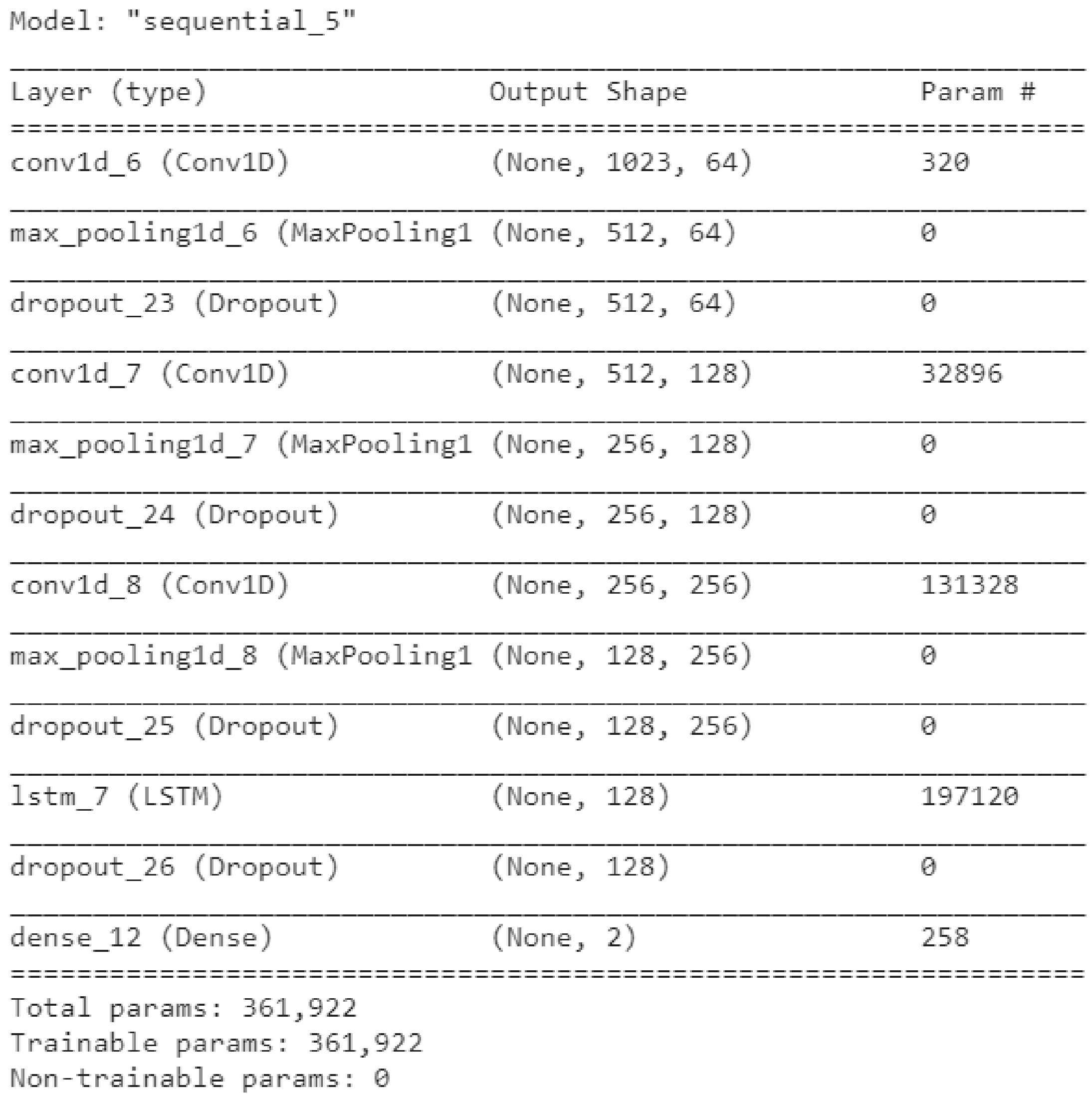
Mikias Berhanu | 2021280115

Assignment 8

DNNs, LSTM, CNN, RNN

1. Experiment with Convolutional Neural Networks

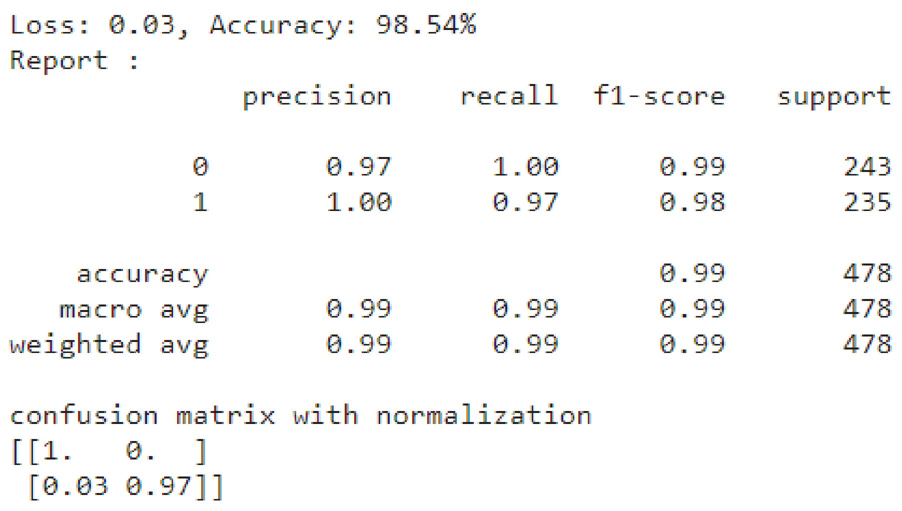
This experiment uses convolutional layers to extract features from the dataset. Convolutional neural networks are widely used in computer vision but they can also be used with one dimensional data. This experiment uses one dimensional Convolution layers to extract features. On top of that it also uses a one dimensional max pooling layer to downsample the features extracted by the Convolution layer. In order to tackle overfitting it use Dropout layer with 1% value, meaning 1 percent of our neurons are turned off randomly during training.



```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv1d_6 (Conv1D)            (None, 1023, 64)          320       
_________________________________________________________________
max_pooling1d_6 (MaxPooling1 (None, 512, 64)           0         
_________________________________________________________________
dropout_23 (Dropout)         (None, 512, 64)           0         
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 512, 128)          32896     
_________________________________________________________________
max_pooling1d_7 (MaxPooling1 (None, 256, 128)          0         
_________________________________________________________________
dropout_24 (Dropout)         (None, 256, 128)          0         
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 256, 256)          131328    
_________________________________________________________________
max_pooling1d_8 (MaxPooling1 (None, 128, 256)          0         
_________________________________________________________________
dropout_25 (Dropout)         (None, 128, 256)          0         
_________________________________________________________________
lstm_7 (LSTM)                (None, 128)               197120    
_________________________________________________________________
dropout_26 (Dropout)         (None, 128)               0         
_________________________________________________________________
dense_12 (Dense)             (None, 2)                 258       
=================================================================
Total params: 361,922
Trainable params: 361,922
Non-trainable params: 0
```

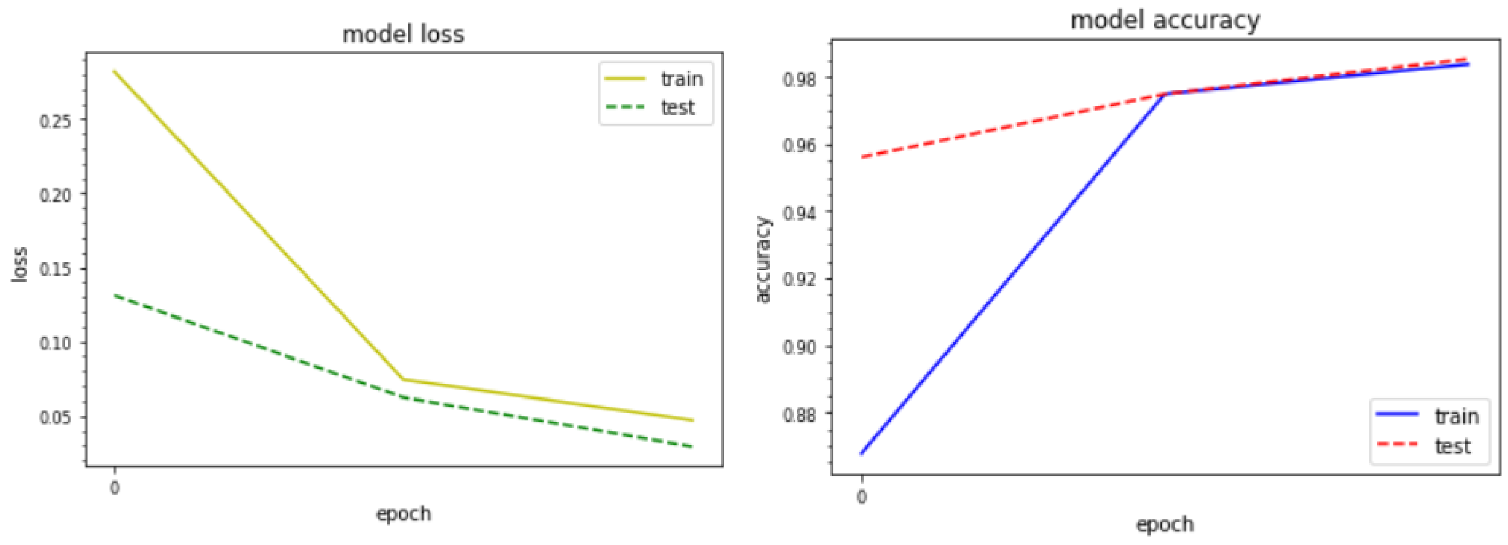
Sequential model summary

The model is compiled with a loss function Categorical Cross Entropy since we have multiple classes to classify and an optimizer Adam which is another variant of Stochastic Gradient Descent algorithm. The metrics used is accuracy meaning we will measure the accuracy of the classification on each training step. The training is done for three epochs where the model's weights and metrics are saved by using check points on each epoch. This is helpful that later we can use the checkpoints rather than retraining the model. The dataset was splitted using the usual 80:20 ratio, 80% for training and 20 percent for testing. The model was capable of achieving 98.54% accuracy and 0.03% loss.

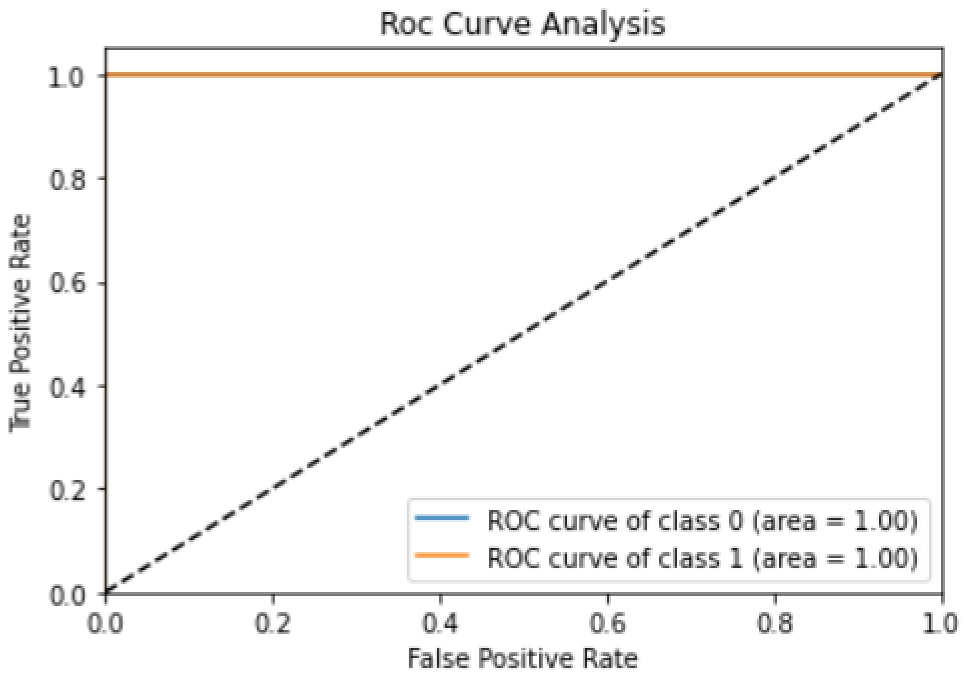


```
Loss: 0.03, Accuracy: 98.54%
Report :
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       243
           1       1.00      0.97      0.98       235

    accuracy                           0.99       478
   macro avg       0.99      0.99      0.99       478
weighted avg       0.99      0.99      0.99       478

confusion matrix with normalization
[[1.   0.  ]
 [0.03 0.97]]
```

Model's Performance



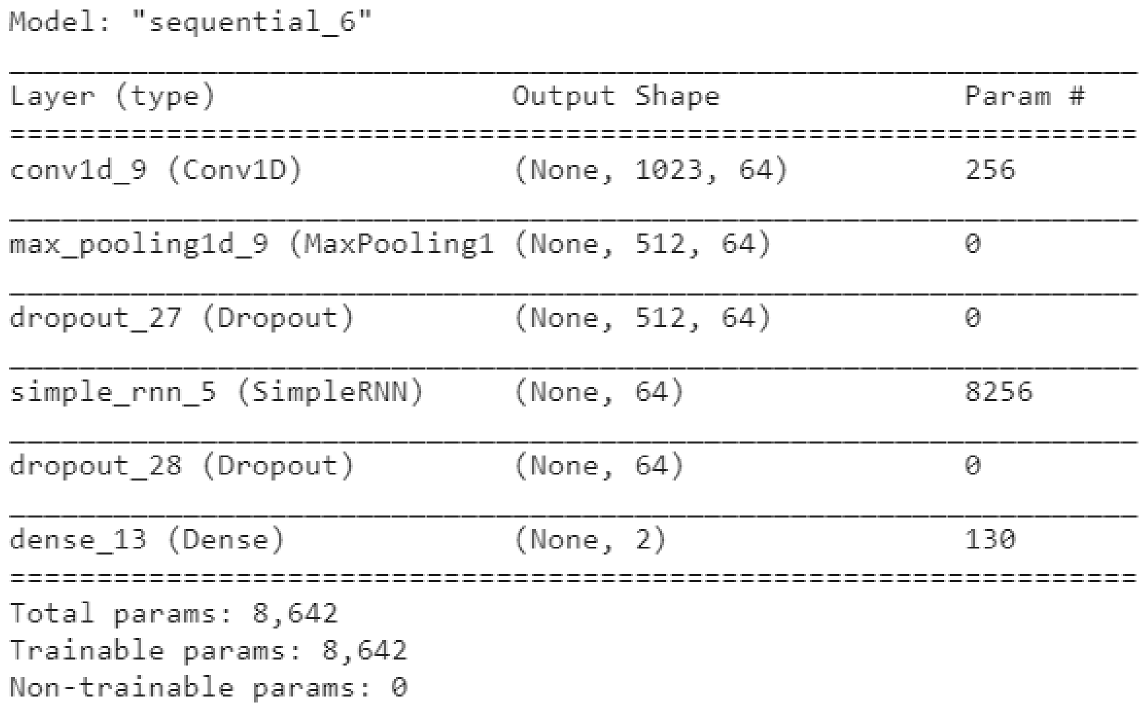
Model's loss and accuracy plot



Model's ROC curve

2. Experiment with Recurrent Neural Networks

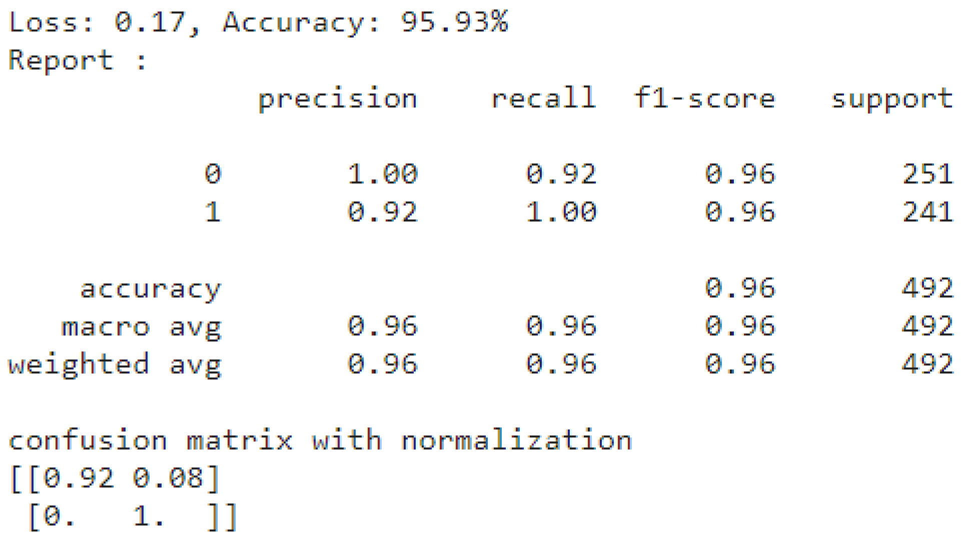
This second experiment uses Recurrent Neural Networks to extract features from the dataset. Recurrent neural networks are used in natural language processing and any form of data that comes in the form of time series like sound waves, weather, stock market etc. RNNs work in a form of recurrence where they hold important information about the past which will be used to predict the future. The model uses a SimpleRNN layer with 64 units and also one Dropout layer with 0.1 value to tackle the overfitting. THe final output layer is Dense layer with an activation of softmax which will force the output to be in between 0 and 1 as a probability distribution for each class.



```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv1d_9 (Conv1D)            (None, 1023, 64)          256       
_________________________________________________________________
max_pooling1d_9 (MaxPooling1 (None, 512, 64)           0         
_________________________________________________________________
dropout_27 (Dropout)         (None, 512, 64)           0         
_________________________________________________________________
simple_rnn_5 (SimpleRNN)     (None, 64)                8256      
_________________________________________________________________
dropout_28 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_13 (Dense)             (None, 2)                 130       
=================================================================
Total params: 8,642
Trainable params: 8,642
Non-trainable params: 0
```

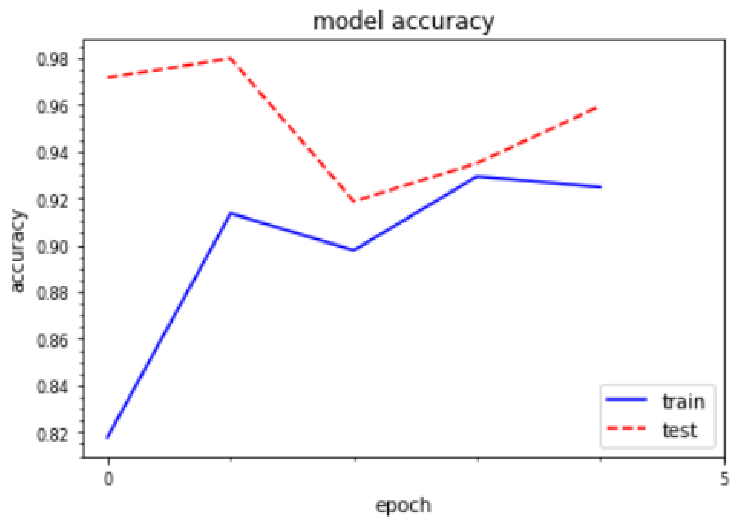
Model structure

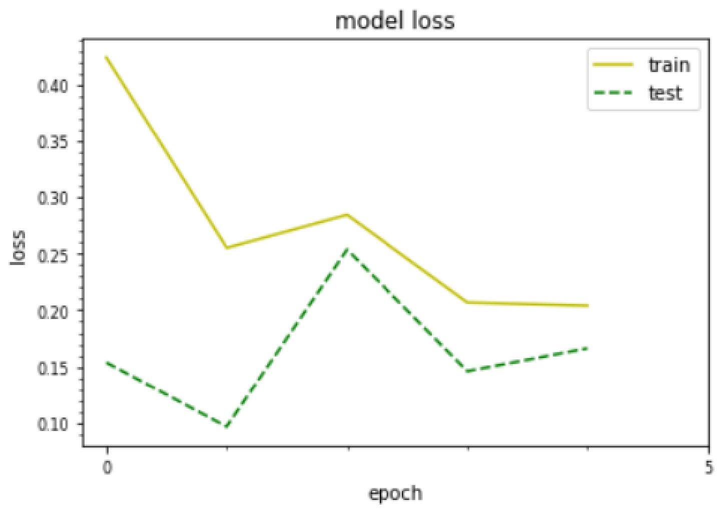
The model is compiled with loss function Categorical Cross Entropy and an optimizer function Adam. The metrics used is accuracy just like the previous one to measure the performance of the model. The data is splitted as 80 percent for training and 20 percent for testing. The model is trained for 5 epochs with a batch size of 32 meaning for each epoch 32 batches of data samples are used for training and on each epoch the model's weights are saved using checkpoints. With this simple setting the model achieves 95.93% accuracy and 0.17% loss which is pretty decent considering how simple the model architecture is.



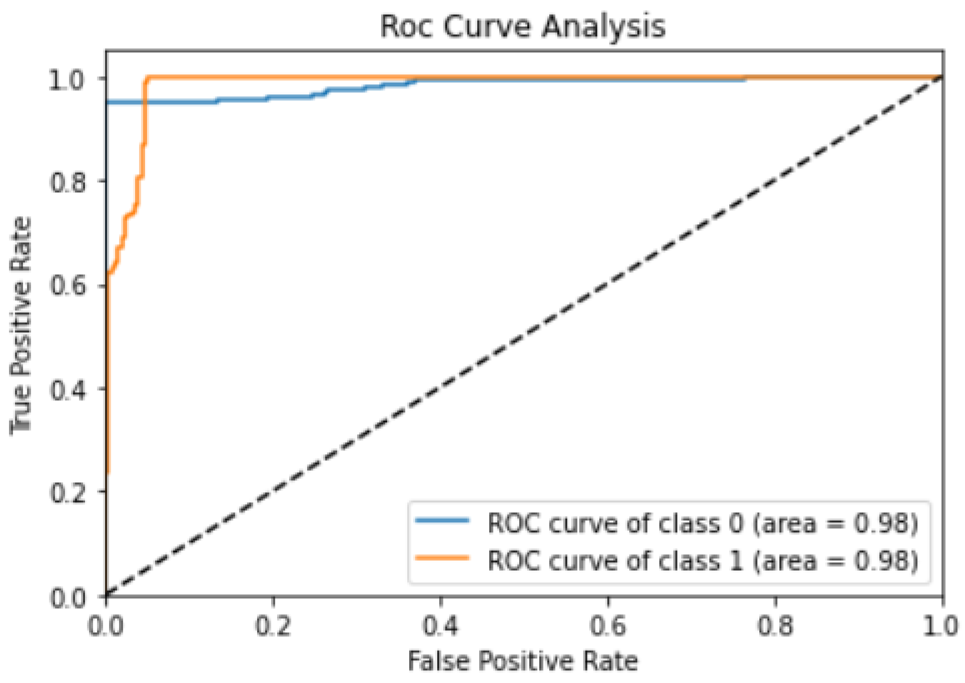
```
Loss: 0.17, Accuracy: 95.93%
Report :
              precision    recall  f1-score   support

           0       1.00      0.92      0.96       251
           1       0.92      1.00      0.96       241

    accuracy                           0.96       492
   macro avg       0.96      0.96      0.96       492
weighted avg       0.96      0.96      0.96       492

confusion matrix with normalization
[[0.92 0.08]
 [0.   1.  ]]
```

Model's Performance





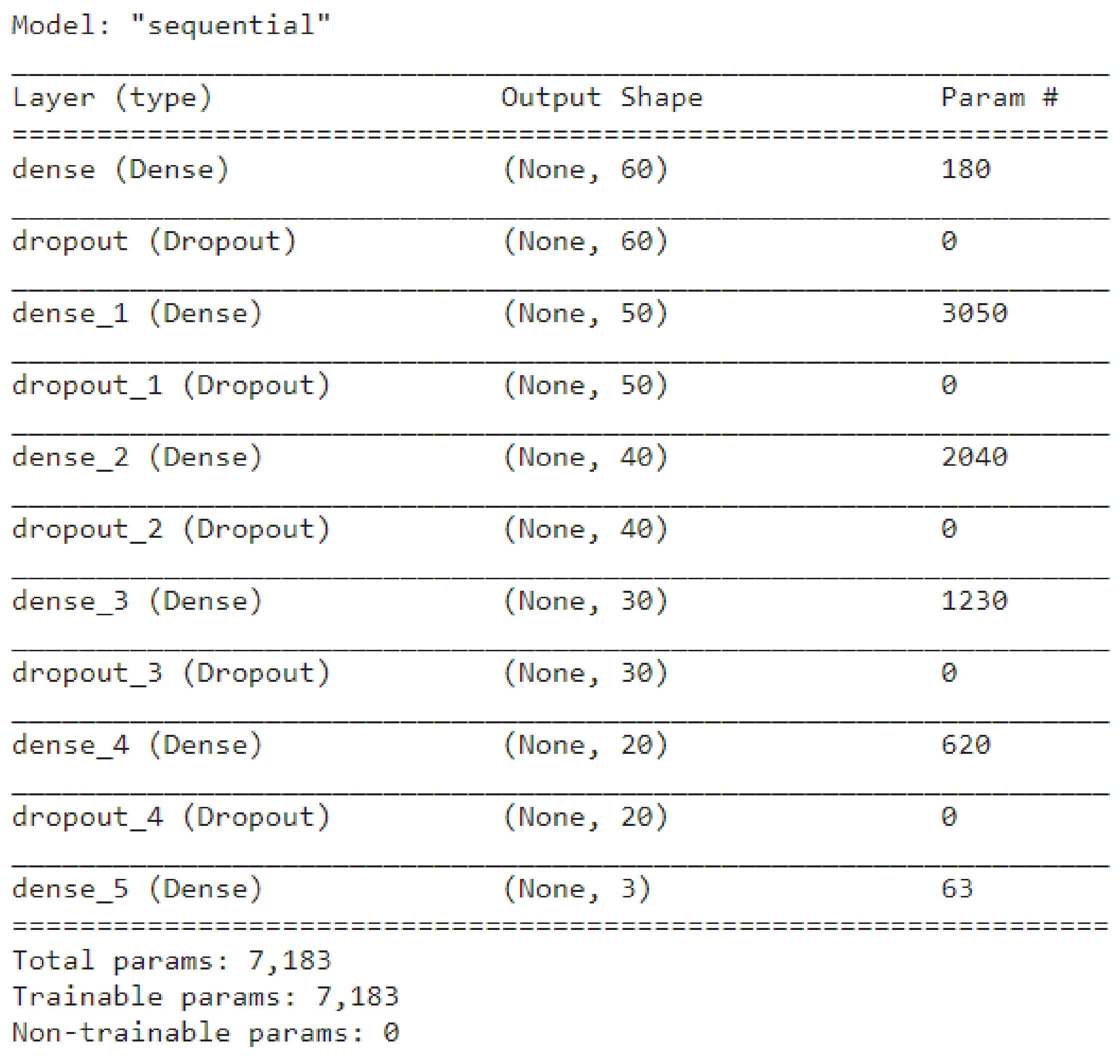
Model's Accuracy and Loss plot



Model's ROC curve

3. Experiment with Deep Neural Networks

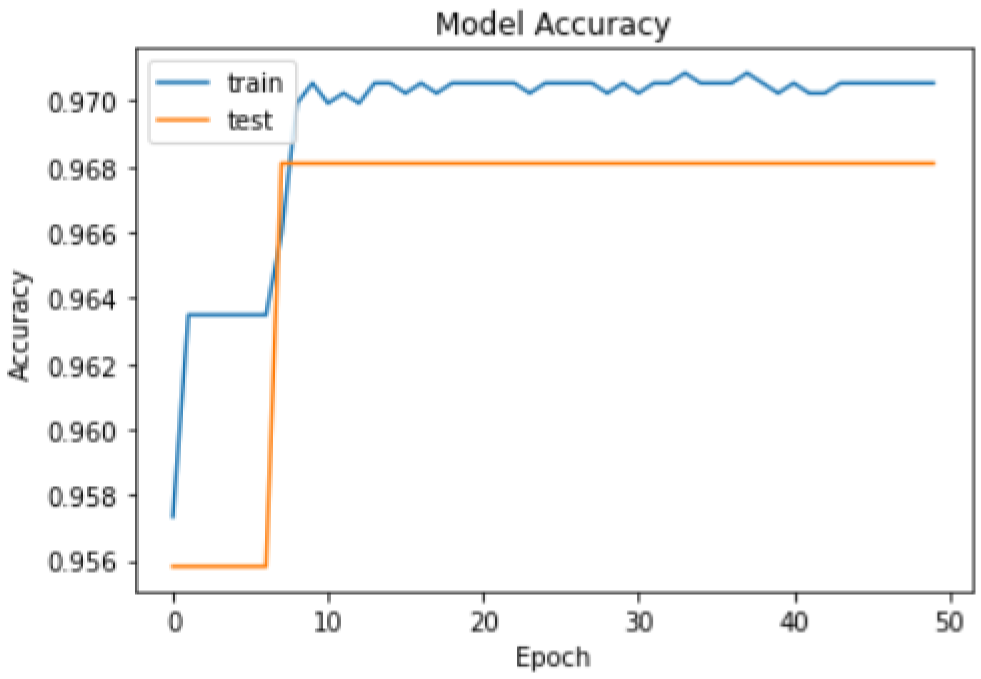
The third experiment uses Dense layers for classifying data samples. Dense layers are just layers of neurons which are connected to the previous one, where each neuron in the current layer is connected to the previous and the next neuron inside another layer. The experiment uses a total of 6 Dense layers, 5 of them used for extracting features with activation function relu and the last one for binary classification with sigmoid activation function. Same as the above two experiments it uses Dropout layers with 0.2 value to tackle overfitting. The number of units used for the dense layers decreases by 10 differences starting from 60 down to 20 and the last layer will have only 3 units of neurons.



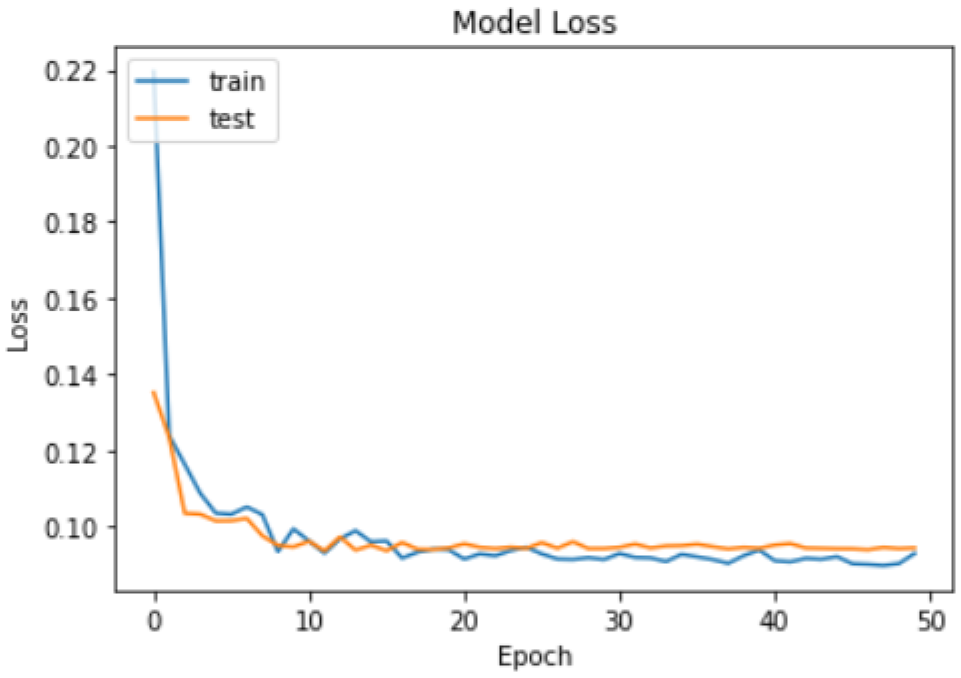
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 60)                180
_________________________________________________________________
dropout (Dropout)            (None, 60)                0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                3050
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 40)                2040
_________________________________________________________________
dropout_2 (Dropout)          (None, 40)                0
_________________________________________________________________
dense_3 (Dense)              (None, 30)                1230
_________________________________________________________________
dropout_3 (Dropout)          (None, 30)                0
_________________________________________________________________
dense_4 (Dense)              (None, 20)                620
_________________________________________________________________
dropout_4 (Dropout)          (None, 20)                0
_________________________________________________________________
dense_5 (Dense)              (None, 3)                 63
=================================================================
Total params: 7,183
Trainable params: 7,183
Non-trainable params: 0
```

Model summary

The model is compiled with Binary Cross Entropy loss function and Adam as an optimizer function. As usual the metrics used for measuring the performance of our model is accuracy. The model is trained over 15 epochs and a batch size of 15, the test set is used for validation which is passed along with the training data. Based on these settings the model's accuracy during training is 0.9705 which is 97% and the loss is 0.0928 and during validation the model's accuracy is 96% and loss of 0.0943.



Model’s accuracy plot

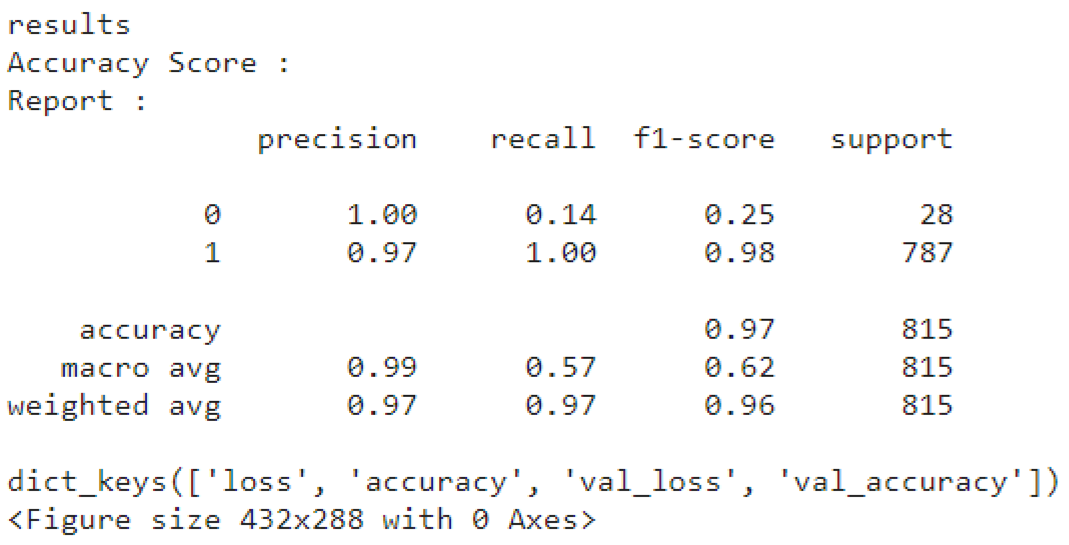


Model’s Loss plot

4. Experiment with Long Short Term Memory(LSTM)

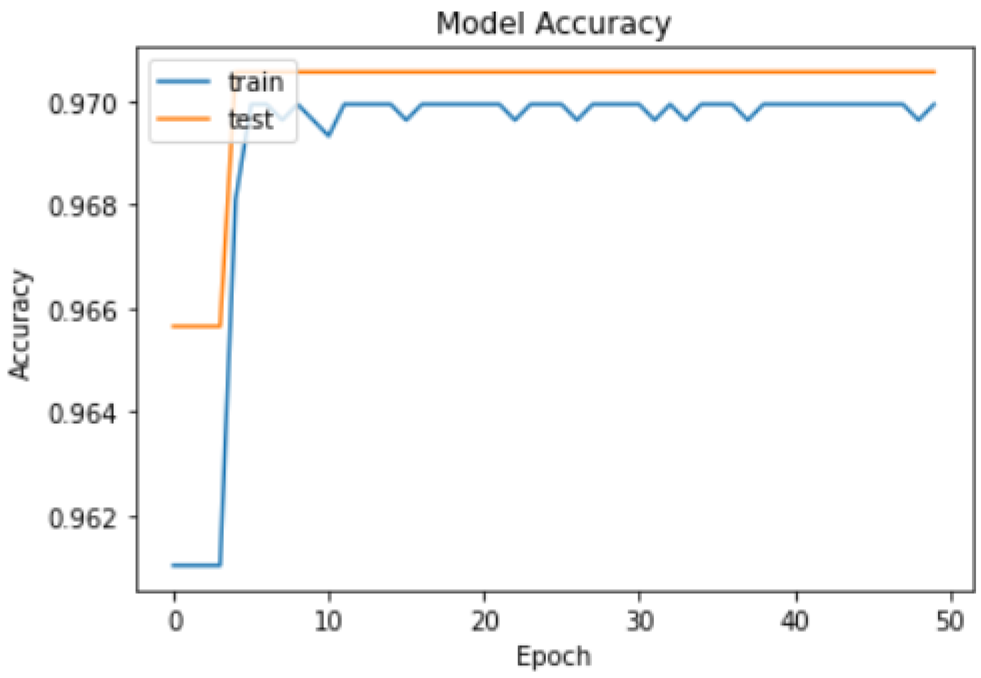
This last experiment uses LSTMs which are another variant of the Recurrent Neural Networks which are created to solve the vanishing gradient problem which is very common in normal Recurrent Neural Networks. LSTMs have extra gates inside their architecture which makes them more computationally expensive and slow but they are more accurate compared to RNNs. This experiment uses five LSTM layers and two Dense layers with relu activation. The first LSTM layer has 80 units and the preceding layers decrease by 10 units. All the LSTM layers return sequences from the previous layer which is quite helpful to make good predictions and the activation function used is relu. The Dense layer which is also set with relu actional function with 20 units. The final Dense layer has 3 units of neurons with sigmoid activation function which will do the final prediction. Just like the previous model's this one is compiled with Binary Cross Entropy loss function and Adam activation function. The performance measure is as usual Accuracy.

The data is splitted 80 percent for training and 20 percent for testing. The model is trained over 50 epochs over 15 batches. The testing data is used as a validation set. Based on this setting the model has an accuracy of 96 percent during training and 97 percent during validation the loss value during training reaches 0.1365 and 0.1334 during validation.

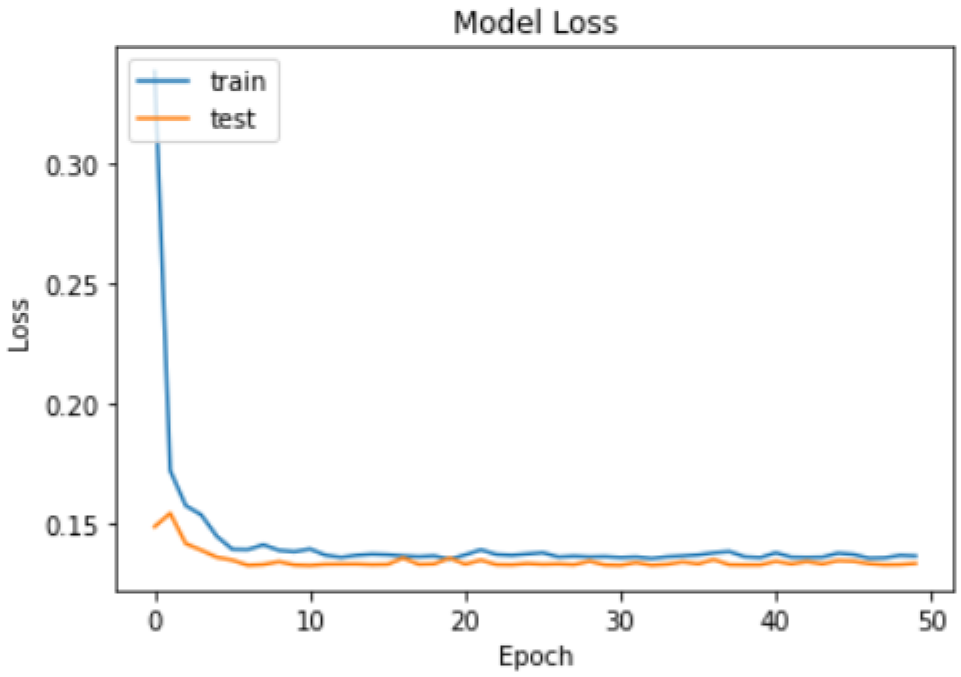


```
results
Accuracy Score :
Report :
              precision    recall  f1-score   support

           0       1.00      0.14      0.25        28
           1       0.97      1.00      0.98       787

    accuracy                           0.97       815
   macro avg       0.99      0.57      0.62       815
weighted avg       0.97      0.97      0.96       815

dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
<Figure size 432x288 with 0 Axes>
```

Model's Performance



Model's Accuracy plot



Model's Loss plot