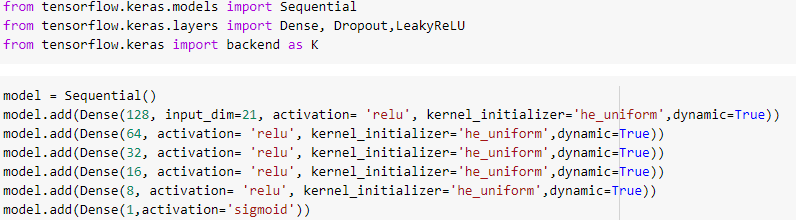
Libraries from the Keras package is called in order to create the neural network.



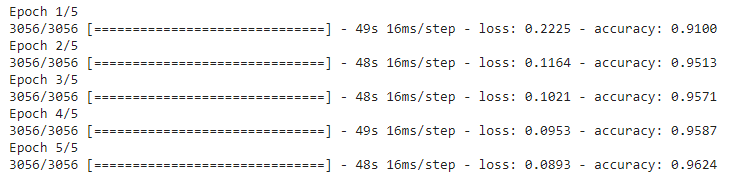
Since neural networks require a series of layers, the *Sequential* function is called to initialize the model. Afterwards, the input layer is created to take in our 21 input variable and then the first hidden layer is created with 128 nodes. A total of 5 hidden layers are created with 64, 32, 16, and 8 layers respectively. The Rectified Linear Unit activation function is used for all hidden layers as it has been noted to achieve generally better performance than the sigmoid and tanh functions.



The model is then compiled with a cross entropy loss function for binary classification problems. The Adam algorithm is used as the optimizer. This is a gradient descent method that automatically tunes itself and adapts the learning rates as the model develops.



Here, the model is fitted using the training data. An epoch of 5 and batch size of 34 is arbitrarily chosen. In other words, the network will go through the training dataset 5 times and take a sample of rows in order to develop the model and update its parameters, where *N* is the total number of rows in the dataset. The results are shown below:

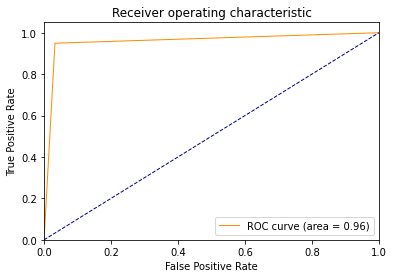


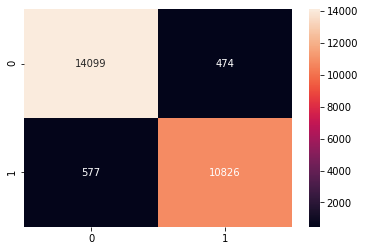
In this execution, running 5 epochs resulted in the highest accuracy rate. Generally speaking as well, the accuracy rate is very close to 1,indicating that this neural network model is performing very well. As for time, it took about 48 seconds to run each epoch, so it was relatively quick.

The model is then performed with the test data and the values are cleaned to ensure that they represent binary responses.



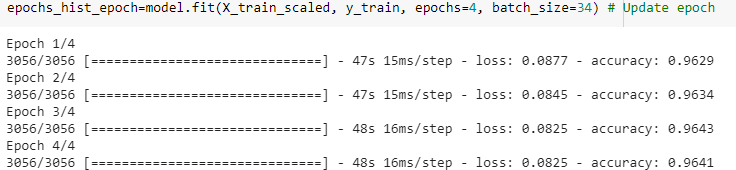
Calling the *accuracy\_score* function shows that we have an accuracy rate of about 96%. We can also see the accuracy by examining the ROC curve and confusion matrix.





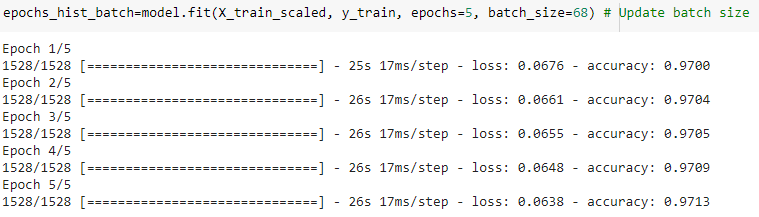
The model has already been fairly optimized since the ReLU activation function allows for a generally high performance and Adam algorithm allows for automatic tuning. However, several models are created to further explore how altering different components can affect the results.

First, we experiment with the epoch and batch size values. Ideally, we would be able to reduce these amounts and maintain accuracy while reducing in the time it takes to run. An epoch of 4 for instance produced the following result:



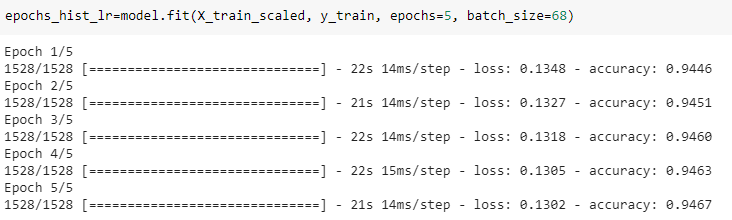
The overall accuracy and loss with this implementation seems to be more consistent than the original model, but the overall results are very similar.

Next, we explore batch size. A lower batch size would mean that we would need more samples for each epoch iteration. This would increase the time it would take to run. Because the model is already very accurate, the batch size is increased to see if we can maintain accuracy while also decreasing the time.



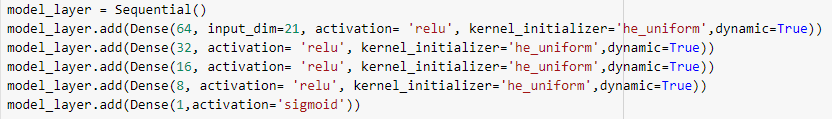
Using a batch size of 68 significantly reduced the time it took to fit the model with 5 epochs. Notably, we also see a very slight increase in accuracy. Since there is an improvement in accuracy and time, this batch size is kept.

Afterwards, we explore the learning rate by using the *SGD* library. The default learning rate in the Keras package is 0.01. We decrease it to 0.001 to see if a lower rate will improve the model.

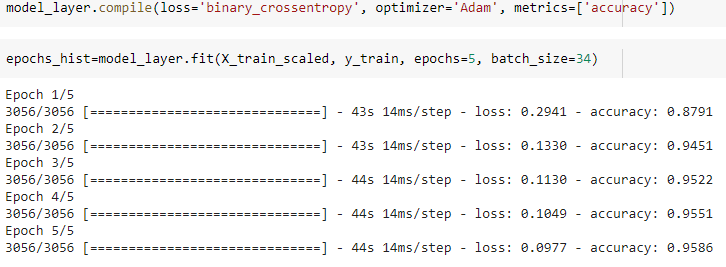


The time improved by a few seconds, but the accuracy suffered with this new learning rate. It is important to note though that smaller learning rates may need more training epochs to prevent the model from converging too quickly to a solution. If we choose not to use an adaptive learning rate, we will need to look alter the other parameters to see what works best with a specific learning rate.

Finally, we see how the number of layers and nodes can affect the model. We take out a layer and update the number of nodes.



All other factors are then left as the original model and then compiled.



We see here that it took less time to fit the model. However, it is slightly less accurate than our original model.

**Reference**

Brownlee, J. (2019, January 25). *Understand the Impact of Learning Rate on Neural Network Performance*. Machine Learning Mastery. Retrieved from https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

Brownlee, J. (2019, July 24). *Your First Deep Learning Project in Python with Keras Step-By- Step*. Machine Learning Mastery. Retrieved from https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/