MIS 432 – Project 1 – Spring 2019

Rachelle Gambol

George Mason University

**Preprocessing**

To build the models, the data had to be preprocessed. A copy of the Takyo dataset was copied and pasted into blank Excel worksheet to scale the variables so that it would not alter the csv file. The Purchase\_Yes variable was created using the function IF(I2=1,1,0) and the Purchase\_No variable using the function IF(I2=0,1,0). Next, the numerical variables last\_update\_days\_ago, 1st\_update\_days\_ago, and Freq were scaled using the 𝑓(𝑥)= (𝑥−𝑚𝑖𝑛)/(𝑚𝑎𝑥−𝑚𝑖𝑛) formula. The new scaled variables were renamed: last\_update\_S, first\_update\_days\_ago\_S, and Freq\_S. After the variables were scaled and created, they were pasted into the Takyo.csv file.

**Constructing the Models**

The first step is to read in our Tayko dataset. The dataset was renamed “catalog” to match the business scenario. The next step is to set the seed to two and to partition our dataset into a 60% training dataset and 40% validation dataset. After partitioning the dataset, the three models had to be constructed based on the specifications provided. Model 1 was the first model that was constructed, and it had one hidden layer with four nodes. The next model was model 2, and it also had one hidden layer. Unlike model 1, model 2 only had three nodes. Model 3 was the last model made and it had two hidden layers with two nodes in each layer. When the models were coded, the order of the binary and numerical variables were in the same order as they were in the catalog dataset.

**Creating New Cases**

For additional analysis, the models were used to predict new cases. Two new datasets called TaykoNC and TaykoNC2 were made. For TakyoNC, dataset represented a scenario where the customer was a male, the address was a US address, the address was a residential one, and the customer placed at least one order on the web. As for the numerical variables, they were scaled so that the variables represented the last update was 1234 days ago, the first update was 365 days, and there were three transactions in the last year. The new case was performed on each of the models to see if the customer would most likely purchase an item from the catalog.

After creating the first new case, the models were tested to see if they would accurately predict the results of a scenario where a customer was not a regular shopper or had minimum interaction with Tayko store. The second new case, TaykoNC2, would predict whether a female customer whose address was not a US address and was not residential would buy an item. The customer’s last update was 100 days ago, the first update was 100 days ago, and there were zero transactions from the customer in the last year.

The order that was performed for each model was: create the neural network according to the specifications, create a Confusion Matrix for each model, and predict a new case using the TaykoNC and TaykoNC2 dataset. The results were rounded to the third decimal place.

**Results**

**Model 1**

The first model depicted a neural network with one hidden layer and four nodes. After performing the Confusion Matrix on this model, the model received an accuracy of 74.625% and an error of 25.375%. The results of predicting the new case TaykoNC showed that there is a 94.609% chance that a customer will buy an item and a 5.417% chance that they will not. For the next new case, the model predicted that there is a 100% guarantee a person who does not have a US address would not buy from the catalog. This model may not be accurate because there could be a small chance that a shopper outside of the United States with no purchases may want to try an item from the company’s catalog.

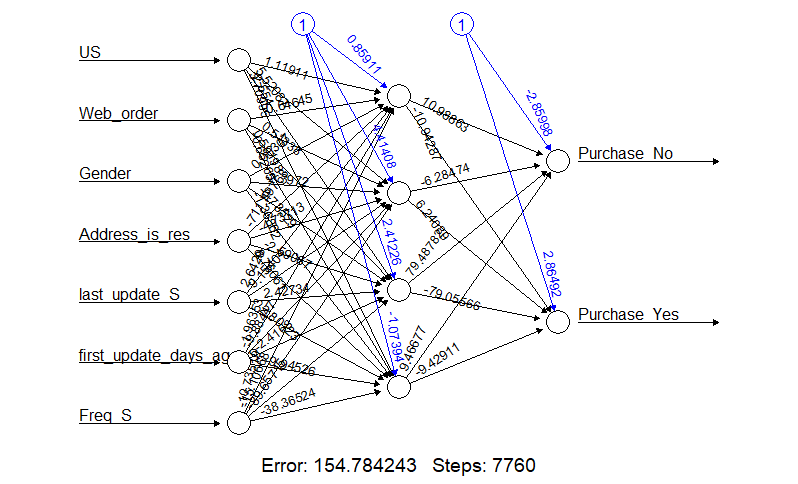


Figure 1: Model 1 NN

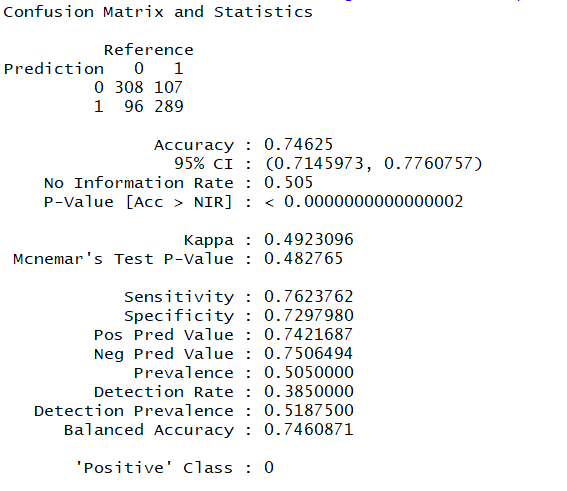


Figure 2: Confusion Matrix 1



Figure 3: New Case NN.1

**Model 2**

The next model, model 2, has a neural network with one hidden layer and three nodes. The confusion matrix showed that our model had an accuracy of 75.5% and an error of 24.5%. This model is more accurate than model 1 by 0.875%. Model 2 also predicted that a customer would most likely purchase an item from the catalog if they had frequent purchases and were an existing customer. According to our prediction, there is around a 3.534% chance that the customer will not buy an item and a 96.519% chance that they will. The predictions of the second new case resulted in 99.972% of a chance not buying a catalog item and a 0.02738% chance of an item being purchased. Like the first model, model 2 predicts a person who is not a frequent buyer would not buy an item. However, unlike model 1, this model has taken into consideration the small subset of people who would purchase an item.

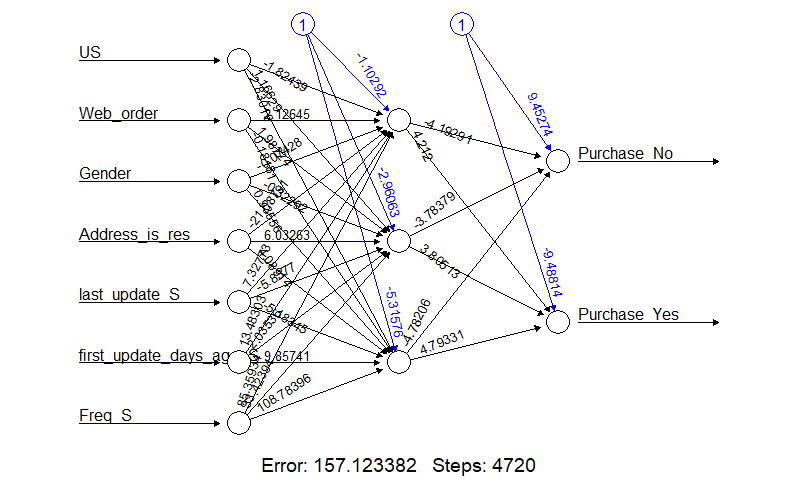


Figure 4: Model 2 NN

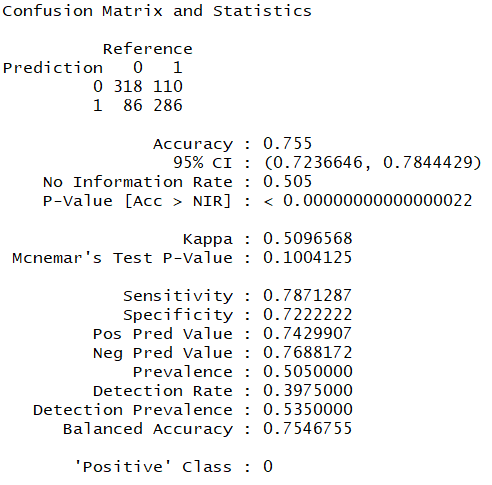


Figure 5: Confusion Matrix 2

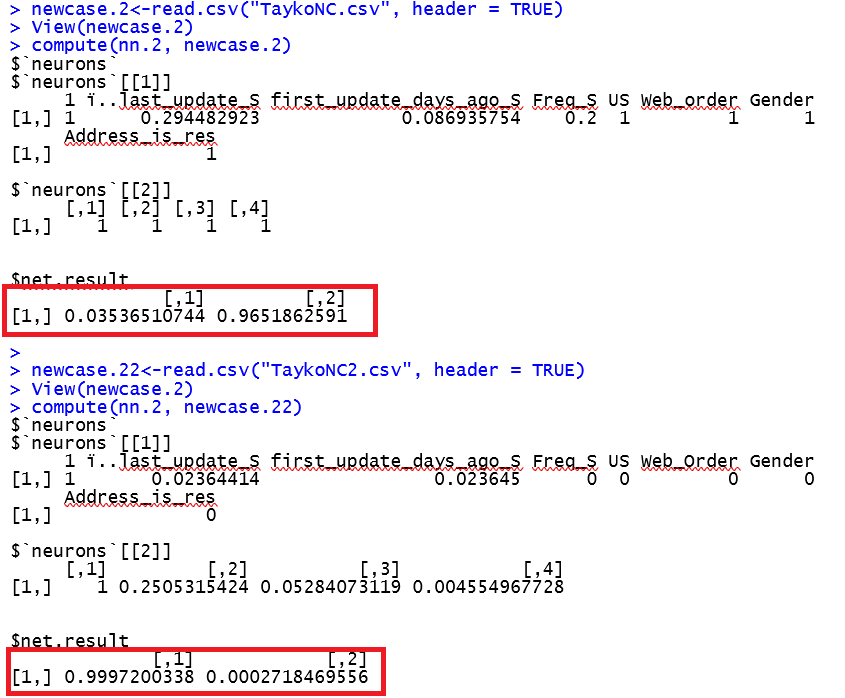


Figure 6: New Case NN.2

**Model 3**

The last model was model 3 and it had two hidden layers with two nodes each. The Confusion Matrix produced showed that the model had an overall accuracy 74.875% and an error of 25.125%. This model had a higher accuracy than model 1, but still did not perform as well as model 2. Model 3 predicted that there is a 96.496% chance of a customer making a purchase and a 3.512% chance that they will not. For the second new case, the model predicted that a customer will most likely not purchase an item. The net result of the new case resulted in a 99.633% for item not being purchased and a 0.365% chance of an item being purchased.

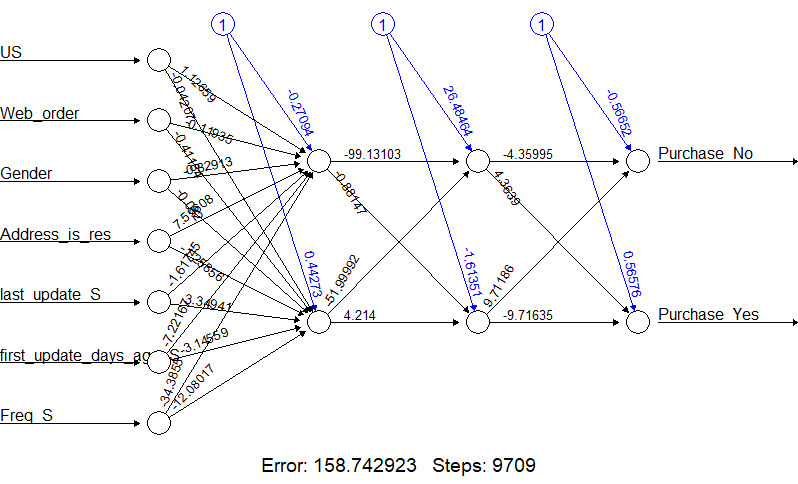


Figure 7: Model 3 NN

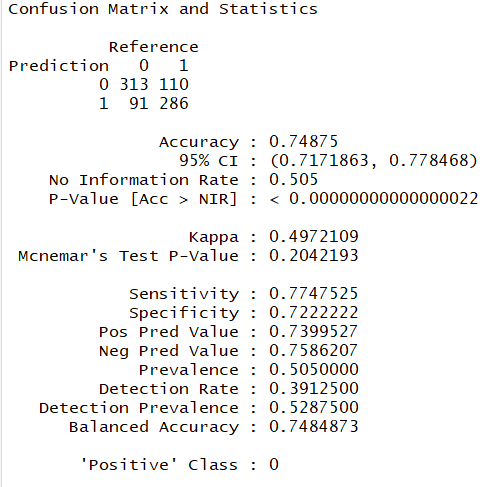


Figure 8: Confusion Matrix 3

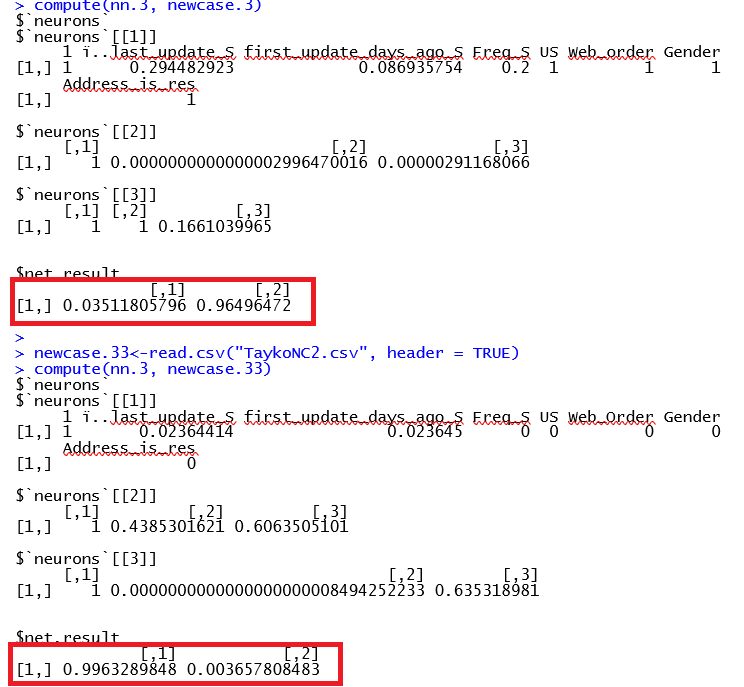


Figure 9: New Case NN.3

**Conclusion**

Out of all the models produced, the best model for Tayko would be model 2. This model would be the best model for Tayko to use because it had the highest accuracy of 75.5% and lowest error 24.5% out of the three models. Model 1 performed the best because it had the right number of nodes and the one ideal hidden layer. Model 3 had the least number of nodes, but it had an additional hidden layer which may have caused it to have a higher level of error than model 2. Model 1 had one hidden layer like model 3, but it had four nodes instead of three. This model had the most error out of the three. By adding one more node, the error increased. Both model 2 and model 3 may have had more error due to overfitting of the dataset.

After concluding that model 1 was the best model, two additional neural networks were tested and Confusion Matrices for each model were constructed to see if changing the number of nodes and hidden layers would affect the accuracy and error of the model. Model 4 composed of one hidden layer with two nodes. Model 5 was the same model as model 3 with one additional hidden layer with one node. Model 4 had was 75.25% accurate and model 5 was 75.75% accurate. By looking at the calculate accuracies for these models, we can conclude that removing one more node would result in a less accurate model and adding a layer could potentially increase the accuracy of the model. Usually, adding more hidden layers would result in overfitting. Maybe this particular dataset was capable of handling additional layers and nodes based on the number of observations the Tayko dataset had. After the new models were constructed, we can conclude that model 2 is ideal since removing nodes would decrease its accuracy just like how increasing its nodes did. If we were to enhance the model, adding layers would help increase its performance.

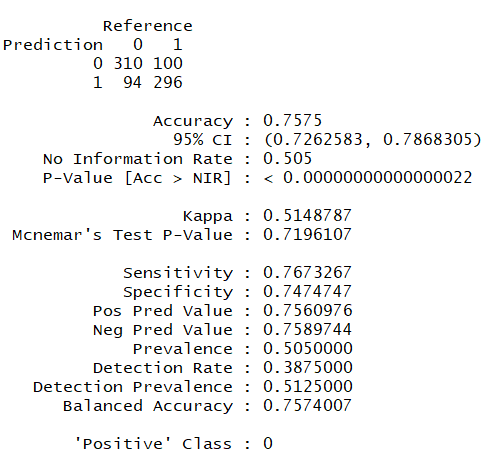
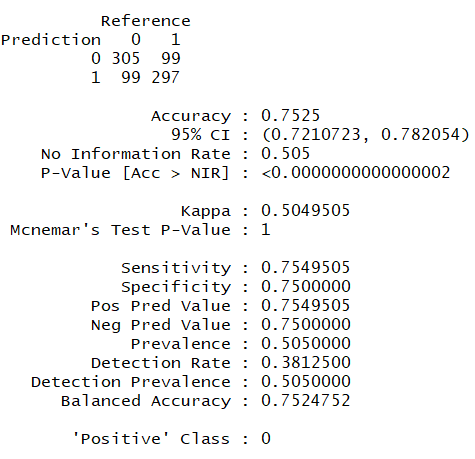


Figure 10: Model 4 Confusion Matrix Figure 11: Model 5 Confusion Matrix

After running new cases on each of the models, model 2 predicted that the customer will most likely buy from the company’s catalog. The other new case was able to reasonably predict that a customer who is not in the US or made frequent purchases would not buy anything. This is an accurate prediction for the scenario because a person who is not a regular customer would not buy an item. Overall model 2 would be the most beneficial for Tayko since the model was able to accurately predict potential customers with certain characteristics such as frequent purchases or address type. Model 2 is the best performing model out of the three. Not only did the model have the highest accuracy, but it had low error and was able to make reasonable predictions for different business problems.

Appendix

|  |
| --- |
| > setwd("~/MIS432 Adv Data Mining")  > catalog.df<-read.csv("Tayko.csv", header=TRUE)  > View(catalog.df)  >  > #Uploading Packages  > library(neuralnet)  Warning message:  package ‘neuralnet’ was built under R version 3.5.2  > library(caret, e1071)  Loading required package: lattice  Loading required package: ggplot2  Need help? Try Stackoverflow: https://stackoverflow.com/tags/ggplot2.  Warning messages:  1: package ‘caret’ was built under R version 3.5.2  2: package ‘ggplot2’ was built under R version 3.5.2  >  > #Partitioning the Dataset  > set.seed(2)  > train.rows.c<-sample(rownames(catalog.df),  + dim(catalog.df)[1]\*.60)  > train.data.c<-catalog.df[train.rows.c,]  > valid.rows.c<-setdiff(rownames(catalog.df), train.rows.c)  > valid.data.c<-catalog.df[valid.rows.c,]  >  > #Model 1 with One Hidden Layer with Four Nodes  > nn.1<-neuralnet(Purchase\_No + Purchase\_Yes ~  + US + Web\_order+ Gender + Address\_is\_res +  + last\_update\_S + first\_update\_days\_ago\_S + Freq\_S,  + data=train.data.c, linear.output=FALSE, threshold=0.05, hidden = 4)  >  > plot(nn.1, rep = "best")  >  > valid.pred.c = compute(nn.1, valid.data.c [ , c(2,21,22,23, 28:30)])  > valid.class.c = apply(valid.pred.c$net.result, 1,which.max)-1  > confusionMatrix(as.factor(valid.class.c),  + as.factor(catalog.df[valid.rows.c, ]$Purchase\_Yes))  Confusion Matrix and Statistics  Reference  Prediction 0 1  0 308 107  1 96 289    Accuracy : 0.74625  95% CI : (0.7145973, 0.7760757)  No Information Rate : 0.505  P-Value [Acc > NIR] : < 0.0000000000000002    Kappa : 0.4923096  Mcnemar's Test P-Value : 0.482765    Sensitivity : 0.7623762  Specificity : 0.7297980  Pos Pred Value : 0.7421687  Neg Pred Value : 0.7506494  Prevalence : 0.5050000  Detection Rate : 0.3850000  Detection Prevalence : 0.5187500  Balanced Accuracy : 0.7460871    'Positive' Class : 0    >  >  >  >  > #Model 2 with One Hidden Layer with Three Nodes  > nn.2<-neuralnet(Purchase\_No + Purchase\_Yes ~  + US + Web\_order+ Gender + Address\_is\_res +  + last\_update\_S + first\_update\_days\_ago\_S + Freq\_S,  + data=train.data.c,linear.output=F, threshold=0.05, hidden = 3)  >  > plot(nn.2, rep = "best")  >  > valid.pred.c = compute(nn.2, valid.data.c [ , c(2,21,22,23, 28:30)])  > valid.class.c = apply(valid.pred.c$net.result, 1,which.max)-1  > confusionMatrix(as.factor(valid.class.c),  + as.factor(catalog.df[valid.rows.c, ]$Purchase\_Yes))  Confusion Matrix and Statistics  Reference  Prediction 0 1  0 318 110  1 86 286    Accuracy : 0.755  95% CI : (0.7236646, 0.7844429)  No Information Rate : 0.505  P-Value [Acc > NIR] : < 0.00000000000000022    Kappa : 0.5096568  Mcnemar's Test P-Value : 0.1004125    Sensitivity : 0.7871287  Specificity : 0.7222222  Pos Pred Value : 0.7429907  Neg Pred Value : 0.7688172  Prevalence : 0.5050000  Detection Rate : 0.3975000  Detection Prevalence : 0.5350000  Balanced Accuracy : 0.7546755    'Positive' Class : 0    >  >  >  > #Model 3 with Two Hidden Layers with Two Nodes Each  > nn.3<-neuralnet(Purchase\_No + Purchase\_Yes ~  + US + Web\_order + Gender + Address\_is\_res +  + last\_update\_S + first\_update\_days\_ago\_S + Freq\_S,  + data=train.data.c, linear.output=FALSE,  + threshold=0.05, hidden = c(2,2))  >  > plot(nn.3,rep="best")  >  > valid.pred.c = compute(nn.3, valid.data.c [ , c(2,21,22,23, 28:30)])  > valid.class.c = apply(valid.pred.c$net.result, 1,which.max)-1  > confusionMatrix(as.factor(valid.class.c),  + as.factor(catalog.df[valid.rows.c, ]$Purchase\_Yes))  Confusion Matrix and Statistics  Reference  Prediction 0 1  0 313 110  1 91 286    Accuracy : 0.74875  95% CI : (0.7171863, 0.778468)  No Information Rate : 0.505  P-Value [Acc > NIR] : < 0.00000000000000022    Kappa : 0.4972109  Mcnemar's Test P-Value : 0.2042193    Sensitivity : 0.7747525  Specificity : 0.7222222  Pos Pred Value : 0.7399527  Neg Pred Value : 0.7586207  Prevalence : 0.5050000  Detection Rate : 0.3912500  Detection Prevalence : 0.5287500  Balanced Accuracy : 0.7484873    'Positive' Class : 0    >  >  > #Predicting Cases  > newcase.1<-read.csv("TaykoNC.csv", header = TRUE)  > View(newcase.1)  > compute(nn.1, newcase.1)  $`neurons`  $`neurons`[[1]]  1 ï..last\_update\_S first\_update\_days\_ago\_S Freq\_S US Web\_order Gender  [1,] 1 0.294482923 0.086935754 0.2 1 1 1  Address\_is\_res  [1,] 1  $`neurons`[[2]]  [,1] [,2] [,3] [,4]  [1,] 1 0 0.0000003154836736 0.0000000000000000000000000000000000000004183580403  [,5]  [1,] 0.00000000000000008031714499  $net.result  [,1] [,2]  [1,] 0.05416761107 0.9460850304  >  > newcase.11<-read.csv("TaykoNC2.csv", header = TRUE)  > compute(nn.1, newcase.11)  $`neurons`  $`neurons`[[1]]  1 ï..last\_update\_S first\_update\_days\_ago\_S Freq\_S US Web\_Order Gender  [1,] 1 0.02364414 0.023645 0 0 0 0  Address\_is\_res  [1,] 0  $`neurons`[[2]]  [,1] [,2] [,3] [,4] [,5]  [1,] 1 0.7048040103 0.9896129963 0.9141876828 0.2686165667  $net.result  [,1] [,2]  [1,] 1 0.00000000000000000000000000000001229495199  >  > newcase.2<-read.csv("TaykoNC.csv", header = TRUE)  > View(newcase.2)  > compute(nn.2, newcase.2)  $`neurons`  $`neurons`[[1]]  1 ï..last\_update\_S first\_update\_days\_ago\_S Freq\_S US Web\_order Gender  [1,] 1 0.294482923 0.086935754 0.2 1 1 1  Address\_is\_res  [1,] 1  $`neurons`[[2]]  [,1] [,2] [,3] [,4]  [1,] 1 1 1 1  $net.result  [,1] [,2]  [1,] 0.03536510744 0.9651862591  >  > newcase.22<-read.csv("TaykoNC2.csv", header = TRUE)  > View(newcase.2)  > compute(nn.2, newcase.22)  $`neurons`  $`neurons`[[1]]  1 ï..last\_update\_S first\_update\_days\_ago\_S Freq\_S US Web\_Order Gender  [1,] 1 0.02364414 0.023645 0 0 0 0  Address\_is\_res  [1,] 0  $`neurons`[[2]]  [,1] [,2] [,3] [,4]  [1,] 1 0.2505315424 0.05284073119 0.004554967728  $net.result  [,1] [,2]  [1,] 0.9997200338 0.0002718469556  >  > newcase.3<-read.csv("TaykoNC.csv", header = TRUE)  > compute(nn.3, newcase.3)  $`neurons`  $`neurons`[[1]]  1 ï..last\_update\_S first\_update\_days\_ago\_S Freq\_S US Web\_order Gender  [1,] 1 0.294482923 0.086935754 0.2 1 1 1  Address\_is\_res  [1,] 1  $`neurons`[[2]]  [,1] [,2] [,3]  [1,] 1 0.0000000000000002996470016 0.00000291168066  $`neurons`[[3]]  [,1] [,2] [,3]  [1,] 1 1 0.1661039965  $net.result  [,1] [,2]  [1,] 0.03511805796 0.96496472  >  > newcase.33<-read.csv("TaykoNC2.csv", header = TRUE)  > compute(nn.3, newcase.33)  $`neurons`  $`neurons`[[1]]  1 ï..last\_update\_S first\_update\_days\_ago\_S Freq\_S US Web\_Order Gender  [1,] 1 0.02364414 0.023645 0 0 0 0  Address\_is\_res  [1,] 0  $`neurons`[[2]]  [,1] [,2] [,3]  [1,] 1 0.4385301621 0.6063505101  $`neurons`[[3]]  [,1] [,2] [,3]  [1,] 1 0.0000000000000000000008494252233 0.635318981  $net.result  [,1] [,2]  [1,] 0.9963289848 0.003657808483  >  > #Additional Models  > #Model 4 with One Hidden Layer with Two Nodes  > nn.4<-neuralnet(Purchase\_No + Purchase\_Yes ~  + US + Web\_order+ Gender + Address\_is\_res +  + last\_update\_S + first\_update\_days\_ago\_S + Freq\_S,  + data=train.data.c, linear.output=FALSE, threshold=0.05,  + hidden = 2)  >  >  > valid.pred.c = compute(nn.4, valid.data.c [ , c(2,21,22,23, 28:30)])  > valid.class.c = apply(valid.pred.c$net.result, 1,which.max)-1  > confusionMatrix(as.factor(valid.class.c),  + as.factor(catalog.df[valid.rows.c, ]$Purchase\_Yes))  Confusion Matrix and Statistics  Reference  Prediction 0 1  0 305 99  1 99 297    Accuracy : 0.7525  95% CI : (0.7210723, 0.782054)  No Information Rate : 0.505  P-Value [Acc > NIR] : <0.0000000000000002    Kappa : 0.5049505  Mcnemar's Test P-Value : 1    Sensitivity : 0.7549505  Specificity : 0.7500000  Pos Pred Value : 0.7549505  Neg Pred Value : 0.7500000  Prevalence : 0.5050000  Detection Rate : 0.3812500  Detection Prevalence : 0.5050000  Balanced Accuracy : 0.7524752    'Positive' Class : 0    >  >  > #Model 5 with Three Hiddens Layer  > nn.5<-neuralnet(Purchase\_No + Purchase\_Yes ~  + US + Web\_order+ Gender + Address\_is\_res +  + last\_update\_S + first\_update\_days\_ago\_S + Freq\_S,  + data=train.data.c, linear.output=FALSE, threshold=0.05, hidden = c(2,2,1)  + )  >  >  > valid.pred.c = compute(nn.5, valid.data.c [ , c(2,21,22,23, 28:30)])  > valid.class.c = apply(valid.pred.c$net.result, 1,which.max)-1  > confusionMatrix(as.factor(valid.class.c),  + as.factor(catalog.df[valid.rows.c, ]$Purchase\_Yes))  Confusion Matrix and Statistics  Reference  Prediction 0 1  0 310 100  1 94 296    Accuracy : 0.7575  95% CI : (0.7262583, 0.7868305)  No Information Rate : 0.505  P-Value [Acc > NIR] : < 0.00000000000000022    Kappa : 0.5148787  Mcnemar's Test P-Value : 0.7196107    Sensitivity : 0.7673267  Specificity : 0.7474747  Pos Pred Value : 0.7560976  Neg Pred Value : 0.7589744  Prevalence : 0.5050000  Detection Rate : 0.3875000  Detection Prevalence : 0.5125000  Balanced Accuracy : 0.7574007    'Positive' Class : 0 |
|  |
|  |