# Report to Tayko

I have performed an analysis of sample mailing based on the Neural Network statistical method. Before we dive into predictions, here are details that are essential for understanding the concept of neural networks.

**Tasks Performed:**

- Pre-processed the dataset in *RStudio*; partitioned the dataset to train the Neural Network model
- Performed the three neural network fittings (models differed in hidden structures); plotted models
- Constructed and compared binary confusion matrices

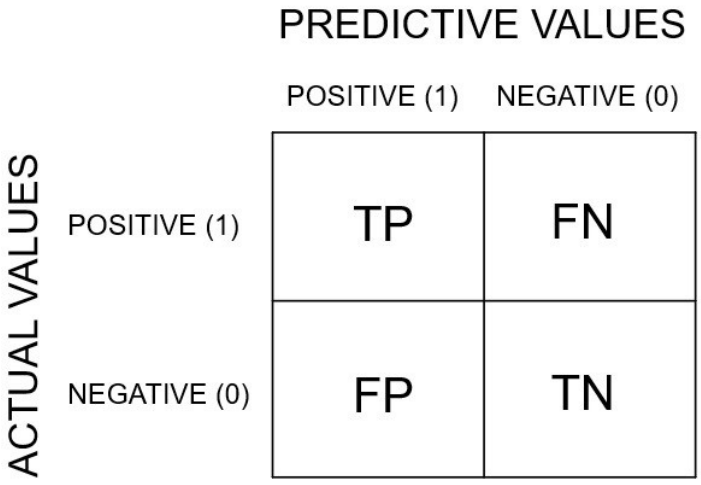
*What do plots mean?*

I have provided some visual support for each network. **Important concepts:**

1. The left-most nodes (i.e., input nodes) are your raw data variables.
2. The arrows in black (and associated numbers) are the **weights** that you can think of as **how much that variable contributes to the next node.** The blue lines are the bias weights. You can find the purpose of these weights in the excellent answer here.
3. The middle nodes (i.e., anything between the input and output nodes) are your hidden nodes. This is where the image analogy helps. **Each of these nodes constitutes a component that the network is learning to recognize.** For example, a nose, mouth, or eye. This is not easily determined and is far more abstract when you are dealing with non-image data.
4. The far-right (output node(s)) node is the final output of your neural network.

*What is Confusion Matrix?*

A confusion matrix provides a summary of the predictive results in a classification problem. Correct and incorrect predictions are summarized in a table with their values and broken down by each class.




*Confusion Matrix for the Binary Classification*

**Definition of the Terms:**
True Positive: You predicted positive and it's true. You predicted that an animal is a cat and it actually is.

True Negative: You predicted negative and it's true. You predicted that animal is not a cat and it actually is not (it's a dog).

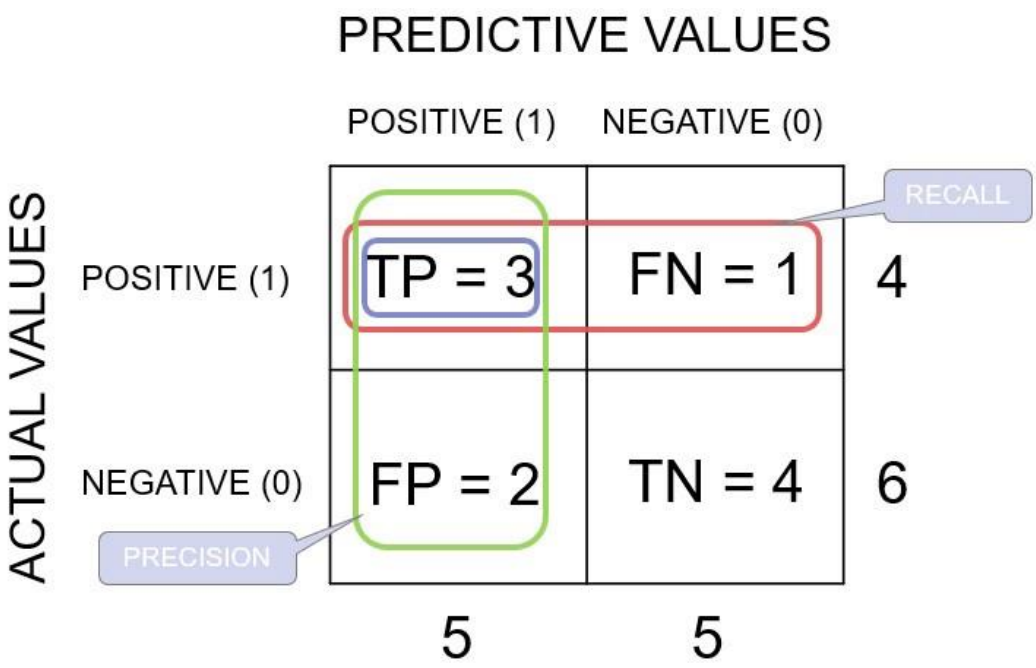
False Positive (Type 1 Error): You predicted positive and it's false. You predicted that animal is a cat but it actually is not (it's a dog).

False Negative (Type 2 Error): You predicted negative and it's false. You predicted that animal is not a cat but it actually is.

**Accuracy** is calculated as the total number of two correct predictions (TP + TN) divided by the total number of a dataset (P + N). Example: Accuracy value of 70% means that identification of 3 of every 10 values is incorrect, and 7 is correct.

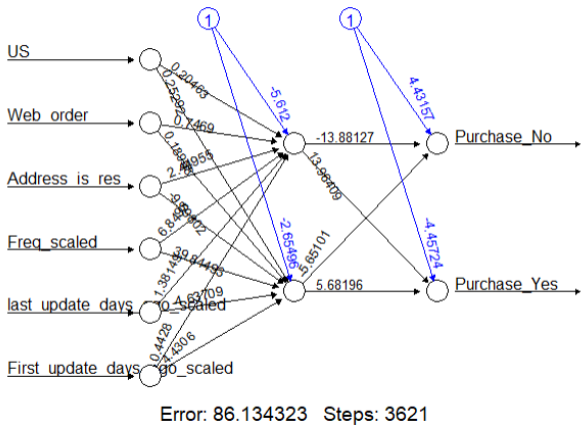
**Recall (Sensitivity):**
Recall is defined as the ratio of the total number of correctly classified positive classes divide by the total number of positive classes. Or, out of all the positive classes, how much we have predicted correctly. Recall should be high.

**Precision:** Precision tells us about when it predicts yes, how often is it correct.



# Analysis

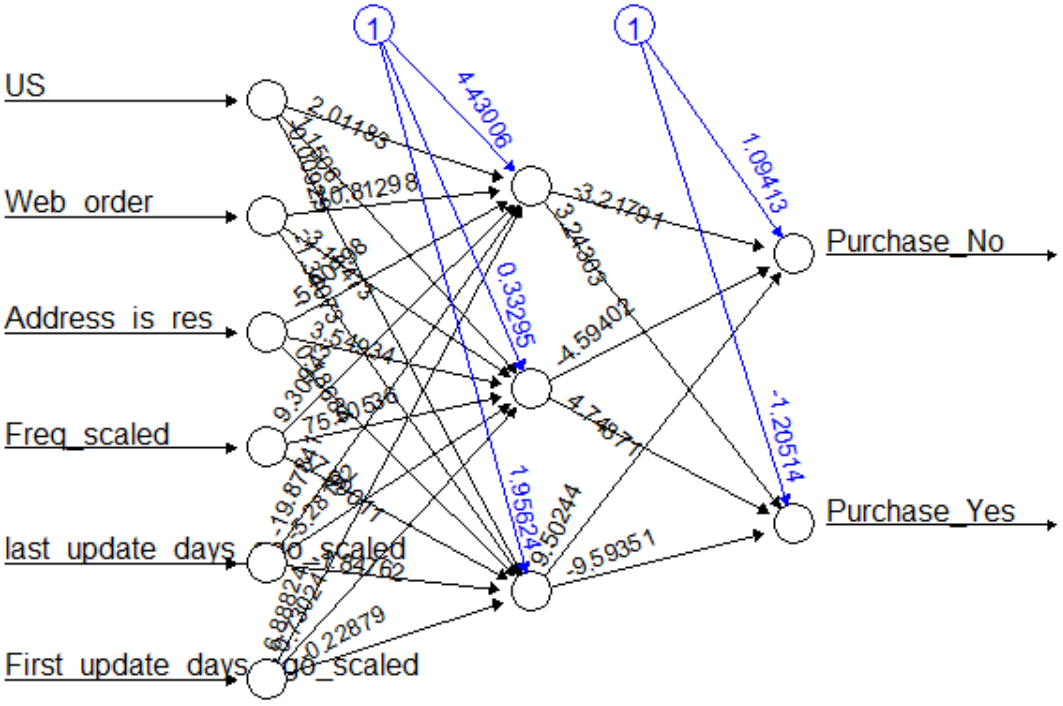
*#Neural Network 1 (one hidden layer with two nodes)*




```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 159  57
         1  34 150

               Accuracy : 0.7725
                 95% CI : (0.7282, 0.8127)
    No Information Rate : 0.5175
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5463

 Mcnemar's Test P-Value : 0.0211

            Sensitivity : 0.8238
            Specificity : 0.7246
         Pos Pred Value : 0.7361
         Neg Pred Value : 0.8152
             Prevalence : 0.4825
         Detection Rate : 0.3975
   Detection Prevalence : 0.5400
      Balanced Accuracy : 0.7742

       'Positive' Class : 0
```

The model predicts values quite accurately (roughly 8 out of 10 predictions are correct). The recall shows that if we predict that people will buy our item, we are correct 8 out of 10 times (same as accuracy). We predicted that 150 people will buy, and they actually did. That is a decent ground for making predictions. The model also did a good job of predicting that people would not buy an item when we expected them not to buy. Also 34 people did not buy an item when we expected them to buy, but 57 people bought an item when we did not expect them to buy. Overall, more people are willing to buy the product.

*#Neural Network 2 (one hidden layer with three nodes)*

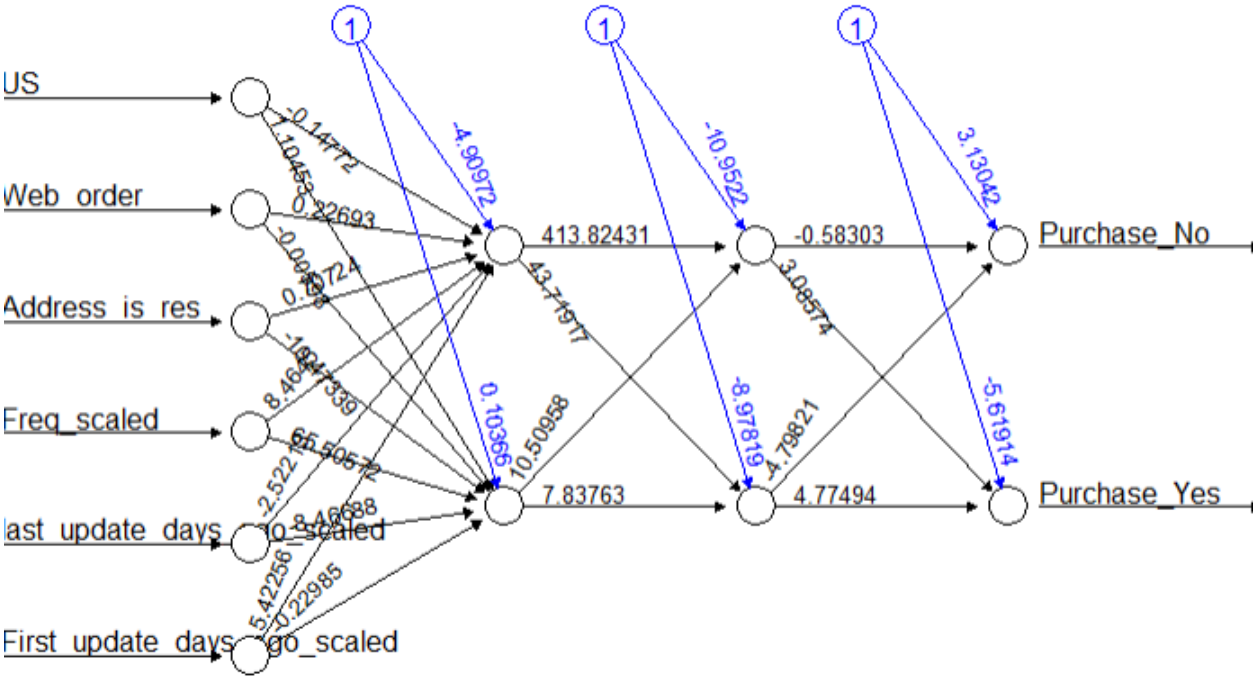



```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 152  60
         1  41 147

               Accuracy : 0.7475
                 95% CI : (0.7019, 0.7894)
    No Information Rate : 0.5175
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.4961

 Mcnemar's Test P-Value : 0.07328

            Sensitivity : 0.7876
            Specificity : 0.7101
         Pos Pred Value : 0.7170
         Neg Pred Value : 0.7819
             Prevalence : 0.4825
         Detection Rate : 0.3800
   Detection Prevalence : 0.5300
      Balanced Accuracy : 0.7489

       'Positive' Class : 0
```

The model predicts values less accurate (roughly 7 out of 10 predictions are correct). The recall shows that if we predict that people will buy our item, we are correct roughly 8 out of 10. We predicted that 147 people will buy, and they actually did. That is a decent ground for making predictions. The model also did relatively good job of predicting that people would not buy an

item when we expected them not to buy. Also 41 people did not buy an item when we expected them to buy, but 60 people bought an item when we did not expect them to buy. Overall, the model shows again that more people are willing to buy the product.

*#Neural Network 3 (two hidden layers with two nodes)*




Error: 85.295681 Steps: 14859

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 151  46
         1  42 161

               Accuracy : 0.78
                 95% CI : (0.7362, 0.8196)
    No Information Rate : 0.5175
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5598

 Mcnemar's Test P-Value : 0.7491

            Sensitivity : 0.7824
            Specificity : 0.7778
         Pos Pred Value : 0.7665
         Neg Pred Value : 0.7931
             Prevalence : 0.4825
         Detection Rate : 0.3775
   Detection Prevalence : 0.4925
      Balanced Accuracy : 0.7801

       'Positive' Class : 0
```

More complex structure of hidden layers and nodes did not make Neural Network model more accurate. We are still in roughly 80% accurate in predictions and recall. The model shows again that more people

are willing to buy an item. The model predicts values less accurate (roughly 7 out of 10 predictions are correct). The recall shows that if we predict that people will buy our item, we are correct roughly 8 out of 10. We predicted that 161 people will buy, and they actually did. That is a decent ground for making predictions. The model also did relatively good job of predicting that people would not buy an item when we expected them not to buy. Also 42 people did not buy an item when we expected them to buy, but 46 people bought an item when we did not expect them to buy.

*Conclusion:*

Third model performed better than another two, but they predict roughly with the same accuracy. Overall, we see that half people are willing to buy items from new catalog, and another half stays indifferent. We might reconsider our target audience to achieve higher conversion of sales.

## References


https://towardsdatascience.com/decoding-the-confusion-matrix-bb4801decbb

https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/


# Appendix

```{r}
Taykopp.df<-read.csv("Taykopp.csv", header = T)
library(neuralnet)
library(caret)
library(e1071)


```

#Taks 1

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

```

```{r}
#Scaling numeric variables
Taykopp.df<- mutate(Tayko.df, Freq_scaled = normalizing(Tayko.df$Freq),
last_update_days_ago_scaled = normalizing(Tayko.df$last_update_days_ago),
First_update_days_ago_scaled = normalizing(Tayko.df$X1st_update_days_ago))

```

```{r}
#Creating new columns for Purchase_Yes and Purchase_No
Purchase_Yes<- ifelse(Taykopp.df$Purchase == 1, "1", "0")
Purchase_No <- ifelse(Taykopp.df$Purchase== 0, "1","0" )

Taykopp.df<- Taykopp.df %>% mutate(Purchase_No, Purchase_Yes)
write.csv(Taykopp.df, "Taykopp.csv")
```

#Task 2

```{r}
#Partitioning Dataset into 60% training and 40% validation datasets
set.seed(1)

train.rows.Tayko <- sample(rownames(Taykopp.df), dim(Taykopp.df)[1]*0.6)
train.data.Tayko <- Taykopp.df[train.rows.Tayko,]

valid.rows.Tayko<-setdiff(rownames(Taykopp.df), train.rows.Tayko)
valid.data.Tayko<-Taykopp.df[valid.rows.Tayko,]

```

#Task 3 & 4

```{r}
#Neural Network 1 (one hidden layer with two nodes)

set.seed(1)
nn1.Tyko <-
neuralnet(Purchase_No+Purchase_Yes~US+Web_order+Address_is_res+Freq_scaled+last_update_days_ago_scaled+First_update_days_ago_scaled, data=train.data.Tayko, linear.output = F, threshold = 0.05, hidden = 2)
plot(nn1.Tyko, rep = "best")

valid.pred.Tayko = compute(nn1.Tyko, valid.data.Tayko[,c(2:5,7:9)])
valid.class.Tayko=apply(valid.pred.Tayko$net.result,
                  1,which.max)-1
confusionMatrix(as.factor(valid.class.Tayko),
            as.factor(Taykopp.df[valid.rows.Tayko,]$Purchase))

```

```{r}
#Neural Network 2 (one hidden layer with three nodes)

set.seed(1)
nn2.Tyko <-
neuralnet(Purchase_No+Purchase_Yes~US+Web_order+Address_is_res+Freq_scaled+last_update_days_ago_scaled+First_update_days_ago_scaled, data=train.data.Tayko, linear.output = F, threshold = 0.05, hidden = 3)
plot(nn2.Tyko, rep = "best")
```

```
valid.pred.Tayko = compute(nn2.Tyko, valid.data.Tayko[,c(2:5,7:9)])
valid.class.Tayko=apply(valid.pred.Tayko$net.result,
                  1,which.max)-1
confusionMatrix(as.factor(valid.class.Tayko),
            as.factor(Taykopp.df[valid.rows.Tayko,]$Purchase))

```

```{r}
#Neural Network 3 (two hidden layers with two nodes)

set.seed(1)
nn3.Tyko <-
neuralnet(Purchase_No+Purchase_Yes~US+Web_order+Address_is_res+Freq_scaled+last_update_days_ago_scaled+First_update_days_ago_scaled, data=train.data.Tayko, linear.output = F, threshold = 0.05, hidden = c(2,2))
plot(nn3.Tyko, rep = "best")

valid.pred.Tayko = compute(nn3.Tyko, valid.data.Tayko[,c(2:5,7:9)])
valid.class.Tayko=apply(valid.pred.Tayko$net.result,
                  1,which.max)-1
confusionMatrix(as.factor(valid.class.Tayko),
            as.factor(Taykopp.df[valid.rows.Tayko,]$Purchase))

```