**University of Western Ontario**
London, Canada
Department of Computer Engineering
Software Engineering Program, Data Analytics Course

# Assignment two: Breast Cancer Detection

**Written by: Tina Gholami**
**Student No: 251190343**

**Date: 15th November 2020**
**Supervisor: Prof. Katarina Grolinger**

## Table of Contents

## (I) Problem

In this problem, the goal is to classify tumors into either benign or malignant, using the medical examination dataset of several patients. Therefore, this is a binary classification problem. The dataset consists of several feature values of tumors, such as the tumor size, thickness, etc. The details of the approach to help solve this classification question is further analyzed in section (III).

Solving this problem can help physicians analyze their patient’s cancer test results with the help of Machine Learning. Physicians can see how their opinions and forecasts on the state of a tumor is in line with the predictions of the Machine Learning model in this paper. The Machine Learning classifier may not win against a physician’s view, since it may lack sufficient and precise information of the patient. But the Machine Learning classifier can certainly indicate underlying features and highlight hidden patters in the data, which a physician may overlook.

Therefore, taking advantage of Machine Learning algorithms will not only assure a doctor on his predictions, but also it may correct him if he is not taking important features and attributes of the examination data into account. Hopefully, by developing more efficient and trustworthy algorithms, Machine Learning might someday give more reliable results than a doctor’s predictions.

## (II) Dataset

The dataset is from UCI's Breast Cancer Wisconsin (Modified) dataset, collected by Dr. William H. Wolberg, University of Wisconsin Hospitals, Madison, Wisconsin, USA. Samples arrive periodically, as Dr. Wolberg reported his clinical cases. The dataset, therefore, reflects this chronological grouping of the data. The dataset involves a total of 683 observations and 11 features. These attributes include:

1. Sample code number: id number
2. Clump thickness: 1 - 10
3. Uniformity of cell size: 1 - 10
4. Uniformity of cell shape: 1 - 10
5. Marginal Adhesion: 1 - 10
6. Single Epithelial Cell Size: 1 - 10
7. Bare Nuclei: 1 - 10
8. Bland Chromatin: 1 - 10
9. Normal Nucleoli: 1 - 10
10. Mitoses: 1 - 10

**11.** Class: (2 for benign, 4 for malignant)

## (III) Methodology and Algorithms

In the previous assignment, after operating necessary data pre-processing techniques on the dataset, three distinct Machine Learning algorithms were designed and evaluated. These models included:

1. Decision Tree Classifier
2. Kernel SVM (support vector machine) Classifier
3. ANN (neural network) Classifier

In this assignment, however, the focus is on parameter tuning and then comparing the tuned results to the previous un-tuned results of Assignment one. The algorithm that is chosen and also instructed to be tuned on, is the ANN (neural network) classifier.

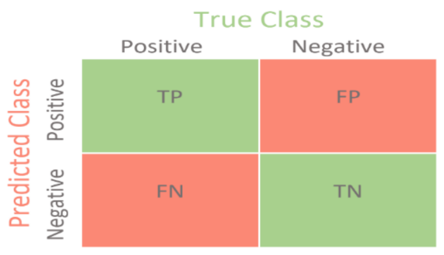
Also, the metrics used to evaluate the models are:

1. Accuracy score and confusion matrix
2. Precision
3. AUC (area under the curve) and ROC (receiver operating characteristic curve) score

Below, is a short description of how each of the abovementioned metrics operate:

1. Accuracy score and confusion matrix:

   As its name suggests, the confusion matrix returns a $4 \times 4$ matrix, explaining the model's performance. Also, this matrix is regarded as a basis for defining other metrics. The columns and rows of the confusion matrix represent true positive, false positive, false negative, and true negative as in the following picture:



***Confusion matrix schematic, towardsdatascience.com***

True positive (TP) refers to the number of predictions where the classifier correctly predicts the positive class as positive.

False positive (FP) refers to the number of predictions where the classifier wrongly predicts the negative class as positive.

True negative (FN) refers to the number of predictions where the classifier correctly predicts the negative class as negative.

False negative (FN) refers to the number of predictions where the classifier wrongly predicts the positive class as negative.

Therefore, the accuracy score can be calculated by taking the average of the values lying across the main diagonal of the confusion matrix over all the samples, i.e.:

$$Accuracy = \frac{True\ positive\ (TP) + True\ negative\ (TN)}{Total\ samples}$$

Thus, accuracy helps in understanding how well the model's performance is. But we have to beware of a common issue with the accuracy score, which happens when one particular class in the labels is quite rare. In that case, we might have a pretty lousy model, but the accuracy is misleadingly high. Therefore, it is suggested to employ other Machine Learning metrics, just in case such a thing happens.

**2.** Precision:

$$Precision = \frac{True\ positive\ (TP)}{True\ positive\ (TP) + False\ positive\ (FP)}$$

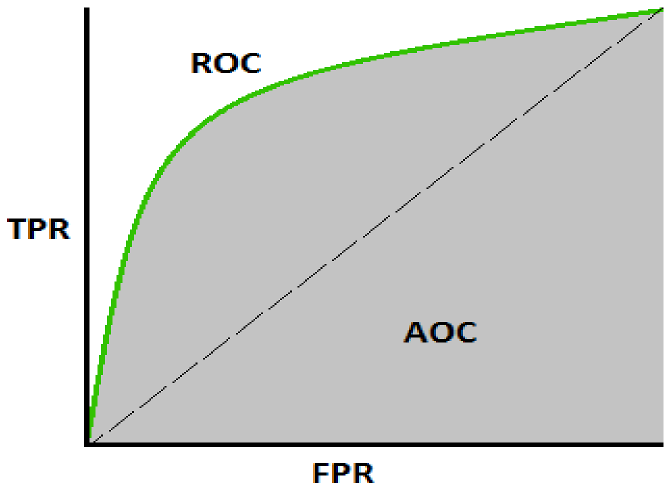
Precision is about how precise our model is from the predicted positives, meaning how many of the predicted positives are actual positives. Precision is a good measure to determine if the costs of false positives are high. For instance, in an email spam detection model, if the cost of false positives is high, many non-spam emails will be predicted as spam. Therefore, the user might lose vital information in the process.

Hence, the precision score is a useful metric to use alongside with the accuracy score.

**3.** AUC (area under the curve) and ROC (receiver operating characteristic curve) score:

This metric computes area under the receiver operating characteristic curve from prediction scores. It is a performance measurement specifically for the classification problems at various thresholds. ROC is a probability curve, and AUC displays a degree of separability. It indicates the model's ability to distinguish between classes. Hence, the higher the AUC, the better the model predicts class 0 as 0 and class 1 as 1. In the case of this paper’s dataset, the higher the AUC, the better the model can distinguish between patients with benign tumors and those with malignant type.

Here is an analogy of AUC and ROC:



***AUC and ROC (X-axis: false positive rate; Y-axis: true positive rate ), towardsdatascience.com***

Where TPR and FPR are:

$$TPR = Recall = Sensetivity = \frac{True\ positive\ (TP)}{True\ positive\ (TP) + False\ negative\ (FN)}$$

$$FPR = 1 - Specificity = \frac{False\ positive\ (FP)}{True\ negative\ (TN) + False\ positive\ (FP)}$$

An excellent model has AUC near to 1, meaning great ability to differentiate classes or separability. And a model with AUC near 0 indicates a worse measure of separability.

## (0) Data Preprocessing

When dealing with any data science and data analytics problem, the first thing that should be considered is how clean and consistent the data is. Towards that end, in the beginning, the raw and messy data should be prepared and preprocessed for further analysis. This process is called data engineering, data preparation, or simply data cleaning. When cleaning the data, one should always look for the followings:

1. Missing values
2. Encoding categorical data
3. Encoding categorical labels
4. Splitting the data into train set and test set
5. Feature scaling

After taking care of the above points, the data will be prepared to be fed into the Machine Learning model.

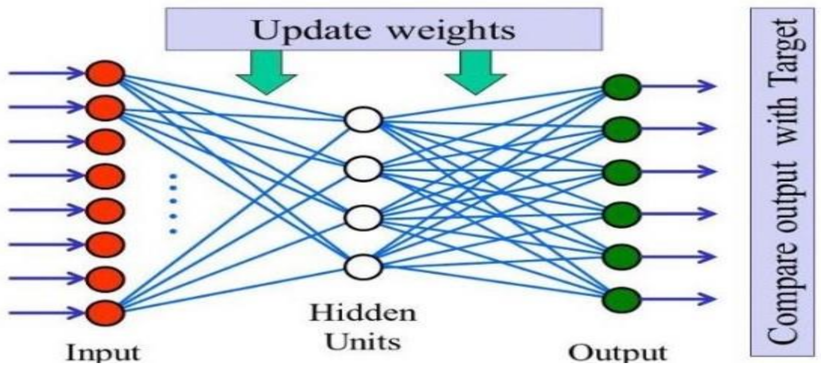
The required libraries are first imported. Second, the data is loaded and its features are separated from the labels. One important thing to mention is that the first column of the dataset, which is merely the ID number of the samples, was excluded from the features. Because this column is merely patients' samples' numbers, which does not contain any valuable information to base the model upon. Therefore, the second column up to the very last column were included in the features set. The last column of the dataset shows the classes, hence, it is included in the labels. Next, missing values were checked, but the function used to detect missing values showed there was none in the dataset. Nonetheless if there ever were any missing values, those rows could either be dropped or filled in using the previous samples' mean. But since the dataset has a sensitive medical application, if the dataset had few missing values, it might have been wiser to drop those rows. Because otherwise, false numbers would appear in the dataset by filling in the missing parts with the mean (or any other method). Next, the hold-out method was used to split the train data from the test data.  Therefore, 25% of the whole dataset was used as the test set and the remaining 75% as the train set. By now, the train set has shape (512, 9) and the test set has shape (171, 9). Accordingly, the train set has 512 labels, and the test set has 171 labels. The next step is feature scaling. Standardization was employed to rescale the data, rather than normalization, since the former will take care of the outliers far better than the latter. This scaler was employed to transform and fit to the train set. But, the scaler was only used to transform the test set. Basically, no scalers can fit to the test set since the model should have no prior knowledge of the test data. But fitting this scaler (and similarly any other scaler) to the test set will calculate its mean and variance; hence, extracts information from the test set (which is supposed to be unknown and new). The dataset is now ready to be fed into the model, which will be explained in the next section.

## (1) ANN (Neural Network) Classifier

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a dataset through a process that mimics how the human brain operates. Towards this end, it uses a set of inputs and assigns them initial weights and a total bias that will be updated later in the back-propagation stage. First, the inputs are multiplied by their weights, and then they will be fed into an activation function, aka a perceptron or a neuron. This layer is called the input layer. Then, the outputs can again go through similar layers, each getting multiplied by different weights. These middle layers are called hidden layers. At last, the values go into the output layer to yield the final output(s). This is the first epoch or iteration of the algorithm. This output is then compared to the true output(s), aka labels, and the relative error (For instance, minimum square error) is calculated. Then this error is back-propagated through the whole network to update the weights and biases. This process repeats all over again until either we reach a desired small error or until the weights converge to a specific constant value. If so, this will be the last epoch. This network is called an artificial neural network (ANN) because it mimics how the brain neurons work. By updating the weights, the network is basically learning the importance of each feature (feature extraction) and then fires it up/down if the input is fed to that neuron in the feature. The whole algorithm should run in several epochs for weights to converge or for the error to decrease. But we have to be aware of overfitting. If the epochs are too much or the model follows the train set too accurately, then the model will give a fair accuracy on the train set but a relatively low accuracy on the new test data. Meaning, the accuracy increase on the train data, but the generalization, which is the goal, decreases.

Moreover, we can use methods such as early stopping to combat overfitting. This method does not let the algorithm run the whole epochs if overfitting begins to happen. So this will reduce run time as well.

Furthermore, different optimization methods are used when dealing with neural networks. A very popular one is the Adam optimizer. We can take advantage of Stochastic gradient descent as well.



***Example of a Neural Network. groupfuturista.com***

In the implementation, after loading the dataset, since ANN model is used, label encoding on labels must be performed. The reason is that the classes are "2" and "4". These are categorical binary data, and the model will predict them as "0" and "1" later on. Hence, for the model to understand and analyze the labels easier and to prevent leakage or detection of unnecessary patterns between the class labels, encoding the label column is a necessary step. Encoding happens right after loading the dataset. Next, the data must be split into the train set and test set by 75% and 25%. After that, feature scaling is performed as the way it was mentioned in section (0) above. Next, the neural network model ANN was designed.

In Assignment one, experimental tuning was done on the ANN model in a trial-and-error way. But in Assignment two, grid search method is used to best tune the ANN hyper-parameters.

**In Assignment one** the following architecture was considered:

1. The first hidden layer containing 10 neurons and Rectified linear unit as the activation function
2. The second hidden layer, containing 10 neurons and Rectified linear unit as the activation function
3. The output layer, containing 1 neuron (since this is a binary classification problem) and Sigmoid activation function

Then the model was trained using Adam optimizer and binary cross entropy as the loss function within 20 epochs with the help of early stopping method to help combat overfitting of the model. This architecture showed a good accuracy of 95.9% compared to the other possible architectures that was tried by experience.

**In this Assignment two**, however, grid search method is used to automatically tune the model parameters, rather than by experience. The final results and comparisons between this method's output and Assignment one's, will be further discussed in section (IV) Results.

## (2) Applying the Grid Search Method

Here is a short description of Grid Search Method:

A model's hyper-parameter is a characteristic of the model that is pre-defined (before the learning process begins), unlike parameters that will be set and updated during the training process. Therefore, hyper-parameters are external to the model and their values cannot be estimated from the data. Hyper-parameters include: number of neurons, number of hidden layers, the number of nearest neighbors, etc.

In contrast, parameters are internal characteristic of the model and their value is estimated from the data. Parameters include: weights, biases, beta coefficients, etc.

Therefore, Grid Search is used to find the optimal value for the hyper-parameters of the model which results in the best and most precise predictions and metric evaluations.

The approach is to put desired values for the hyper-parameters in separate related lists, and then use each of them in a loop to train the model. For instance, if the goal is the check four hyper-parameters, each containing three different values, then the time complexity would be of order $3 \times 3 \times 3 \times 3 = 81$. Hence, it will usually take long to implement grid search method.

For each number of hidden layers in the ANN architecture, four different hyper-parameters are checked within nested loops. Therefore, in general, the following hyper-parameters are evaluated for the ANN model:

1. Number of hidden layers: 3, 4, 5
2. Number of neurons: 10, 30, 50
3. Activation functions: Rectified linear unit, Tangent hyperbolic, Sigmoid
4. Optimization functions: Adam, Stochastic gradient descent, Root mean square propagation
5. Number of epochs: 10, 20, 30

After each learning process is complete, the results are saved in a dictionary and appended to a list. By searching in the saved dictionaries (each dictionary corresponds to a unique set of hyper-parameters), the highest accuracy (or any other desired metrics, such as precision and ROC and AUC score) will be found. Therefore, for each number of specific hidden layers in the ANN model, there are the hyper-parameter values that will result in the highest metric of choice (accuracy, for example).

This process is applied to all the ANN models having different number of hidden layers (3, 4, 5). The results are shown in the (IV) Results Comparison section.

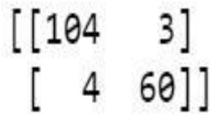
But before moving to the results, it is also important to mention why other models were not used, instead of the ANN model. ANN models have vast applications for classification problems. By using more hidden layers, one can usually increase the model's performance (without overfitting, indeed). On the other hand, ANN models might be hard to interpret by mathematical equations but they are equally easy to implement. These models use neurons and activation functions (in the form of Perceptrons) for each specific and vital feature. Therefore, ANN models can find hidden patters and relationships between the labels and the features. RNN models, on the other hand, are a little harder to implement and equally more time consuming to run. RNNs work best with time-series data. But here, the dataset was not dependent on time or any order. Therefore, RNN was not a smart choice for the dataset in this paper. CNNs are not also a wise choice, because they are mostly suitable for image data. The act of convolution and max pooling aims to extract highly detailed and specific features in higher dimensional data. But here, the dataset didn't qualify for such massive work. GAN models have a generator and a detector part, such as in the fake image detectors. The model is trained to become intelligent enough to detect nuances in the data by feeding in the slightly changed data to itself in the form of a feedback loop to both the generator and detector parts. Similarly, Autoencoders could not help us find whether a tumor is benign or malignant, since Autoencoders try to recreate the original data in the decoder section. These models, first, extract the most important features in the dataset and then having only these features, they try to recreate the data in the output. Therefore, Autoencoders are highly used in Telecommunications, noise removal, dimensionality reduction, and even recommendation systems. Moreover, the dataset was not large enough to be considered Big data, in order to take advantage of Hadoop file systems. We could also tune SVM or Decision tree classifiers, instead of the ANN classifier. But the goal in this assignment was to tune an ANN model. That being said, we can now head to the next section to see the results of the tuned ANN model compared to the un-tuned versions.

## (IV) Results Comparison

**In Assignment one**, the following results were drawn for each model without using a legitimate parameter tuning method, unless for an experimental one (which is not much reliable compared to a grid search):

1. **Decision Tree Classifier:**

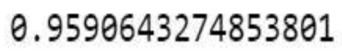
The confusion matrix is:



```
[[104   3]
 [  4  60]]
```

***Confusion matrix for the decision tree classifier***

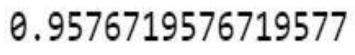
And the accuracy is:



```
0.9590643274853801
```

***Accuracy score of the decision tree classifier***

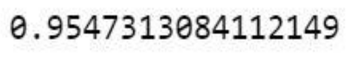
And the precision is:



```
0.9576719576719577
```

***Precision score of the decision tree classifier***

And the ROC_AUC score is:

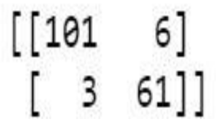


```
0.9547313084112149
```

***ROC_AUC score of the decision tree classifier***

2. **Kernel SVM (Support Vector Machine):**

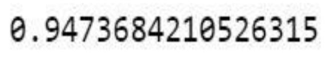
The confusion matrix is:



```
[[101   6]
 [  3  61]]
```

***Confusion matrix for Kernel SVM model***

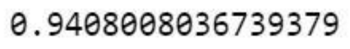
And the accuracy is:



```
0.9473684210526315
```

***Accuracy score for the Kernel SVM model***

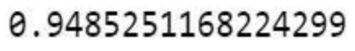
And the precision is:



```
0.9408008036739379
```

***Precision score for the Kernel SVM model***

And the ROC_AUC score is:

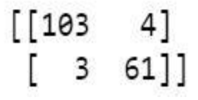


```
0.9485251168224299
```

***ROC_AUC score for the Kernel SVM model***

**3. ANN (Neural Network) Classifier:**

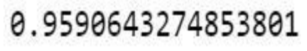
The confusion matrix is:



```
[[103   4]
 [  3  61]]
```

***Confusion matrix of the ANN model***

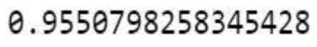
And the accuracy is:



```
0.9590643274853801
```

***Accuracy score of the ANN model***

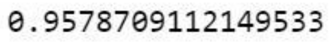
And the precision is:



```
0.9550798258345428
```

***Precision score of the ANN model***

And the ROC_AUC score is:



```
0.9578709112149533
```

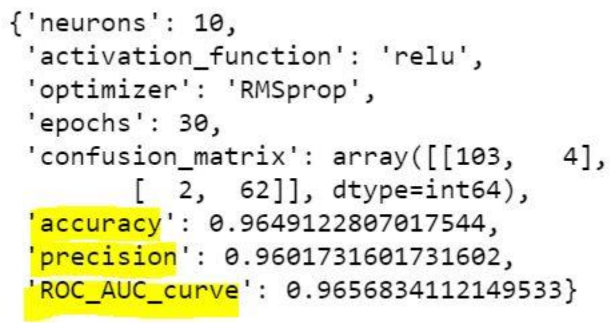
***ROC_AUC score of the ANN model***

Therefore, **in Assignment one**, it was concluded that the best models according to their performance metrics are: (in order)

1. ANN model
2. Decision tree model
3. Kernel SVM model

**In Assignment two**, the output of the tuned ANN model for each unique number of hidden layers in the model are as follows:

1. **ANN with Three Hidden Layers:**



```
{'neurons': 10,
 'activation_function': 'relu',
 'optimizer': 'RMSprop',
 'epochs': 30,
 'confusion_matrix': array([[103,   4],
        [  2,  62]], dtype=int64),
 'accuracy': 0.9649122807017544,
 'precision': 0.9601731601731602,
 'ROC_AUC_curve': 0.9656834112149533}
```

***Hyper-parameters and performance metrics for ANN model with three hidden layers***

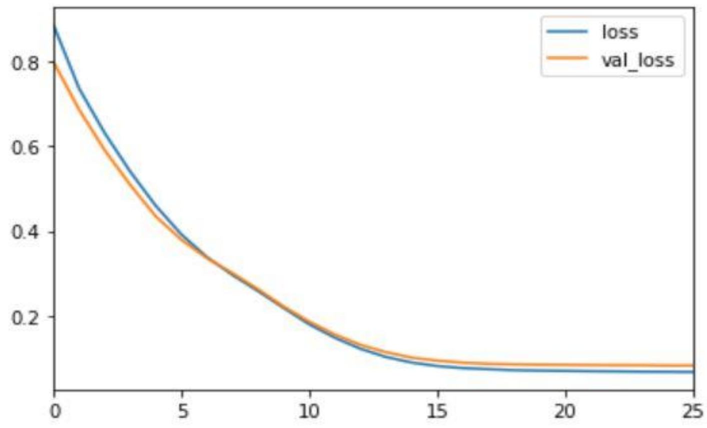
As can be seen above, the best performance that can be reached in the ANN model with only three hidden layers is when the model has 10 neurons, and Rectified linear unit as the activation function, and Root mean square propagation as the optimizer within 30 epochs.

The accuracy is about 96.49%.

The precision is about 96.01%.

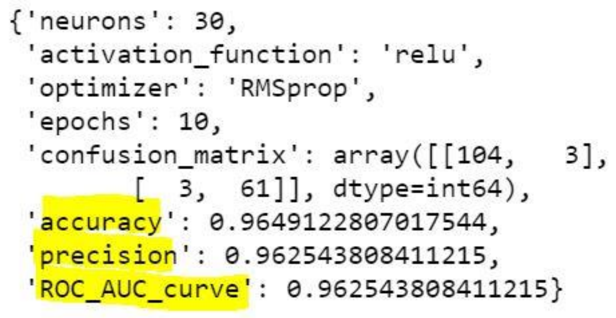
And the ROC_AUC score is about 96.56%.

As can be seen in the following loss function plot, no overfitting has occurred since the loss on the test set never increases. This was expected, for Early stopping method was used.



*Train loss and test loss in the best ANN model with three hidden layers*

## 2. ANN with Four Hidden Layers:



```
{'neurons': 30,
 'activation_function': 'relu',
 'optimizer': 'RMSprop',
 'epochs': 10,
 'confusion_matrix': array([[104,   3],
        [  3,  61]], dtype=int64),
 'accuracy': 0.9649122807017544,
 'precision': 0.962543808411215,
 'ROC_AUC_curve': 0.962543808411215}
```

*Hyper-parameters and performance metrics for ANN model with four hidden layers*

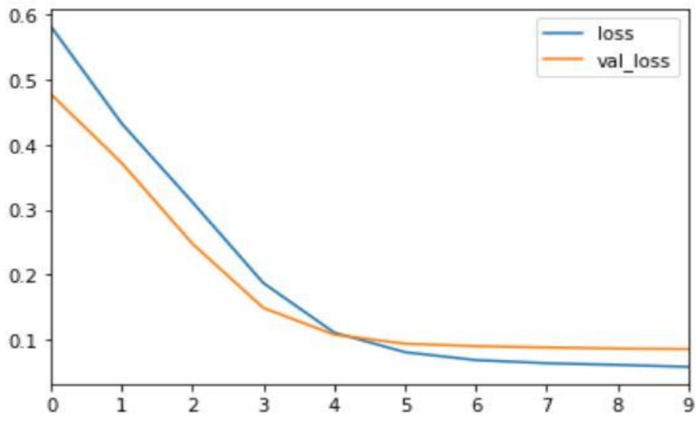
As can be seen above, the best performance that can be reached in the ANN model with only four hidden layers is when the model has 30 neurons, and Rectified linear unit as the activation function, and Root mean square propagation as the optimizer within 10 epochs.

The accuracy is about 96.49%.

The precision is about 96.25%.

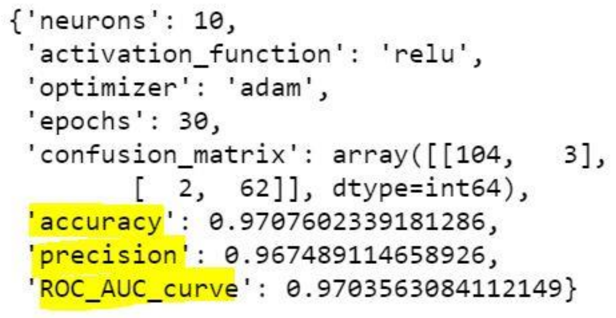
And the ROC_AUC score is about 96.25%.

As can be seen in the following loss function plot, no overfitting has occurred since the loss on the test set never increases. This was expected, for Early stopping method was used.



*Train loss and test loss in the best ANN model with four hidden layers*

## 3. ANN with Five Hidden Layers:



```
{'neurons': 10,
 'activation_function': 'relu',
 'optimizer': 'adam',
 'epochs': 30,
 'confusion_matrix': array([[104,   3],
        [  2,  62]], dtype=int64),
 'accuracy': 0.9707602339181286,
 'precision': 0.967489114658926,
 'ROC_AUC_curve': 0.9703563084112149}
```

*Hyper-parameters and performance metrics for ANN model with five hidden layers*

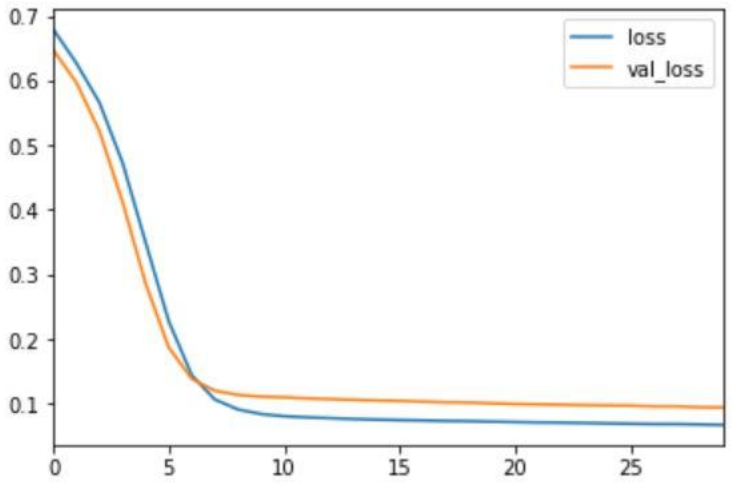
As can be seen above, the best performance that can be reached in the ANN model with only five hidden layers is when the model has 10 neurons, and Rectified linear unit as the activation function, and Adam as the optimizer within 30 epochs.

The accuracy is about 97.07%.

The precision is about 96.74%.

And the ROC_AUC score is about 97.03%.

As can be seen in the following loss function plot, no overfitting has occurred since the loss on the test set never increases. This was expected, for Early stopping method was used.



***Train loss and test loss in the best ANN model with five hidden layers***

As a result, according to the above outputs, the **tuned ANN model with five hidden layers** has the highest accuracy, precision and ROC-AUC score. Its loss function plot also decreases faster, meaning that the algorithm converges sooner.

Thus, after tuning the ANN model that was implemented in Assignment one, it is concluded that Assignment one's ANN model was not the best model after all. It only had 2 hidden layers, each having only 10 neurons that was trained through 20 epochs and its accuracy (95.90%), precision (95.78%) and recall (95.50%) were all less than the ANN model in Assignment two with 5 hidden layers, each having 10 neurons with the same optimizer (Adam) and activation function (Rectified linear unit) through 30 epochs. Therefore, in the tuned ANN model, more hidden layers and epochs are used. But the other hyper-parameters are similar to that of the un-tuned ANN model in Assignment one. Furthermore, the tuned ANN model with five hidden layers logically wins over the SVM and the Decision tree models implemented in Assignment one, since the even un-tuned ANN model also won over the SVM and the Decision tree models.