DEEP NEURAL NETWORK REPORT

a) Overall discovery process:

- We discovered that cleaning-up our data was more important than the actual model since garbage in, garbage out. We check the dictionary and learn about different features and their nature. By utilizing class() and unique() function, we have learned and classify different types of columns, whether they are categorical variables or continuous variables. We check and count the "NA" by column and by row to see the characteristics and drawbacks of the dataset as well as if there is any potential that we could explore.

We had two approaches to data preprocessing as explained below:

Approach 1:

- We removed columns with NA and missing values combined accounting to more than 65% of the number of samples. There were 15 such columns and they were removed
- We then split the cleaned data initially into categorical and numerical subsets and performed lasso and random forest for feature selection. Also, to compensate for the imbalanced data we did oversampling. However, the model did not perform well, the specificity was 0.49.

Approach 2:

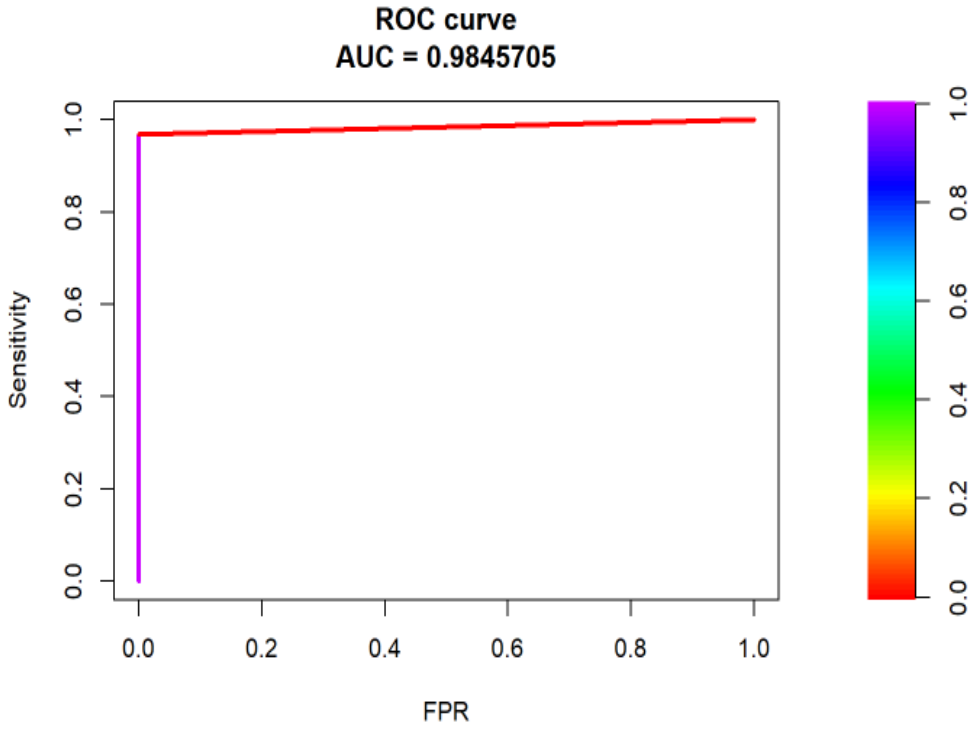
- We removed columns with NA and missing values combined accounting to more than 95% of the number of samples. Also, we removed any rows with all NA values.
- We accounted for the entire dataset and performed lasso regression for feature selection. We selected 180 features for the final model, which we normalized before using for the model building process. With this approach, we achieved a specificity of 0.96 and therefore we selected this model.

b) List of variables used in the model and any mechanism that you use to address whether it is important or not:

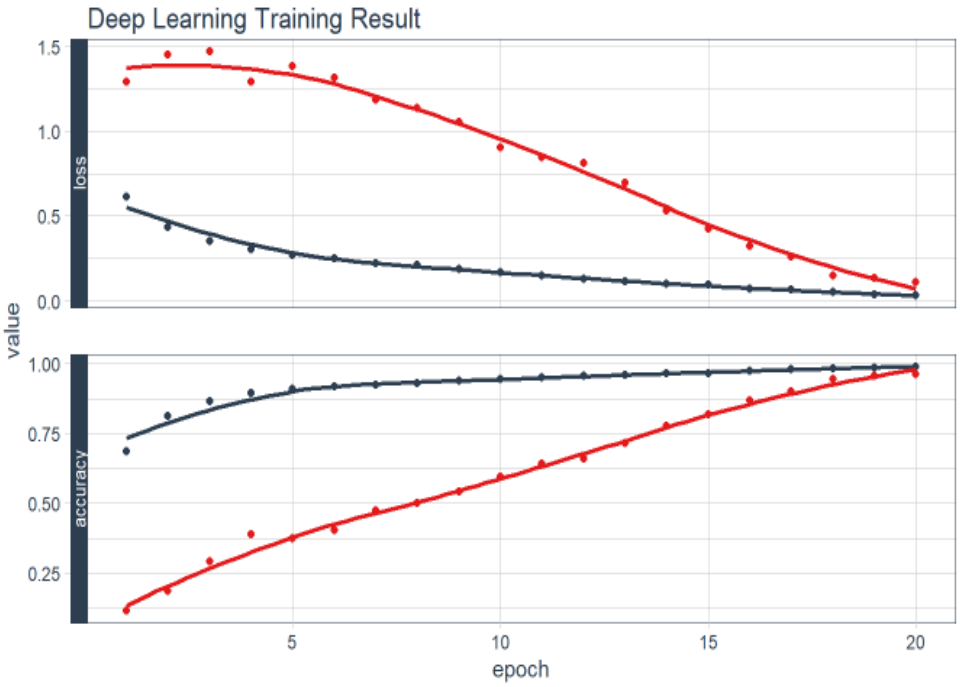
- Our mechanism is to utilize Lasso with our latest dataset after updating and find the best lambda.1se and corresponding coefficient list with chosen predicted variables.
- 180 columns: refer to attached excel spreadsheet (I will write column names to a csv)
- We found that the method followed in feature extraction did impact the performance of the model.

(c) Detailed analysis on model performance (e.g. ROC curves, confusion matrix, etc.)

The model seems too good to be true and not so realistic but it has the best specificity so we go with this model. Below is the confusion matrix, ROC curves and deep learning training results of the model. We use the Lasso lambda.1se instead of the lambda min to decide our feature selection since it return less feature in our scope of coefficient and make R compiler faster when running while giving the same results.




```
plot(final_model) + theme_tq() +
scale_color_tq() +scale_fill_tq() + labs(title = "Deep Learning Training Result")
```



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2035   37
##          1    0 1162
##
##                Accuracy : 0.9886
##                  95% CI : (0.9843, 0.9919)
##     No Information Rate : 0.6293
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9753
##
##  Mcnemar's Test P-Value : 3.252e-09
##
##             Sensitivity : 1.0000
##             Specificity : 0.9691
##          Pos Pred Value : 0.9821
##          Neg Pred Value : 1.0000
##              Prevalence : 0.6293
##          Detection Rate : 0.6293
##    Detection Prevalence : 0.6407
##       Balanced Accuracy : 0.9846
##
##        'Positive' Class : 0
##
```

(d) Limiting factors/issues encountered and assumptions made.

- An issue we found that the data was heavily imbalanced. We did undersampling to compensate for it.
- An assumption that we made was that we treated empty strings as NA but that is not always the case since empty is not directly implied as NA, it could have and bring different meaning and value to the dataset.
- Another issue that we have is that we encountered a large amount of categorical and continuous features that we had to find a way to encode categorical ones to select the best ones later. One hot function in R makes each level become separated columns and results in large amounts of columns, which makes the Rmarkdown slower to compile.