****

**BUDT758T  
  
DATA MINING AND PREDICTIVE ANALYTICS**

**Individual Assignment 4**

**NAME (in capitals): ROHIT KOTHAVADE**

* Please see the instructions at <https://docs.google.com/document/d/1uwOFS-LVKDBAzjEonmfggJMAmNWhWxAJRbL71nsVg4A/edit?usp=sharing> and submit on Canvas.
* Your submission should consist of this document (with the answers filled in the appropriate places).
* Please ensure that answers are appropriately numbered and clearly legible.
* In the space below please enter the following text and initial below: “I pledge on my honor that I have not given or received unauthorized assistance on this assignment.”

|  |
| --- |
| HONOR PLEDGE: “I pledge on my honor that I have not given or received unauthorized assistance on this assignment.”    YOUR INITIALS: RVK |

This is an individual assignment. Your submission must represent your own work.

The goal of this homework is to guide you through developing Neural Network models for (1) classification, and for   
(2) numerical prediction. You will change the key parameters (the number of hidden layers and nodes) and develop an understanding of how this affects results.

**Data**

The data set for this assignment is the Airlines data set used in Assignment 1.

The data in the accompanying file “Airline data 2.csv” (posted on Canvas) was assembled by Professor Robert Windle of the Smith School with assistance from Oliver Yao. The file contains information on 627 air routes in the United States. A route refers to a pair of airports. Note that some cities are served by more than one airport. In such cases, the airports are distinguished by their 3-letter code. The data was collected for the third quarter of 1996 (3Q96). The variables in the data set are:

1. S\_CODE: starting airport’s code
2. S\_CITY: starting city
3. E\_CODE: ending airport’s code
4. E\_CITY: ending city
5. COUPON: average number of coupons (a one-coupon flight is a non-stop flight, a two-coupon flight is a one stop flight, etc.) for that route
6. NEW: number of new carriers entering that route between Q3-96 and Q2-97
7. VACATION: whether a vacation route (Yes) or not (No); Florida and Las Vegas routes are generally considered vacation routes
8. SW: whether Southwest Airlines serves that route (Yes) or not (No)
9. HI: Herfindahl Index – airlines use this as a measure of market concentration (a larger value indicates greater concentration)
10. S\_INCOME: starting city’s average personal income
11. E\_INCOME: ending city’s average personal income
12. S\_POP: starting city’s population
13. E\_POP: ending city’s population
14. SLOT: whether either endpoint airport is slot controlled or not; this is a measure of airport congestion
15. GATE: whether either endpoint airport has gate constraints or not; this is another measure of airport congestion
16. DISTANCE: distance between two endpoint airports in miles
17. PAX: number of passengers on that route during period of data collection
18. FARE: average fare on that route

We will **not** use the first four attributes (S\_CODE, S\_CITY, E\_CODE, and E\_CITY) in our analysis.

1. Data Preparation
   1. Read the data set in *R*. We will only use the following variables in the analysis: SW, VACATION, SLOT, FARE, DISTANCE, HI, GATE, and PAX.
   2. For Neural Networks all explanatory variables have to be numerical. So convert all factors to binary dummy variables.

Ans –

str(airline$VACATION)

## Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 2 1 1 ...

airline$VACATION<-as.numeric(airline$VACATION)-1  
airline$VACATION\_Yes=as.numeric(airline$VACATION==1)  
airline$VACATION\_No=as.numeric(airline$VACATION==0)  
  
str(airline$SW)

## Factor w/ 2 levels "No","Yes": 2 1 1 2 2 2 1 2 2 2 ...

airline$SW<-as.numeric(airline$SW)-1  
airline$SW\_Yes=as.numeric(airline$SW==1)  
airline$SW\_No=as.numeric(airline$SW==0)  
  
str(airline$SLOT)

## Factor w/ 2 levels "Controlled","Free": 2 2 2 1 2 2 2 2 2 2 ...

airline$SLOT<-as.numeric(airline$SLOT)-1  
airline$SLOT\_Free=as.numeric(airline$SLOT==1)  
airline$SLOT\_Controlled=as.numeric(airline$SLOT==0)  
  
str(airline$GATE)

## Factor w/ 2 levels "Constrained",..: 2 2 2 2 2 2 2 2 2 2 ...

airline$GATE<-as.numeric(airline$GATE)-1  
airline$GATE\_Free=as.numeric(airline$GATE==1)  
airline$GATE\_Constrained=as.numeric(airline$GATE==0)

* 1. Rescale all numerical variables to lie in the 0-1.

Ans –

airline$COUPON = (airline$COUPON-min(airline$COUPON))/(max(airline$COUPON)-min(airline$COUPON))  
airline$NEW = (airline$NEW-min(airline$NEW))/(max(airline$NEW)-min(airline$NEW))  
airline$HI = (airline$HI-min(airline$HI))/(max(airline$HI)-min(airline$HI))  
airline$S\_INCOME = (airline$S\_INCOME-min(airline$S\_INCOME))/(max(airline$S\_INCOME)-min(airline$S\_INCOME))  
airline$E\_INCOME = (airline$E\_INCOME-min(airline$E\_INCOME))/(max(airline$E\_INCOME)-min(airline$E\_INCOME))  
airline$S\_POP = (airline$S\_POP-min(airline$S\_POP))/(max(airline$S\_POP)-min(airline$S\_POP))  
airline$E\_POP = (airline$E\_POP-min(airline$E\_POP))/(max(airline$E\_POP)-min(airline$E\_POP))  
airline$PAX = (airline$PAX-min(airline$PAX))/(max(airline$PAX)-min(airline$PAX))  
airline$DISTANCE = (airline$DISTANCE-min(airline$DISTANCE))/(max(airline$DISTANCE)-min(airline$DISTANCE))  
airline$FARE = (airline$FARE-min(airline$FARE))/(max(airline$FARE)-min(airline$FARE))

* 1. Delete all missing observations.

Ans –

airline<-airline[complete.cases(airline),]

* 1. Set the seed to 71923

Ans - set.seed(71923)

* 1. Randomly partition the data set into the *training* and *test* data sets. The proportion of observations in the training data set should be 60%. The remaining 40% of observations should be in the test data set.
* *Use the “sample” function to partition the data.*

Ans –

splitrule<-sample(x = nrow(airline), size = nrow(airline)\*0.6)  
df\_train<-airline[splitrule,]  
df\_test<-airline[-splitrule,]

1. You will first build a model to predict what routes Southwest will choose to enter. Run a logistic regression model of SW on all the other variables enumerated in 1(a) above. Use only the training data set for this.
   1. Present the output as **Exhibit A**.

Ans –

fit1<-glm(SW ~ VACATION + SLOT + HI + FARE + DISTANCE + PAX + GATE,data = df\_train, family = "binomial")  
summary(fit1)

##   
## Call:  
## glm(formula = SW ~ VACATION + SLOT + HI + FARE + DISTANCE + PAX +   
## GATE, family = "binomial", data = df\_train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.2446 -0.4661 -0.1044 0.4399 2.9297   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 1.3892 0.9068 1.532 0.125541   
## VACATION -1.5031 0.3802 -3.954 7.70e-05 \*\*\*  
## SLOT 0.8730 0.4745 1.840 0.065819 .   
## HI -0.7662 0.9811 -0.781 0.434815   
## FARE -17.2214 2.2699 -7.587 3.28e-14 \*\*\*  
## DISTANCE 4.7229 1.2679 3.725 0.000195 \*\*\*  
## PAX -1.3468 1.1309 -1.191 0.233680   
## GATE 1.2197 0.5551 2.197 0.027996 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 465.57 on 372 degrees of freedom  
## Residual deviance: 240.03 on 365 degrees of freedom  
## AIC: 256.03  
##   
## Number of Fisher Scoring iterations: 6

**Exhibit A**

* 1. Use a cutoff of 0.5 and compute the classification matrix. What are the error rate, the sensitivity and the specificity of the classifier? (Compute these for both the training and test data.)

Ans – **For training data set:**

cutoff=0.5  
pred\_train<-ifelse(predict(fit1,df\_train,type = "response") > cutoff,1,0)  
pred\_1 <-predict(fit1,df\_test,type="response")  
pred\_test<-ifelse(pred\_1>cutoff,1,0)  
confusion\_train<-table(df\_train$SW, pred\_train)  
rownames(confusion\_train)<-c("SW = No","SW=Yes")  
colnames(confusion\_train)<-c("SW = No","SW=Yes")  
confusion\_train

## pred\_train  
## SW = No SW=Yes  
## SW = No 234 21  
## SW=Yes 26 92

(sensitivity\_tr1<-sum(pred\_train==1 & df\_train$SW\_Yes==1)/sum(df\_train$SW\_Yes==1))

## [1] 0.779661

(specificity\_tr1<-sum(pred\_train==0 & df\_train$SW\_Yes==0)/sum(df\_train$SW\_Yes==0))

## [1] 0.9176471

(accuracy\_tr1<-sum(df\_train$SW\_Yes==pred\_train)/nrow(df\_train))

## [1] 0.8739946

(error.rate\_tr1<-1-accuracy\_tr1)

## [1] 0.1260054

Sensitivity = 0.7797

Specificity = 0.9176

Error rate = 0.1260

**For test data set:**

confusion\_test<-table(df\_test$SW, pred\_test)  
rownames(confusion\_test)<-c("SW = No","SW=Yes")  
colnames(confusion\_test)<-c("SW = No","SW=Yes")  
confusion\_test

## pred\_test  
## SW = No SW=Yes  
## SW = No 157 18  
## SW=Yes 20 55

(sensitivity\_te1<-sum(pred\_test==1 & df\_test$SW\_Yes==1)/sum(df\_test$SW\_Yes==1))

## [1] 0.7333333

(specificity\_te1<-sum(pred\_test==0 & df\_test$SW\_Yes==0)/sum(df\_test$SW\_Yes==0))

## [1] 0.8971429

(accuracy\_te1<-sum(df\_test$SW\_Yes==pred\_test)/nrow(df\_test))

## [1] 0.848

(error.rate\_te1<-1-accuracy\_te1)

## [1] 0.152

Sensitivity = 0.7333

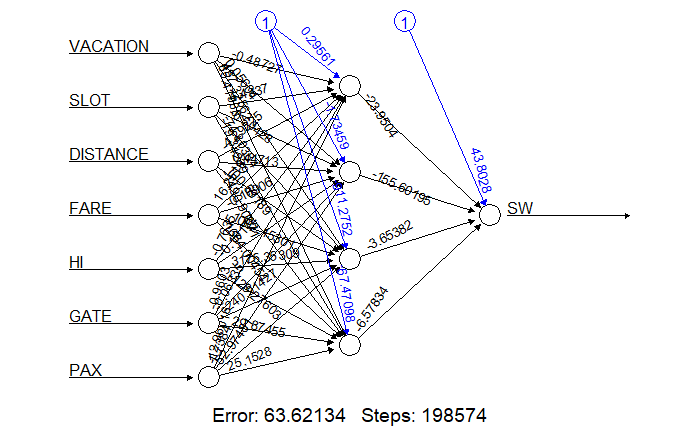
Specificity = 0.8971

Error rate = 0.152

1. Build a Neural Network with one hidden layer, and 4 nodes in that layer. Use the default settings of the *neuralnet* function except that, since this is a classification problem, set *err.fct = “ce”* and *linear.output = FALSE*.

* *Set the seed to 13 (there is randomness in the choice of initial weights)*
* *Use the following order of variables: SW~VACATION+SLOT+DISTANCE+FARE+HI+GATE+PAX*
  1. Plot the net and attach as **Exhibit B**.

Ans –



**Exhibit B**

* 1. Compute the confusion matrix for both the training and test data sets.

Ans – **For training data set:**

prob\_train<-compute(nn, df\_train)$net.result  
predict\_tr <- ifelse(prob\_train>0.5,1,0)  
confusion\_tr<-table(df\_train$SW,predict\_tr)  
confusion\_tr

## predict\_tr  
## 0 1  
## 0 244 11  
## 1 13 105

**For test data set:**

prob\_test<-compute(nn, df\_test)$net.result  
predict\_te <- ifelse(prob\_test>0.5,1,0)  
confusion\_te<-table(df\_test$SW,predict\_te)  
confusion\_te

## predict\_te  
## 0 1  
## 0 152 23  
## 1 19 56

* 1. Compute the error rate, sensitivity, and specificity in each case.

Ans – **For training data set:**

(sensitivity.tr2<-confusion\_tr[2,2]/(sum(confusion\_tr[2,1],confusion\_tr[2,2])))

## [1] 0.8898305

(specificity.tr2<-confusion\_tr[1,1]/(sum(confusion\_tr[1,2],confusion\_tr[1,1])))

## [1] 0.9568627

(accuracy\_tr2<-(sum(confusion\_tr[2,2],confusion\_tr[1,1]))/nrow(df\_train))

## [1] 0.9356568

(error.rate\_tr2<-1-accuracy\_tr2)

## [1] 0.06434316

Sensitivity = 0.8898

Specificity = 0.9568

Error rate = 0.0643

**For test data set:**

(sensitivity.te2<-confusion\_te[2,2]/(sum(confusion\_te[2,1],confusion\_te[2,2])))

## [1] 0.7466667

(specificity.te2<-confusion\_te[1,1]/(sum(confusion\_te[1,2],confusion\_te[1,1])))

## [1] 0.8685714

(accuracy\_te2<-(sum(confusion\_te[2,2],confusion\_te[1,1]))/nrow(df\_test))

## [1] 0.832

(error.rate\_te2<-1-accuracy\_te2)

## [1] 0.168

Sensitivity = 0.7467

Specificity = 0.8686

Error rate = 0.168

* 1. Compare your results with the logistic regression classification model in (2).

Ans – For training data:

The values for sensitivity and error rate are better in the Neural Net, but the specificity is higher in Neural Net as compared to the Logistic model.

For test data:

The values for sensitivity and specificity are better in the Neural Net, but the error rate is a bit higher as compared to the Logistic model.

The values are as follows:

**Logistic model**

Training data:

Sensitivity = 0.7797

Specificity = 0.9176

Error rate = 0.1260

Test data:

Sensitivity = 0.7333

Specificity = 0.8971

Error rate = 0.152

**Neural net**

Training data:

Sensitivity = 0.8898

Specificity = 0.9568

Error rate = 0.0643

Test data:  
Sensitivity = 0.7467

Specificity = 0.8686

Error rate = 0.168

1. In this section of the assignment you will explore changing the specification of the hidden layer and determine how this affects the performance of your model.
   1. Compare models with one hidden layer with between 0 to 7 nodes in that layer. Use the error rate as the basis for your comparison. Compare both the training and test error rates.

Ans – Considering Training error rates:

The model with 7 nodes in the hidden layer turns to converge with the lowest error rate of 0.01608. Models with 0,1 and 2 nodes in the hidden layer give pretty much same error rate i.e. 0.1260, 0.1286 and 0.1206 respectively.

Considering Test error rates:

The error rate 0.144 is lowest for the model with 4 nodes in the hidden layer. The models with 0,1 and 2 nodes in the hidden layer have similar error rate i.e. 0.152, 0.152 and 0.156 respectively.

* 1. Now compare models with two hidden layers. Set the number of nodes in the first layer at 4, and let the number of nodes in the second layer vary from 1 to 4. Use the error rate as the basis for your comparison. Compare both the training and test error rates.

Ans – Considering Training error rates:

The model with the combination (4,4) provides the lowest error rate of 0.0375.

Considering Test error rates:

The model with combinations (4,1) and (4,2) have the lowest error rates of 0.144.

* 1. Which, among the models examined in 4(a) and 4(b), is the best model? Why?

Ans – According to me, the 4(b) model is best among the two. This is because even though the error in 1st model gradually decreases and becomes very low as compared to that of the 2nd model, if we look at the error rates calculated, they are higher for model 4(a) as compared to 4(b) for both training and test data sets.

* 1. What do you observe from changing the layers and number of nodes? Summarize your findings.

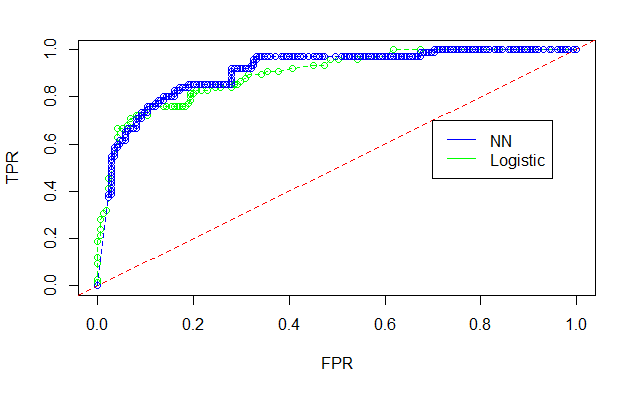
Ans – We can see from both the above models that as the number of nodes in a hidden layer increases, the error gradually decreases along with an increase in time required for the algorithm to converge. Also, the error rate decreases for training data but increases for the test data set.

However, when we increase the number of hidden layers with lesser number of nodes in each as compared to above, the error rates obtained are less as compared to model with single hidden layer.

To conclude, as the number of hidden layers increase the error rate decreases but the converging time increases. Sometimes, a single hidden layer also works but it depends on the number of nodes in the input and output layers.

* 1. Using the model that you judge to be the best, plot the ROC curve. On the same graph, plot the ROC curve for the logistic regression model in (2). Use the test data for this. Comment.

Ans – Model with combination (4,1) has the least error rate. The neural net model with combination (4,1) performs better than the logistic regression model when the TPR is above 0.8. Above 0.8 TPR, both the models are performing similarly.



1. The Neural Network algorithm can also be used with continuous dependent variables. In this section, you will build a model to predict FARE based on the other variables listed in (1) above. Include SW, but this time as a predictor.

* *ReSet the seed to 13 (there is randomness in the choice of initial weights)*
* *Use the following order of variables: FARE~ SW+VACATION+SLOT+DISTANCE+HI+GATE+PAX*
  1. As benchmark, run a multiple regression model for FARE, and compute the RMSE for both the training and test data sets.

Ans – RMSE for training – 0.1024

RMSE for test – 0.1038

fit2<-lm(FARE ~ SW + VACATION + SLOT + DISTANCE + HI + GATE + PAX,data = df\_train)  
summary(fit2)

##   
## Call:  
## lm(formula = FARE ~ SW + VACATION + SLOT + DISTANCE + HI + GATE +   
## PAX, data = df\_train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.266461 -0.061140 -0.005743 0.069184 0.309288   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.26962 0.02350 11.472 < 2e-16 \*\*\*  
## SW -0.14252 0.01333 -10.689 < 2e-16 \*\*\*  
## VACATION -0.12941 0.01302 -9.942 < 2e-16 \*\*\*  
## SLOT -0.06909 0.01370 -5.042 7.28e-07 \*\*\*  
## DISTANCE 0.59238 0.02394 24.746 < 2e-16 \*\*\*  
## HI 0.19101 0.03030 6.304 8.39e-10 \*\*\*  
## GATE -0.08065 0.01453 -5.551 5.46e-08 \*\*\*  
## PAX -0.06856 0.03203 -2.140 0.033 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1035 on 365 degrees of freedom  
## Multiple R-squared: 0.7806, Adjusted R-squared: 0.7764   
## F-statistic: 185.5 on 7 and 365 DF, p-value: < 2.2e-16

predicttrain <- predict(fit2, newdata=df\_train)  
rmse(df\_train$FARE,predicttrain)

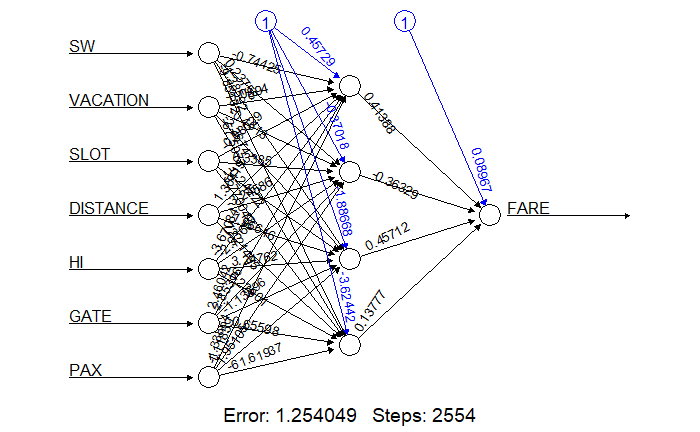
## [1] 0.1024064

predicttest <- predict(fit2, newdata=df\_test)  
rmse(df\_test$FARE,predicttest)

## [1] 0.1038087

* 1. Now build a Neural Network with one hidden layer and 4 nodes in that layer. Set *err.fct = “sse”* (this is the default, so really only need to ensure that you do not set this to anything else) and *linear.output = TRUE*. Plot the net and attach as **Exhibit C**.

Ans –



**Exhibit C**

* 1. Compute the RMSE for both the training and test data set, and compare with your regression model in 5(a).

Ans – RMSE for training – 0.082

RMSE for test – 0.0943

predprobnn\_train <- compute(nn\_5, df\_train[-18])$net.result  
rmse(df\_train$FARE,predprobnn\_train)

## [1] 0.08200076

predprobnn\_test <- compute(nn\_5, df\_test)$net.result  
rmse(df\_test$FARE,predprobnn\_test)

## [1] 0.09435702

The RMSE for both the training and test data sets decrease when we use Neural Net and thus gives a better model as compared to multiple regression.

* 1. Change the number of nodes as described in 4(a) and 4(b). Only, this time, use the RMSE as the basis for your comparison (for both the training and test error rates). Describe what you find.

Ans – The error rate for training data sets is higher in model with 1 hidden layer than the model with 2 hidden layers. As the no of nodes increase the error decreases in both the models. The lowest RMSE is for the model with combination (4,2).

The error rate for test data is lower in general for model with 2 hidden layers. But the lowest test RMSE is for model with single hidden layer with 5 nodes = 0.0915

Single Hidden layer with nodes 0 to 7:

Training data RMSE -

print(errorRate\_Train\_nn)

## [1] 0.10240661 0.09912663 0.08843405 0.08392535 0.07974218 0.08202100  
## [7] 0.08342028 0.0824306

Test data RMSE –

print(errorRate\_Test\_nn)

## [1] 0.10380459 0.10145675 0.09628636 0.09456294 0.09564183 0.09147345  
## [7] 0.09424004 0.09427222

Hidden Layer (4,x,where x=1 to 4)

Training data RMSE –

print(errorRate\_Train\_nn1)

## [1] 0.08261156 0.07892992 0.08067247 0.08197283

Test data RMSE –

print(errorRate\_Test\_nn1)

## [1] 0.09163609 0.09518652 0.09722189 0.09251383