor(y)

# since data size is small, i will use N-fold cross validation

## 2-3. Build the model with 10-Folds cross validation ##

model = train(x,y,'nb',trControl=trainControl(method='cv',number=10),na.action=na.pass)

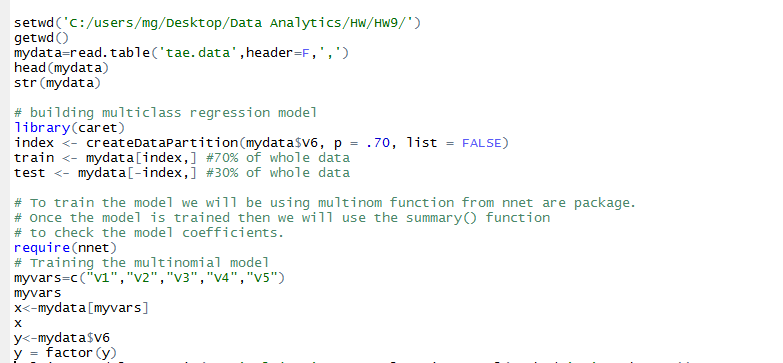
## 2-4. Make the predictions ##

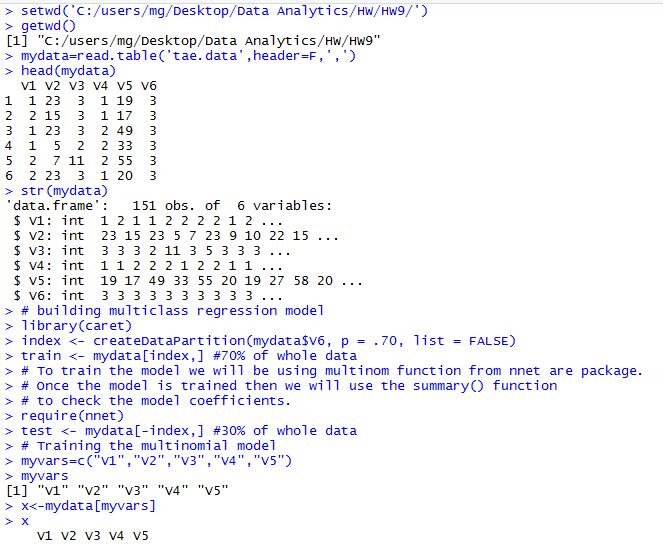
predict(model$finalModel,x)

## 2-5. Get accuracy ##

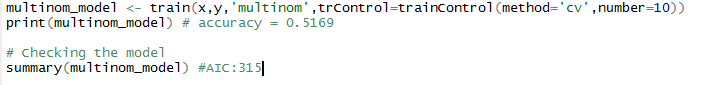
print(model) #accuracy = 0.5411905

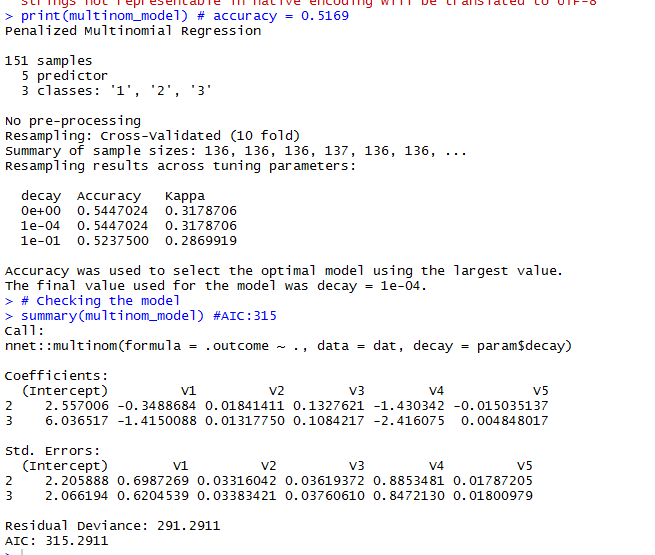
**Using Logistic Regression**

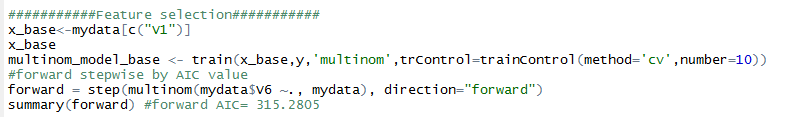


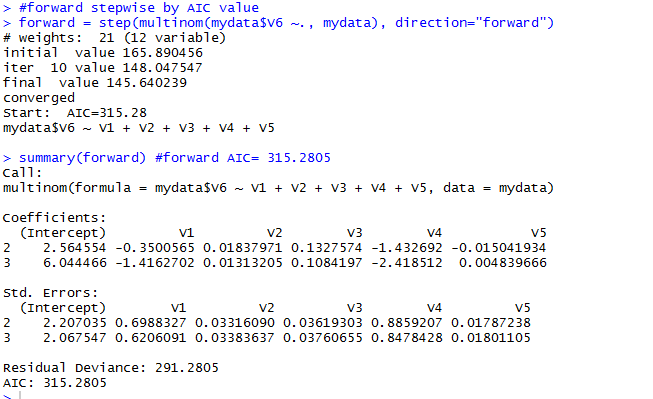


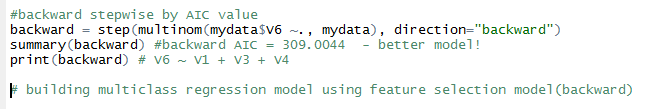


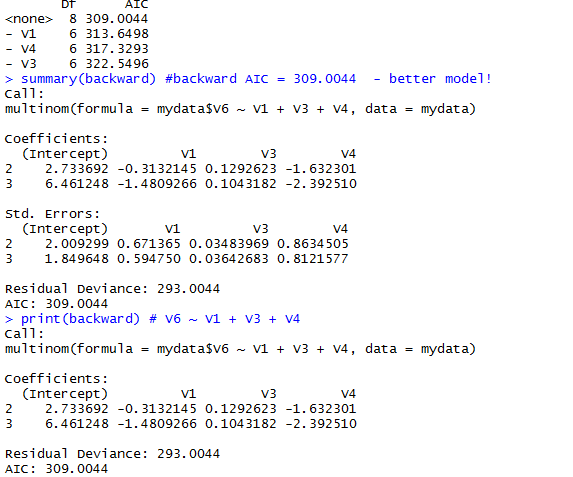


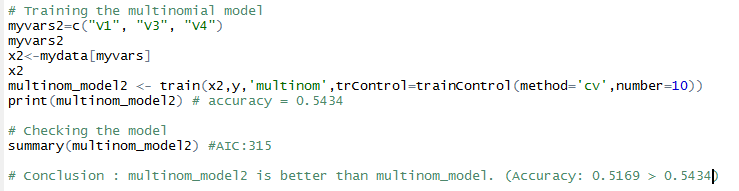


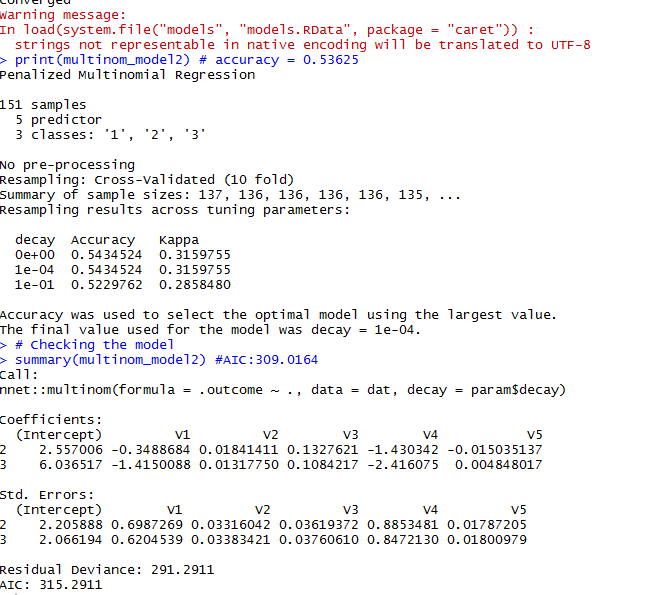












**Logistic Regression code**

"""

Objective:

This is the data on the teaching performance of TA (151 students) during the summer / general semester.

So, can we predict the y variable (teaching performance of TA) with x variables?

"""

"""

Attribute Information:

1. Whether of not the TA is a native English speaker (binary); 1=English speaker, 2=non-English speaker

2. Course instructor (categorical, 25 categories)

3. Course (categorical, 26 categories)

4. Summer or regular semester (binary) 1=Summer, 2=Regular

5. Class size (numerical)

6. Class attribute (categorical) 1=Low, 2=Medium, 3=High

Number of Instances: 151

Number of Attributes: 6 (including the class attribute)

Missing Attribute Values: None

"""

setwd('C:/users/mg/Desktop/Data Analytics/HW/HW9/')

getwd()

mydata=read.table('tae.data',header=F,',')

head(mydata)

str(mydata)

# building multiclass regression model

library(caret)

index <- createDataPartition(mydata$V6, p = .70, list = FALSE)

train <- mydata[index,] #70% of whole data

test <- mydata[-index,] #30% of whole data

# To train the model we will be using multinom function from nnet are package.

# Once the model is trained then we will use the summary() function

# to check the model coefficients.

require(nnet)

# Training the multinomial model

myvars=c("V1","V2","V3","V4","V5")

myvars

x<-mydata[myvars]

x

y<-mydata$V6

y = factor(y)

multinom\_model <- train(x,y,'multinom',trControl=trainControl(method='cv',number=10))

print(multinom\_model) # accuracy = 0.5169

# Checking the model

summary(multinom\_model) #AIC:315

###########Feature selection###########

x\_base<-mydata[c("V1")]

x\_base

multinom\_model\_base <- train(x\_base,y,'multinom',trControl=trainControl(method='cv',number=10))

#forward stepwise by AIC value

forward = step(multinom(mydata$V6 ~., mydata), direction="forward")

summary(forward) #forward AIC= 315.2805

#backward stepwise by AIC value

backward = step(multinom(mydata$V6 ~., mydata), direction="backward")

summary(backward) #backward AIC = 309.0044 - better model!

print(backward) # V6 ~ V1 + V3 + V4

# building multiclass regression model using feature selection model(backward)

# Training the multinomial model

myvars2=c("V1", "V3", "V4")

myvars2

x2<-mydata[myvars]

x2

multinom\_model2 <- train(x2,y,'multinom',trControl=trainControl(method='cv',number=10))

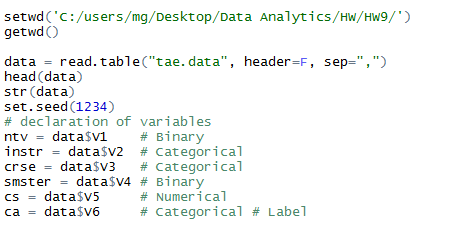
print(multinom\_model2) # accuracy = 0.5434

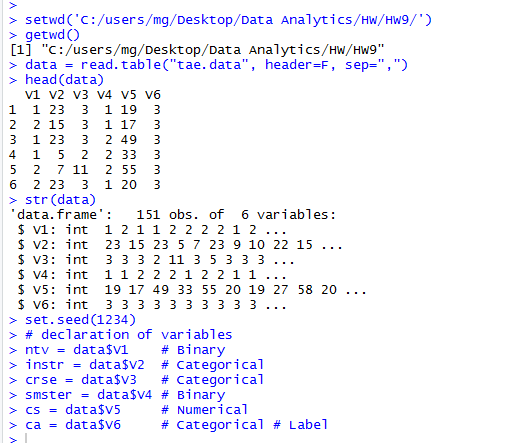
# Checking the model

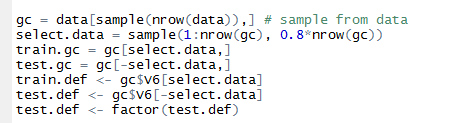
summary(multinom\_model2) #AIC:315

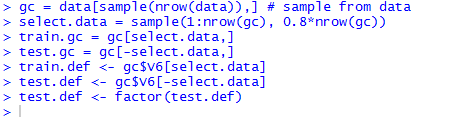
# Conclusion : multinom\_model2 is better than multinom\_model. (Accuracy: 0.5169 > 0.5434)

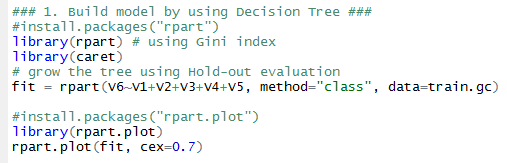
**Using Decision Tree**

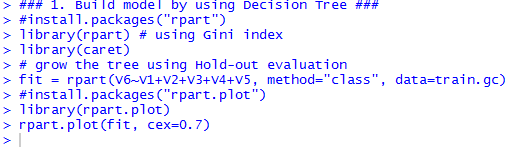


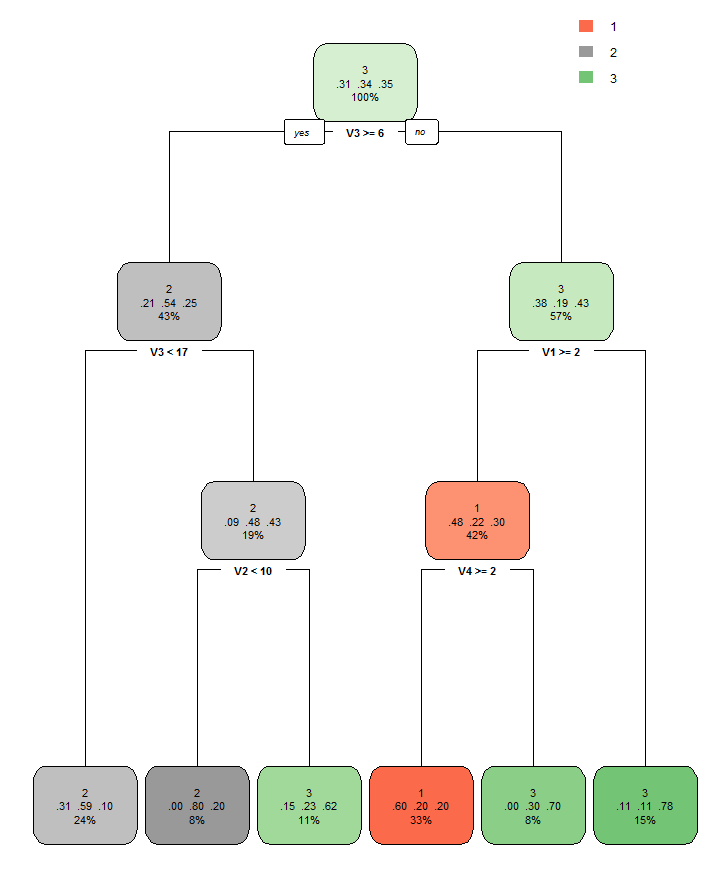


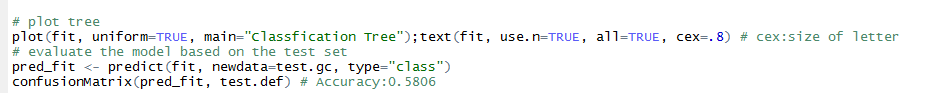


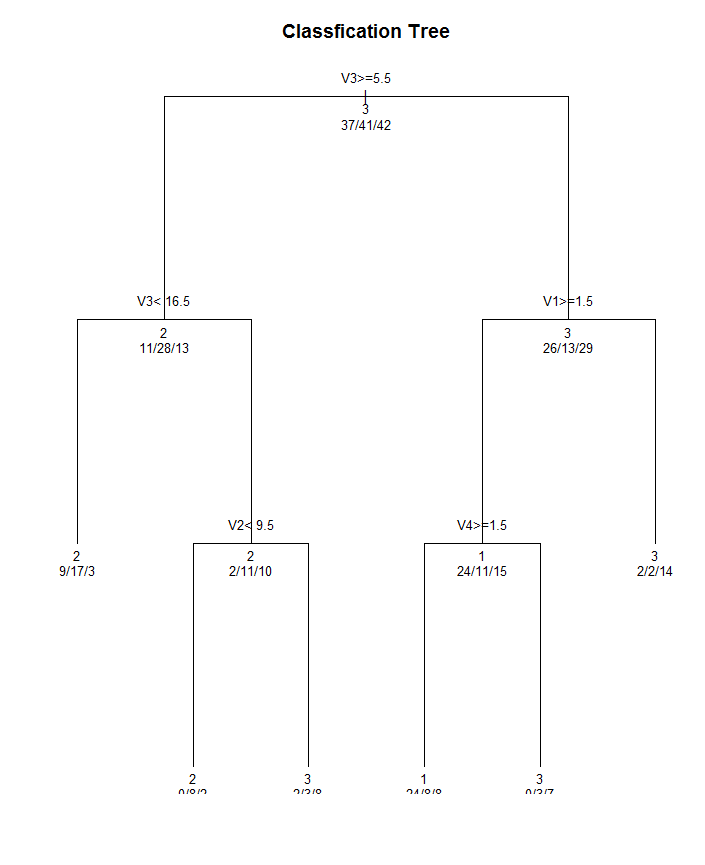


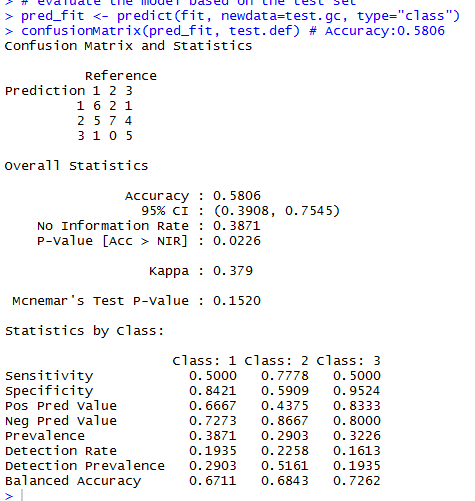


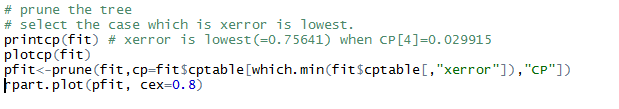


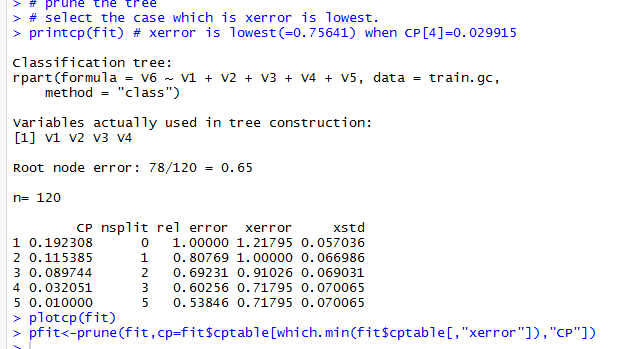


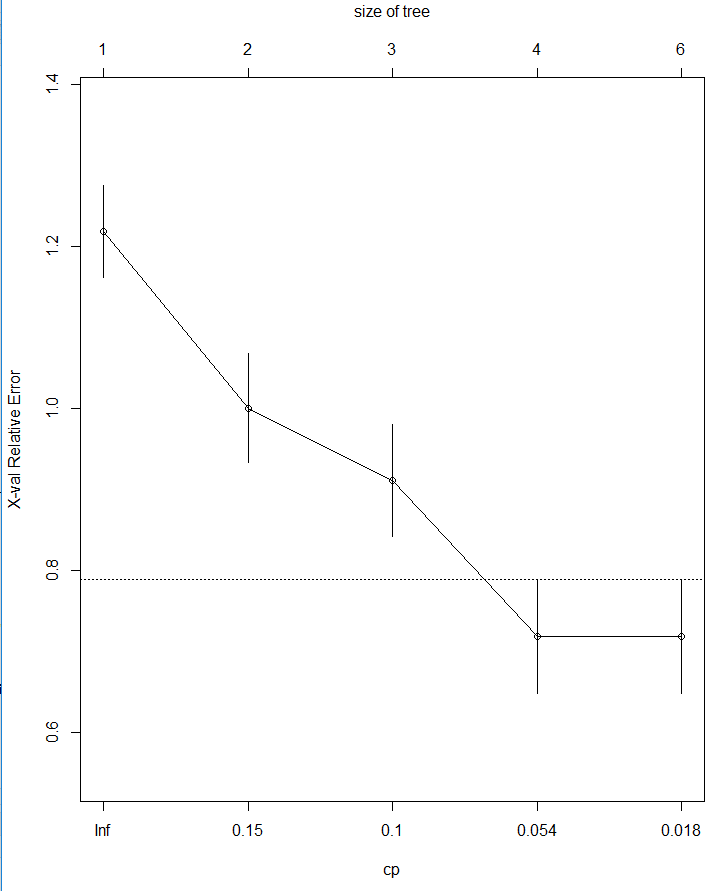


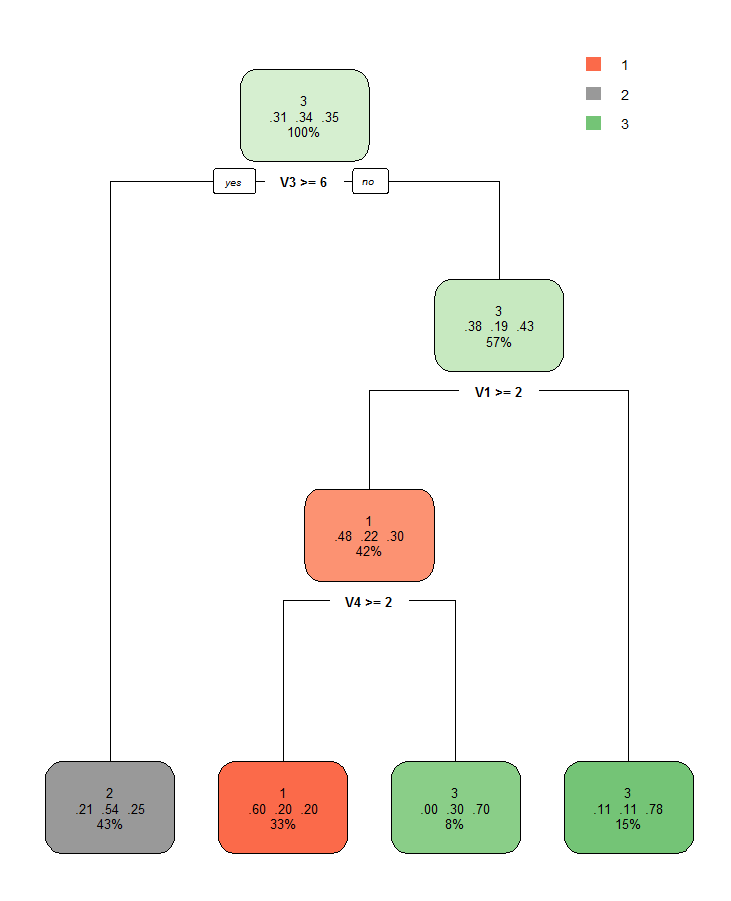


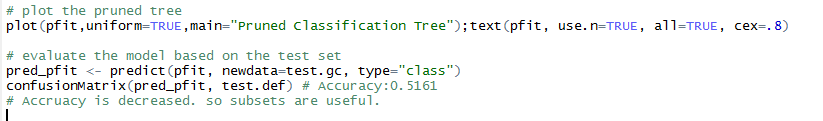


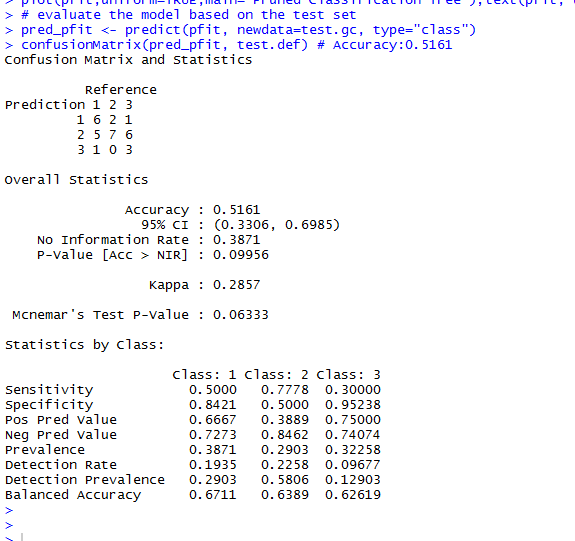


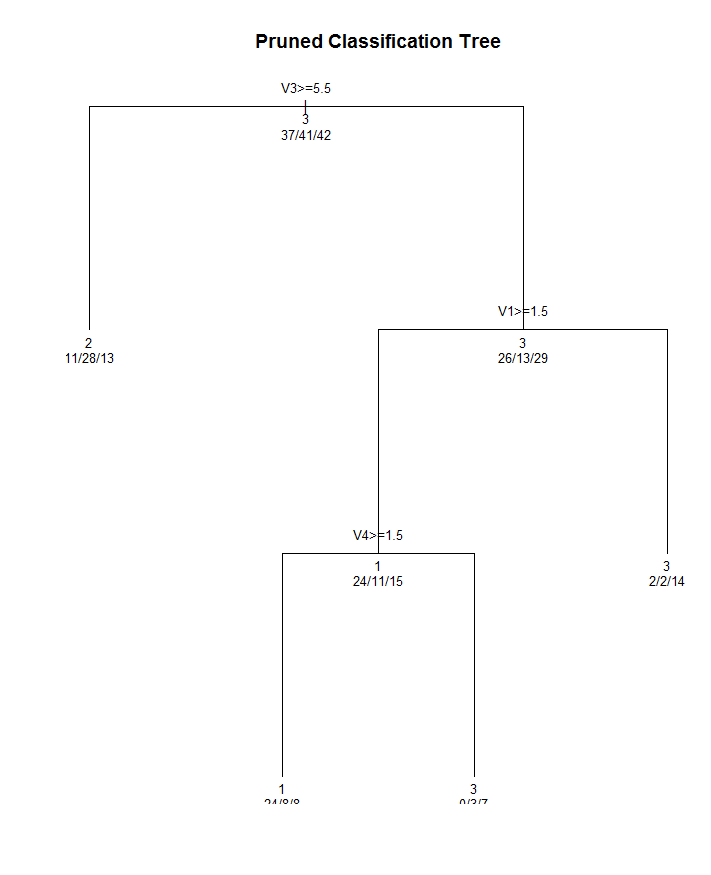












**Decision Tree code**

"""

Attribute Information:

1. Whether of not the TA is a native English speaker (binary); 1=English speaker, 2=non-English speaker

2. Course instructor (categorical, 25 categories)

3. Course (categorical, 26 categories)

4. Summer or regular semester (binary) 1=Summer, 2=Regular

5. Class size (numerical)

6. Class attribute (categorical) 1=Low, 2=Medium, 3=High

Number of Instances: 151

Number of Attributes: 6 (including the class attribute)

Missing Attribute Values: None

"""

setwd('C:/users/mg/Desktop/Data Analytics/HW/HW9/')

getwd()

data = read.table("tae.data", header=F, sep=",")

head(data)

str(data)

set.seed(1234)

# declaration of variables

ntv = data$V1 # Binary

instr = data$V2 # Categorical

crse = data$V3 # Categorical

smster = data$V4 # Binary

cs = data$V5 # Numerical

ca = data$V6 # Categorical # Label

gc = data[sample(nrow(data)),] # sample from data

select.data = sample(1:nrow(gc), 0.8\*nrow(gc))

train.gc = gc[select.data,]

test.gc = gc[-select.data,]

train.def <- gc$V6[select.data]

test.def <- gc$V6[-select.data]

test.def <- factor(test.def)

### 1. Build model by using Decision Tree ###

#install.packages("rpart")

library(rpart) # using Gini index

library(caret)

# grow the tree using Hold-out evaluation

fit = rpart(V6~V1+V2+V3+V4+V5, method="class", data=train.gc)

#install.packages("rpart.plot")

library(rpart.plot)

rpart.plot(fit, cex=0.7)

# plot tree

plot(fit, uniform=TRUE, main="Classfication Tree");text(fit, use.n=TRUE, all=TRUE, cex=.8) # cex:size of letter

# evaluate the model based on the test set

pred\_fit <- predict(fit, newdata=test.gc, type="class")

confusionMatrix(pred\_fit, test.def) # Accuracy:0.5806

# prune the tree

# select the case which is xerror is lowest.

printcp(fit) # xerror is lowest(=0.75641) when CP[4]=0.029915

plotcp(fit)

pfit<-prune(fit,cp=fit$cptable[which.min(fit$cptable[,"xerror"]),"CP"])

rpart.plot(pfit, cex=0.8)

# plot the pruned tree

plot(pfit,uniform=TRUE,main="Pruned Classification Tree");text(pfit, use.n=TRUE, all=TRUE, cex=.8)

# evaluate the model based on the test set

pred\_pfit <- predict(pfit, newdata=test.gc, type="class")

confusionMatrix(pred\_pfit, test.def) # Accuracy:0.5161

# Accruacy is decreased. so subsets are useful.

**Conclusion:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | KNN | Naïve Bayes | Logistic Regression | Decision Tree |
| Accuracy | 0.6872 | 0.5411 | 0.5434 | 0.5806 |

Given highest accuracy in the each of methods, accuracy of KNN is highest.

Therefore, the model from KNN is the best.