STAT 574

Derrick Edwards

Homework 3

Problem 1. Use the data in the file “card\_transdata.csv” to fit a naïve Bayes binary classifier. Compute prediction accuracy.

**R Output**

"accuracy= 64.99 %"

**R Code**

# Data

card\_data <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/HW1STAT574S23/DATA SETS/card\_transdata.csv")

# split data 80 train 20 test

set.seed(389284)

sample <- sample(c(TRUE, FALSE), nrow(card\_data), replace=TRUE,

prob = c(0.8,0.2))

train <- card\_data[sample,]

test <- card\_data[!sample,]

test.x <- data.matrix(test[-8])

test.y <- data.matrix(test[8])

print(names(train))

# fit naive bayes binary classifier

library(e1071)

nb.class <- naiveBayes(as.factor(fraud) ~ distance\_from\_home +

distance\_from\_last\_transaction +

ratio\_to\_median\_purchase\_price + repeat\_retailer +

used\_chip + used\_pin\_number + online\_order,

data=train)

# compute prediction accuracy for testing data

pred.y <- as.numeric(predict(nb.class, test.x))-1

match <- c()

for(i in 1:length(pred.y)){

match[i] <- ifelse(test.y[i]==pred.y[i],1,0)

}

print(paste('accuracy=', round(mean(match)\*100, digits = 2), '%'))

**Python Output**

Accuracy: 53.75 %

**Python Code**

*# Problem 1 - Naive Bayes Binary Classification*

*# compute prediction accuracy*

import pandas

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

card\_data=pandas.read\_csv(**r**'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data Mining\HW1STAT574S23\DATA SETS\card\_transdata.csv')

X=card\_data.iloc[:,0:7].values

y=card\_data.iloc[:,7]

*#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS*

X\_train, X\_test, y\_train, y\_test=train\_test\_split(X, y, test\_size=0.20,

random\_state=9994445)

*#FITTING NAIVE BAYES BINARY CLASSIFIER*

from sklearn.naive\_bayes import GaussianNB

gauss\_nb=GaussianNB()

gauss\_nb.fit(X\_train, y\_train)

*#COMPUTING PREDICTION ACCURACY FOR TESTING DATA*

y\_pred = gauss\_nb.predict(X\_test)

print('Accuracy:', round(metrics.accuracy\_score(y\_test, y\_pred)\*100, 2),'%')

Problem 2. Use the data in the file “concussions\_data.csv” to fit a naïve Bayes multinomial classifier. Compute prediction accuracy.

**R Output**

"accuracy= 77.88 %"

**R Code**

concuss\_data <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/HW1STAT574S23/DATA SETS/concussions\_data.csv")

# split data test and train

set.seed(3849433)

sample <- sample(c(TRUE,FALSE), nrow(concuss\_data), replace=TRUE,

prob=c(0.8,0.2))

train <- concuss\_data[sample,]

test <- concuss\_data[!sample,]

test.x <- data.matrix(test[-5])

test.y <- data.matrix(test[5])

print(names(concuss\_data))

#FITTING NAIVE BAYES BINARY CLASSIFIER

library(e1071)

nb.multiclass<- naiveBayes(as.factor(concussion) ~ age + nyearsplaying +

position + prevconc, data=train)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA

pred.y<- as.numeric(predict(nb.multiclass, test.x))

print(paste('accuracy=', round((1-mean(test.y!=pred.y))\*100, digits=2), '%'))

**Python Output**

Accuracy: 0.9333333333333333

**Python Code**

*# Problem 2 - Naive Bayes Multinomial Classification*

*# compute prediction accuracy*

import pandas

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from statistics import mean

from sklearn.metrics import accuracy\_score

concussion\_data=pandas.read\_csv(**r**'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data Mining\HW1STAT574S23\DATA SETS\concussions\_data.csv')

code\_position={'Offensive Lineman': 1, 'Cornerback': 2, 'Running Back': 3,'Wide Receiver': 4,

'Quarterback': 5}

code\_concussion={'mild': 1, 'moderate': 2, 'severe': 3}

concussion\_data['position']=concussion\_data['position'].map(code\_position)

concussion\_data['concussion']=concussion\_data['concussion'].map(code\_concussion)

X=concussion\_data.iloc[:,0:4]

y=concussion\_data.iloc[:,4]

*#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS*

X\_train, X\_test, y\_train, y\_test=train\_test\_split(X, y, test\_size=0.20,

random\_state=457752)

*#FITTING NAIVE BAYES BINARY CLASSIFIER*

gnb=GaussianNB()

gnb.fit(X\_train, y\_train)

*#COMPUTING PREDICTION ACCURACY FOR TESTING DATA*

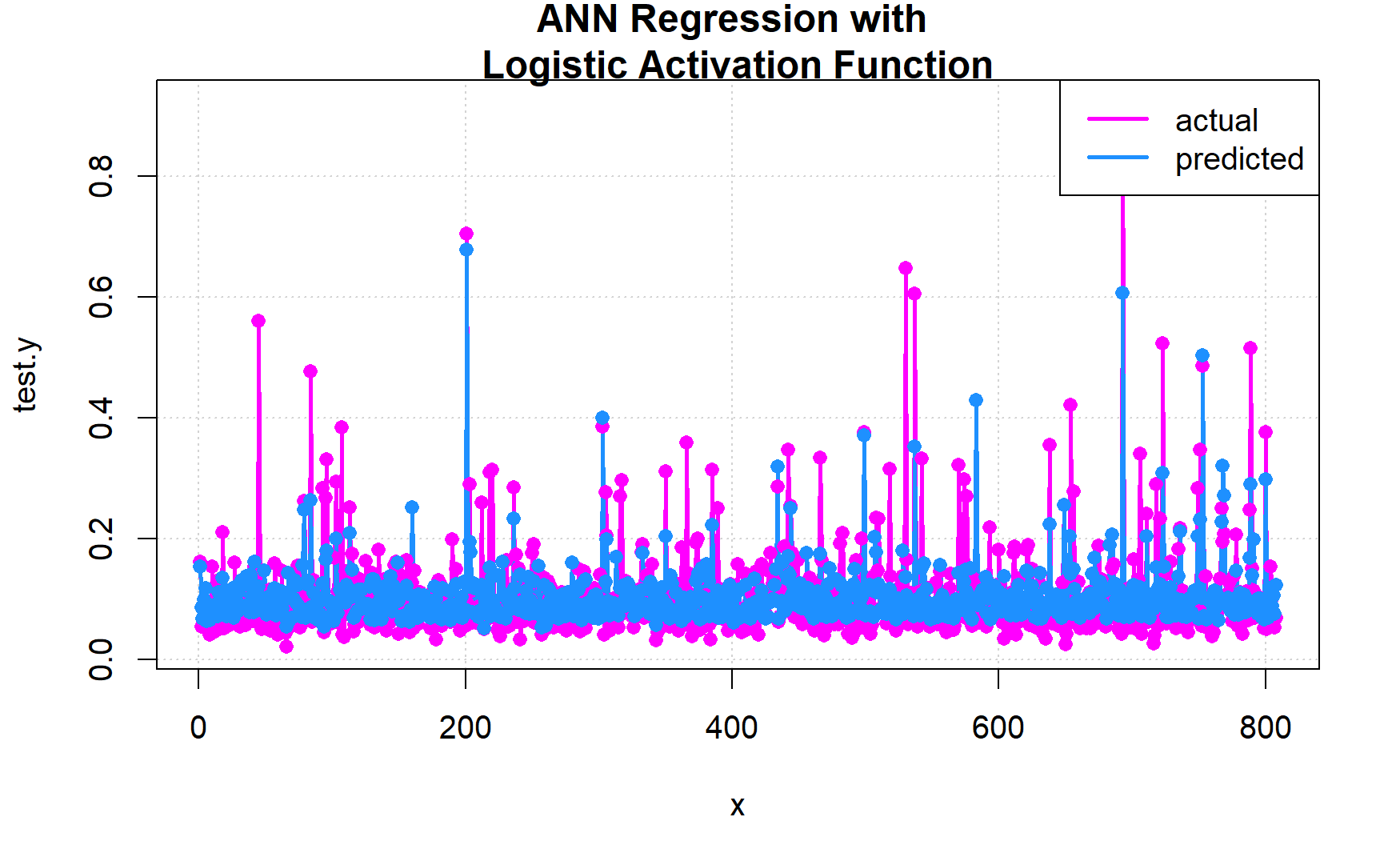
y\_pred=gnb.predict(X\_test)

accuracy=accuracy\_score(y\_test, y\_pred)

print("Accuracy: ", accuracy)

Problem 3. Use the data in the file “hospital\_data.csv” to fit an artificial neural network. Plot the actual and predicted values and compute prediction accuracy within 10%, 15%, and 20% of the true values.

**R Output**



[1] "Prediction Accuracy"

[1] "within 10%: 0.2215"

[1] "within 15%: 0.3082"

[1] "within 20%: 0.4059"

**R Code**

# Data

hospital\_data <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/HW1STAT574S23/DATA SETS/hospital\_data.csv")

hospital\_data$gender <- ifelse(hospital\_data$gender=='M',1,0)

# remove medical ID

hospital\_data <- hospital\_data[,-1]

View(hospital\_data)

#SCALING VARIABLES TO FALL IN [0,1]

library(dplyr)

scale01 <- function(x){

(x-min(x))/(max(x)-min(x))

}

hospital\_data <- hospital\_data %>% mutate\_all(scale01)

# split data to test and train

set.seed(3848058)

sample <- sample(c(TRUE, FALSE), nrow(hospital\_data), replace=TRUE,

prob=c(0.8,0.2))

train<- hospital\_data[sample,]

test<- hospital\_data[!sample,]

test.x<- data.matrix(test[-6])

test.y<- data.matrix(test[6])

print(names(hospital\_data))

#FITTING ANN WITH LOGISTIC ACTIVATION FUNCTION

library(neuralnet)

ann.reg<- neuralnet(surgery\_cost ~ gender + age + ASA + BMI

+ surgery\_duration\_min, data=train,

hidden=3, act.fct="logistic")

#PLOTTING THE DIAGRAM

plot(ann.reg)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA

pred.y<- predict(ann.reg, test.x)

#accuracy within 10%

accuracy10<- ifelse(abs(test.y-pred.y)<0.10\*test.y,1,0)

#accuracy within 15%

accuracy15<- ifelse(abs(test.y-pred.y)<0.15\*test.y,1,0)

#accuracy within 20%

accuracy20<- ifelse(abs(test.y-pred.y)<0.20\*test.y,1,0)

print('Prediction Accuracy')

print(paste('within 10%:', round(mean(accuracy10),4)))

print(paste('within 15%:', round(mean(accuracy15),4)))

print(paste('within 20%:', round(mean(accuracy20),4)))

#PLOTTING ACTUAL AND PREDICTED VALUES FOR TESTING DATA

x<- 1:length(test.y)

plot(x, test.y, type="l", lwd=2, col="magenta", main="ANN Regression with

Logistic Activation Function", panel.first=grid())

lines(x, pred.y, lwd=2, col="dodgerblue")

points(x,test.y, pch=16, col="magenta")

points(x, pred.y, pch=16, col="dodgerblue")

legend("topright", c("actual", "predicted"), lty=1, lwd=2,

col=c("magenta","dodgerblue"))

**Python Output**

accuracy within 10% = 0

accuracy within 15% = 0

accuracy within 20% = 0

Chart, histogram

Description automatically generated

**Python Code**

import numpy

import pandas

from statistics import mean

*# read in data*

hospital=pandas.read\_csv(**r**'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data Mining\HW1STAT574S23\DATA SETS\hospital\_data.csv')

coding={'M': 1, 'F': 0}

hospital['gender']=hospital['gender'].map(coding)

*#SCALING VARIABLES TO FALL IN [0,1]*

from sklearn import preprocessing

scaler=preprocessing.MinMaxScaler()

scaler\_fit=scaler.fit\_transform(hospital)

scaled\_housing\_data=pandas.DataFrame(scaler\_fit, columns=hospital.columns)

X=hospital.iloc[:,1:6].values

y=hospital.iloc[:,6].values

*#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS*

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test=train\_test\_split(X, y, test\_size=0.20,

random\_state=7950321)

*#FITTING AN ARTIFICIAL NEURAL NETWORK*

from keras.models import Sequential

from keras.layers import Dense

model=Sequential()

*#Defining the input layer and one hidden layer*

model.add(Dense(units=3, input\_dim=5, kernel\_initializer='uniform',

activation='tanh'))

*#Defining the output neuron*

model.add(Dense(1))

*#Compiling the model*

model.compile(loss='mean\_squared\_error')

*#Fitting the ANN to the training set*

model.fit(X\_train, y\_train)

*#COMPUTING PREDICTION ACCURACY FOR TESTING DATA*

y\_pred=model.predict(X\_test)

ind10=[]

ind15=[]

ind20=[]

for sub1, sub2 in zip(y\_pred, y\_test):

    ind10.append(1) if abs(sub1-sub2)<0.10\*sub2 else ind10.append(0)

    ind15.append(1) if abs(sub1-sub2)<0.15\*sub2 else ind15.append(0)

    ind20.append(1) if abs(sub1-sub2)<0.20\*sub2 else ind20.append(0)

*#accuracy within 10%*

accuracy10=mean(ind10)

print('accuracy within 10% =', accuracy10)

*#accuracy within 15%*

accuracy15=mean(ind15)

print('accuracy within 15% =', accuracy15)

*#accuracy within 20%*

accuracy20=mean(ind20)

print('accuracy within 20% =', accuracy20)

*#plotting actual and predicted obsevations vs. observation number*

import matplotlib.pyplot as plt

n\_obs=list(range(0,len(y\_test)))

plt.plot(n\_obs, y\_test, label="actual")

plt.plot(n\_obs, y\_pred, label="predicted")

plt.xlabel('n\_obs')

plt.ylabel('median\_house\_value')

plt.title('ANN Regression')

plt.legend()

plt.show()

Problem 4. Use the data in the file “card\_transdata.csv” to fit an artificial neural network. Compute prediction accuracy.

**R Output**

[1] "accuracy= 0.0221" – logistic activation

[1] "accuracy= 0.0221" – logistic activation

[1] "accuracy= 0.0172" – tanh activation

**R Code**

# Data

card\_data <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/HW1STAT574S23/DATA SETS/card\_transdata.csv")

#SCALING VARIABLES TO FALL IN [0,1]

library(dplyr)

scale01 <- function(x){

(x-min(x))/(max(x)-min(x))

}

card\_data<- card\_data %>% mutate\_all(scale01)

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS

set.seed(839754)

sample <- sample(c(TRUE, FALSE), nrow(card\_data), replace=TRUE, prob=c(0.8,0.2))

train<- card\_data[sample,]

test<- card\_data[!sample,]

train.x<- data.matrix(train[-8])

train.y<- data.matrix(train[8])

test.x<- data.matrix(test[-8])

test.y<- data.matrix(test[8])

library(neuralnet)

#FITTING ANN WITH LOGISTIC ACTIVATION FUNCTION AND ONE LAYER WITH THREE NEURONS

ann.class<- neuralnet(as.factor(fraud) ~ distance\_from\_home +

distance\_from\_last\_transaction +

ratio\_to\_median\_purchase\_price + repeat\_retailer +

used\_chip + used\_pin\_number + online\_order, data=train,

hidden=3, act.fct="logistic")

#PLOTTING THE DIAGRAM

plot(ann.class)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA

pred.prob<- predict(ann.class, test.x)[,1]

match<- c()

for (i in 1:length(test.y)){

pred.y[i]<- ifelse(pred.prob[i]>0.5,1,0)

match[i]<- ifelse(test.y[i]==pred.y[i],1,0)

}

print(paste("accuracy=", round(mean(match), digits=4)))

####################################################################

#FITTING ANN WITH LOGISTIC ACTIVATION FUNCTION AND C(2,3) LAYERS

ann.class<- neuralnet(as.factor(fraud) ~ distance\_from\_home +

distance\_from\_last\_transaction +

ratio\_to\_median\_purchase\_price + repeat\_retailer +

used\_chip + used\_pin\_number + online\_order, data=train,

hidden=3, act.fct="logistic")

#PLOTTING THE DIAGRAM

plot(ann.class)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA

pred.prob<- predict(ann.class, test.x)[,1]

match<- c()

pred.y<- c()

for (i in 1:length(test.y)){

pred.y[i]<- ifelse(pred.prob[i]>0.5,1,0)

match[i]<- ifelse(test.y[i]==pred.y[i],1,0)

}

print(paste("accuracy=", round(mean(match), digits=4)))

####################################################################

#FITTING ANN WITH TANH ACTIVATION FUNCTION

ann.class<- neuralnet(as.factor(fraud) ~ distance\_from\_home +

distance\_from\_last\_transaction +

ratio\_to\_median\_purchase\_price + repeat\_retailer +

used\_chip + used\_pin\_number + online\_order, data=train,

hidden=2, act.fct="tanh")

#PLOTTING THE DIAGRAM

plot(ann.class)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA

pred.prob<- predict(ann.class, test.x)[,1]

match<- c()

pred.y<- c()

for (i in 1:length(test.y)){

pred.y[i]<- ifelse(pred.prob[i]>0.5,1,0)

match[i]<- ifelse(test.y[i]==pred.y[i],1,0)

}

print(paste("accuracy=", round(mean(match), digits=4)))

**Python Output**

Accuracy: 90.25 %

**Python Code**

import numpy

import pandas

from sklearn.model\_selection import train\_test\_split

card\_data=pandas.read\_csv(**r**'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data Mining\HW1STAT574S23\DATA SETS\card\_transdata.csv')

*#SCALING VARIABLES TO FALL IN [0,1]*

from sklearn import preprocessing

scaler=preprocessing.MinMaxScaler()

scaler\_fit=scaler.fit\_transform(card\_data)

scaled\_pneumonia\_data=pandas.DataFrame(scaler\_fit, columns=card\_data.columns)

X=card\_data.iloc[:,0:7].values

y=card\_data.iloc[:,7]

*#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS*

X\_train, X\_test, y\_train, y\_test=train\_test\_split(X, y, test\_size=0.20,

random\_state=505606)

*#FITTING AN ARTIFICIAL NEURAL NETWORK*

import keras

from keras.models import Sequential

from keras.layers import Dense

biclassifier=Sequential()

*#Defining the input layer and first hidden layer*

biclassifier.add(Dense(units=3, activation='sigmoid'))

*#Defining the output neuron*

biclassifier.add(Dense(1))

*#Compiling the model*

biclassifier.compile(loss='binary\_crossentropy')

*#Fitting the ANN to the training set*

biclassifier.fit(X\_train, y\_train)

*#COMPUTING PREDICTION ACCURACY FOR TESTING DATA*

y\_pred=numpy.round(biclassifier.predict(X\_test),0) *#predicted probability of 1*

from sklearn import metrics

print('Accuracy:', round(metrics.accuracy\_score(y\_test, y\_pred)\*100, 2),'%')

Problem 5. Use the data in the file “concussions\_data.csv” to fit an artificial neural network. Compute prediction accuracy.

**R Output**

0.8913043

**R Code**

# Data

concuss\_data <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/HW1STAT574S23/DATA SETS/concussions\_data.csv")

View(concuss\_data)

# convert character variables to numeric

concuss\_data$position <-

ifelse(concuss\_data$position=='Offensive Lineman',1,

ifelse(concuss\_data$position=='Cornerback',2,

ifelse(concuss\_data$position=='Running Back',3,

ifelse(concuss\_data$position=='Quarterback',4,5))))

concuss\_data$concussion <-

ifelse(concuss\_data$concussion=='mild',1,

ifelse(concuss\_data$concussion=='moderate',2,3))

#SCALING VARIABLES TO FALL IN [0,1]

library(dplyr)

scale01 <- function(x){

(x-min(x))/(max(x)-min(x))

}

concuss\_data<- concuss\_data %>% mutate\_all(scale01)

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS

set.seed(76309)

sample <- sample(c(TRUE, FALSE), nrow(concuss\_data), replace=TRUE, prob=c(0.8,0.2))

train<- concuss\_data[sample,]

test<- concuss\_data[!sample,]

train.x<- data.matrix(train[-5])

train.y<- data.matrix(train[5])

test.x<- data.matrix(test[-5])

test.y<- data.matrix(test[5])

#FITTING ANN WITH LOGISTIC ACTIVATION FUNCTION

library(neuralnet)

set.seed(38759)

ann.mclass<- neuralnet(as.factor(concussion) ~ age + nyearsplaying +

position + prevconc, data=train,

hidden=3, act.fct="logistic")

#PLOTTING THE DIAGRAM

plot(ann.mclass)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA

pred.prob <- predict(ann.mclass, test.x)

pred.prob<- as.data.frame(pred.prob)

colnames(pred.prob)<- c(0, 0.5, 1)

pred.class<- apply(pred.prob, 1, function(x) colnames(pred.prob)[which.max(x)])

match<- c()

for (i in 1:length(test.y)) {

match[i]<- ifelse(pred.class[i]==as.character(test.y[i]),1,0)

}

print(accuracy<- mean(match))

**Python Output**

Accuracy: 0.2495

**Python Code**

*# Problem 5 - ANN multinomial*

*# compute prediction accuracy*

import numpy

import pandas

from sklearn.model\_selection import train\_test\_split

from statistics import mean

concussion\_data=pandas.read\_csv(**r**'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data Mining\HW1STAT574S23\DATA SETS\concussions\_data.csv')

code\_position={'Offensive Lineman': 1, 'Cornerback': 2, 'Running Back': 3,'Wide Receiver': 4,

'Quarterback': 5}

code\_concussion={'mild': 1, 'moderate': 2, 'severe': 3}

concussion\_data['position']=concussion\_data['position'].map(code\_position)

concussion\_data['concussion']=concussion\_data['concussion'].map(code\_concussion)

*#SCALING VARIABLES TO FALL IN [0,1]*

from sklearn import preprocessing

scaler=preprocessing.MinMaxScaler()

scaler\_fit=scaler.fit\_transform(concussion\_data)

scaled\_movie\_data=pandas.DataFrame(scaler\_fit, columns=concussion\_data.columns)

X=concussion\_data.iloc[:,0:4]

y=concussion\_data.iloc[:,4:8]

*#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS*

X\_train, X\_test, y\_train, y\_test=train\_test\_split(X, y, test\_size=0.20, random\_state=116008)

y\_train=y\_train[:,1:4]

y\_true=y\_test[:,0]

y\_test=y\_test[:,1:4]

*#FITTING AN ARTIFICIAL NEURAL NETWORK*

import keras

from keras.models import Sequential

from keras.layers import Dense

import tensorflow

tensorflow.random.set\_seed(454545)

multiclassifier=Sequential()

*#Defining one hidden layer*

multiclassifier.add(Dense(units=2, activation='sigmoid'))

*#Defining the output neuron*

multiclassifier.add(Dense(units=3, activation='tanh'))

*#Compiling the model*

multiclassifier.compile(loss='categorical\_crossentropy')

*#Fitting the ANN to the training set*

multiclassifier.fit(X\_train, y\_train)

*#COMPUTING PREDICTION ACCURACY FOR TESTING DATA*

pred\_prob=pandas.DataFrame(multiclassifier.predict(X\_test))

y\_pred=0.25\*pred\_prob.idxmax(axis=1)

match=[]

for i in range(len(y\_pred)):

    if y\_pred[i]==y\_true[i]:

        match.append(1)

    else:

        match.append(0)

print('accuracy=', round(mean(match),4))