5/8/2020

Use of an MLP for prediction of Breast cancer

Advance machine learning

Word count 5000

Ismail bhamjee 16010047

COS 7045

Contents

Task A

The data set which was used consists of breast cancer prognostic data, this data is in numerical form and contains 569 records spanning 32 attributes. It was obtained from Kaggle (1) which is a reputable data repository. Furthermore, it had many papers relating to its content of which was an advantage for comparisons and previous works.
The data is rich in terms of items which are specific to breast growths which describe the cell nucelli these are obtained from computed digital image. Next, the split between was breast cancer masses (M) and fibrocystic breast masses(B). All of the attributes are numerical except for the label which nominal.

In the paper *"Computerized breast cancer diagnosis and prognosis from fine needle aspirates"* (2)
Their primary method was cross validation, this was done using their own multi surface tree method. This used linear programming to separate the samples and place them in their correct places
This report did not use an existing data set, instead they developed it using an image analysis software known as Xcyt from 569 patients. With this they also developed subsets for further extrapolation of information. This was done using image analysis

of 190 patients. which lead to a visual diagnosis accuracy of 90%. From this, samples were visually selected to prevent overlap.
They developed their own classification procedures which was a variation the MSM-tree. Which used linear separation this was done by 10-fold classification

Various works with this data.
In the paper "AN ANALYSIS OF THE METHODS EMPLOYED FOR BREAST CANCER DIAGNOSIS" the data set was used to try to diagnose breast cancer, in this they used an artificial neural network made in MATLAB. In this they tested various neural networks and machine learning tools along with a different selection of attributes to produce viable results which were that a RBFN with 16-10 attributes produced the highest accuracy of diagnosis ranging from 94.75% to 97.5% however no other metrics were given. Overall, this paper sort to make a NN and test it against other existing ones along with existing results.
(7)

The paper "An Evolutionary Artificial Neural Networks Approach for Breast Cancer Diagnosis" uses the Wisconsin data set to try yet again predict the diagnosis of breast cancer. However, in this paper an unseen ANN is used based on the pareto differential evolution algorithm. In this research they aim to reduce disagreement and inconsistences on mammographic interpretation. During testing they gained 98.12 % accuracy at 5100 epochs. With these results they compared with others which around the 94% mark. Overall, this paper found that the ANN used has better generalisation in comparison to other approaches along with lower computational costs. (8)

They are two fundamental ways this data was used. First to simply test an algorithm and to predict the tumour as seen by the latter two papers based on the numerical data set. In these two other methods were also compared to see how well the performance was. The other way was to extract the data using image processing tools to generate raw figures to help with diagnosis.

Question:
"The correct prediction of a malignant or benign tumour using previous records with the help of machine learning."
In order to answer this, we would need to run tests to see if we can correctly classify a tumour.
Next, we would need to tune them in order to try and get the best results with minimal errors.
With this we would expect there to be a range of value between the tests which could indicate what affects the classification.
Alongside this, a suitable algorithm or hyper parameters (such as batch size and learning rate) would be sought out through means of testing.
During so, multiple tests will be done to network in order to see which best affect decision making and classification.

Task B Data analysis

Problems with data.

Lack of training data is when the algorithm does not have an enough data to “learn” of. Hindering tasks such as prediction and classification.

Correct data representation. This is when you select the correct data to use for your algorithm for it to work on desired results. For example, if you were to distinguish from apples and oranges you would naturally remove all non-spherical objects to allow the model to make accurate predictions. It is essential to use a data set which is representative to the cases you want to predict. If the set is too small then there will be sampling noise, too big and sampling bias might occur.

Quality of data.

This is also a vital part of the process as your results are based on what you put in. If the data set contains outliers and errors then it will make the system unable to decide t patterns. Hence the process of cleaning the data is needed. 3 key aspects of data cleaning exist.

Data removal

If the data is not needed or does not provided any meaningful value then it should be removed at it make the rest of the system inefficient and inaccurate. This is a viable option for whole attributes as well as individual cases. Also
reduced dimensionality my help some algorithms perform better as well as using less computational resources

Data repairs.

If the data is missing and a logical approach can be found to “repair” the data then by all means it should be, this could be in the form of imputing from other records or even using the mean of the attribute if it seems appropriate

Data transformation.

Data may not always be in a state which it can be used, so transformation allows us to change the data into a more appropriate manner. For example, if I have money in word format then it would be appropriate to transform it into numerical. From numerical we can transform it into binary, if they are a limited number of values.

Problems with the machine learning algorithm itself

Over fitting.

Overfitting is when the model chosen is only able to perform well on the data set it was given and is unable to generalise on others. This often happens when the model is too complex relative to the noise of the data. To solve this, we can use algorithms with fewer attributes which will lead to fewer parameters. Furthermore, removing outliers with addition to more data for training will help.

Under fitting.

This is when the model is too simple and lacks suitable parameters for the task. Such as using linear regression on a data set which nonlinear. However, this can be quickly soled using a better model with a more suitable set of parameters along with reducing constraints on a model.

Testing and validating to see if our model work

A huge part of machine learning is to see if the model we have chosen works and is able to predict new records. A way of doing this is to split the data set into a training set and a testing set. This split is dependent on the quantity of data in the set usually around 90% to 10%. By looking at how well the algorithm performs on the test set, an out of sample error (generalisation error) is known. Which shows how well the model performs on unknown instances.

Data analysis of our data.

Firstly, our data comprises of 32 columns with 570 records. This is the shape of our data. Next we can see that our data contains mostly numerical values except for the “diagnosis” column. This has a nominal output of either M for malignant or B for benign. Next, we can see the independent ID column which is an integer. Lastly, we have the attributes of the growths.
These are in the form of a float.

Now we have seen what our data looks like we have to look for potential problems with our data. These could be in the form of miss typed numbers, letter appearing as well as much more. Away to see this is to use method “DF.datatypes”. This will display the type of variable in the column and from there we can see if it has any discrepancies.

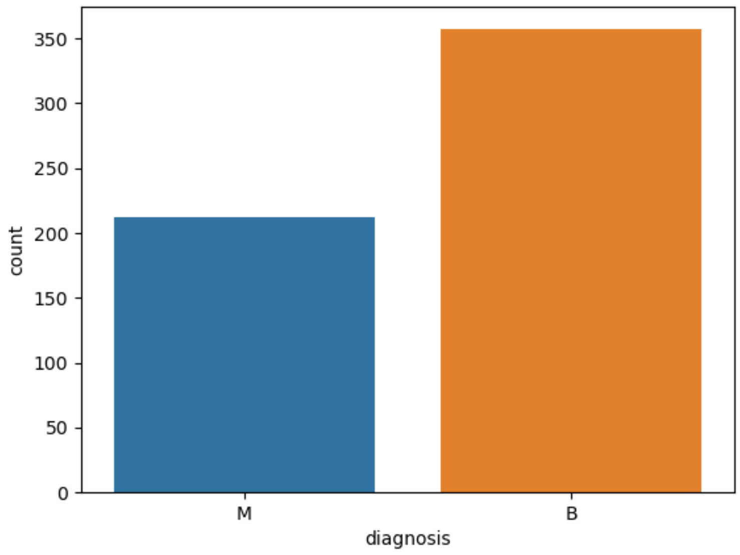
Problems with our data

Our data was well constructed and contained no null values after it had been tested via python pandas. Next, we did find an unknown32 column which was removed due to it having no real benefit, we also removed the independent ID column as it did nothing. As the data was only 570 columns, we could see visually that they were no substantial problems. This was confirmed by the data types method which returned floats along every attribute and a string for the label.

Data analysis using python.

Python has many libraries such as pandas and matplotlib which allow us to visualise our data in a form which can be easily interpreted by the reader. Next, we can use it to clearly see trends in the data without having to perform specific tasks. This may be used to see the effect of one attribute or to see the effect of many in relation to each other. Lastly it is easier than looking at hundreds or thousands of lines on a spreadsheet.

Label analysis



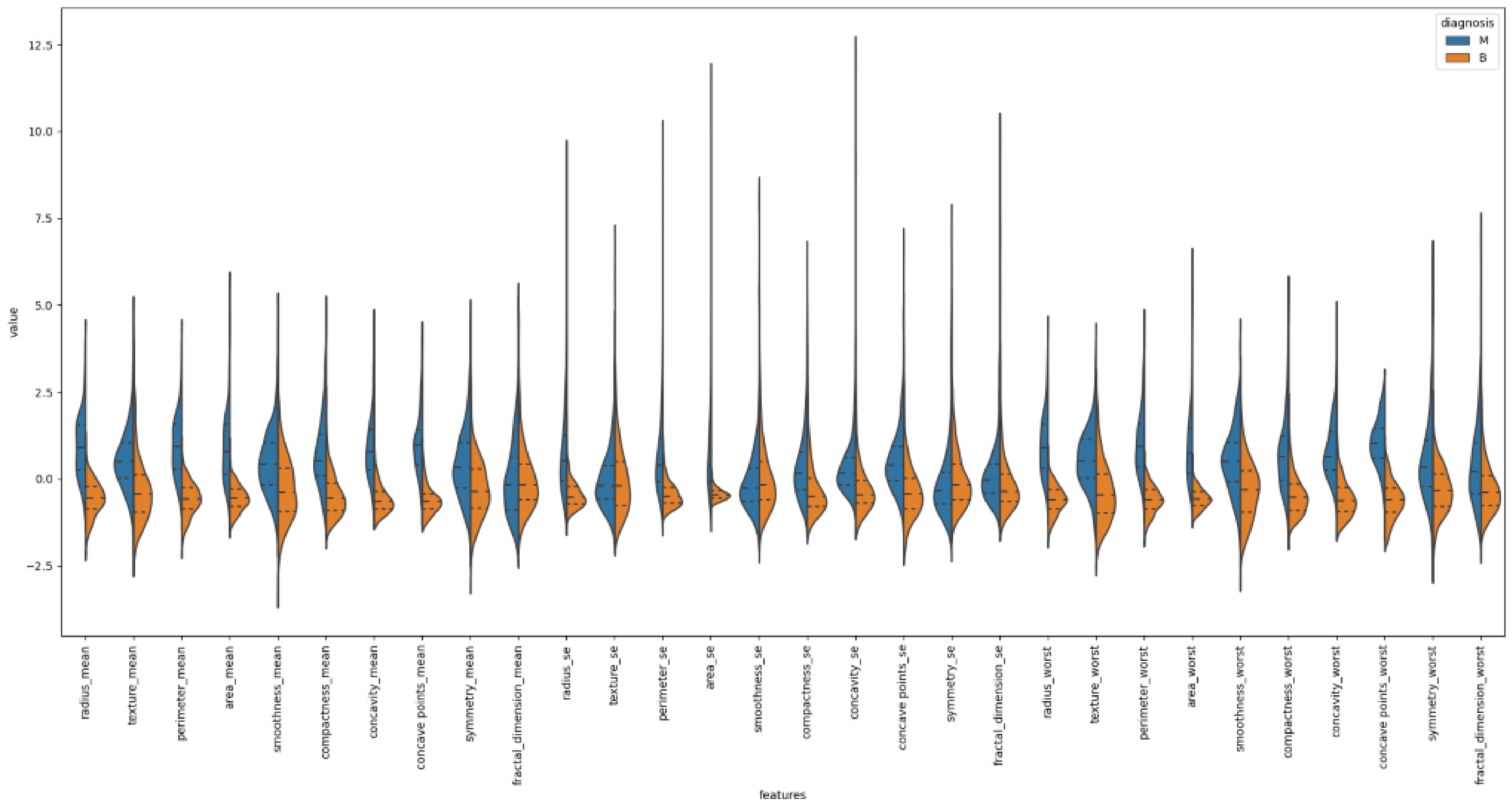
In this visualisation we can see diagnosis which is the label. In this we can see the amount of malignant cancers diagnosis against the number of benign diagnostics. As we can see we can see with the addition of a few pre-existing facts that the number of malignant diagnosis is roughly 1/3 of all.

Metrics

	Texture mean	Area mean	Concavity mean	Area se	Concavity se	Smoothness worst	Concavity worst	Symmetry worst
min	9.71	143.5	0	6.802	0	0.071	0	0.159
max	39.28	2501	0.427	542.2	0.396	0.223	1.252	0.664
mean	19.29	654.89	0.089	40.337	0.032	0.132	0.272	0.29
Std dev	4.301	351.9	0.8	45.491	0.03	0.023	0.209	0.062

Next, we can see using. Describe method in python to allow is to see what our data is truly like and whether it is numerically suitable for us or it harbours some type of outliers. Furthermore, we can see important metrics such as standard deviation and mean which will be helpful in data analysis.

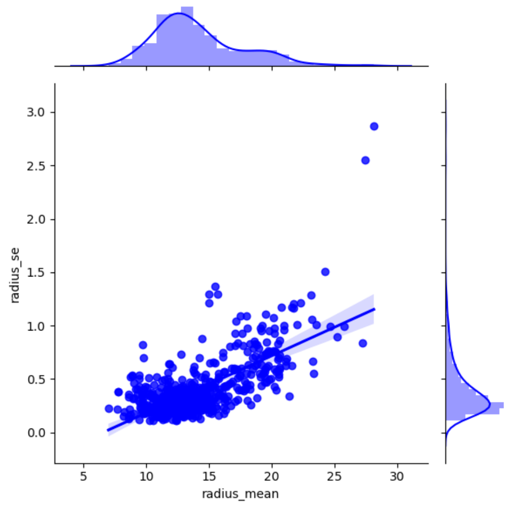
Violin plots



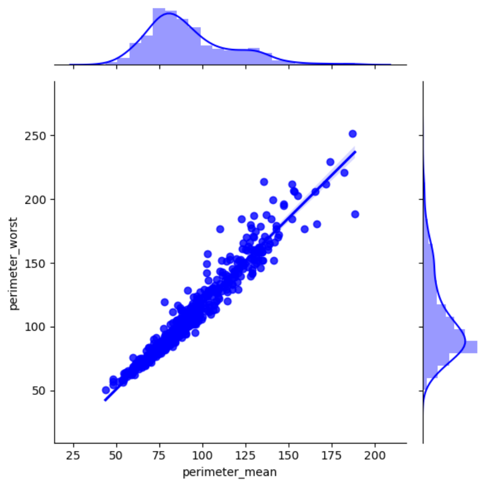
diagnosis
M
B
value
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave points_se
symmetry_se
fractal_dimension_se
radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst
concavity_worst
concave points_worst
symmetry_worst
fractal_dimension_worst
features

Here we have a number of violin plots to show what the relationship between malignant and benign tumours are between a single attribute. For example, we can see concave points worst and texture mean are shown to be separate from each other which could indicate a relationship between each other in further classification. Next ones which are closing resembling each other such as fractal dimensions and smoothness _se would pose a problem in seeing if that attribute leads to a different result.

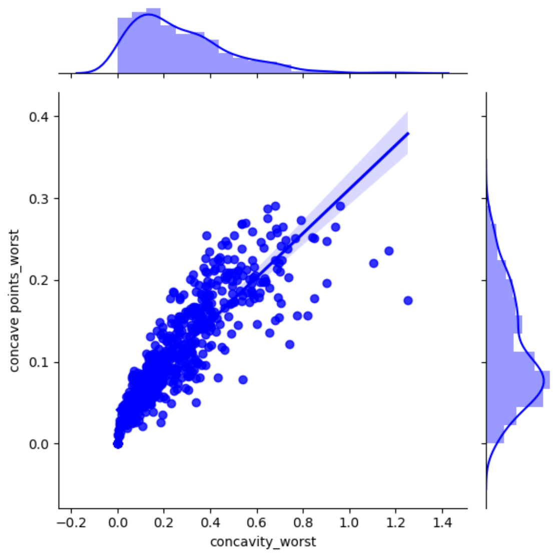
Joint plots



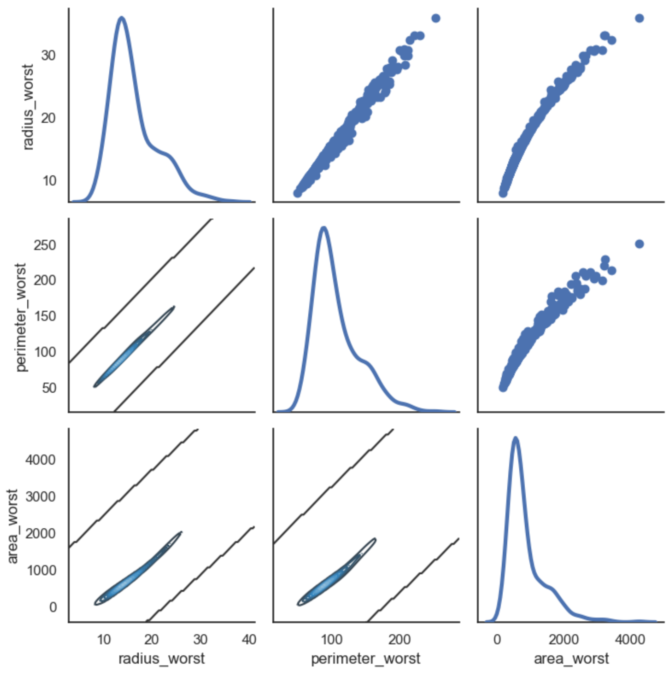
This joint plot shows that radius _se and radius mean are not highly correlated by the spread, as it is sporadic.



This plot shows that the correlation between perimeter_worst and perimeter mean is a strong positive one with the low spread and the distribution theme.



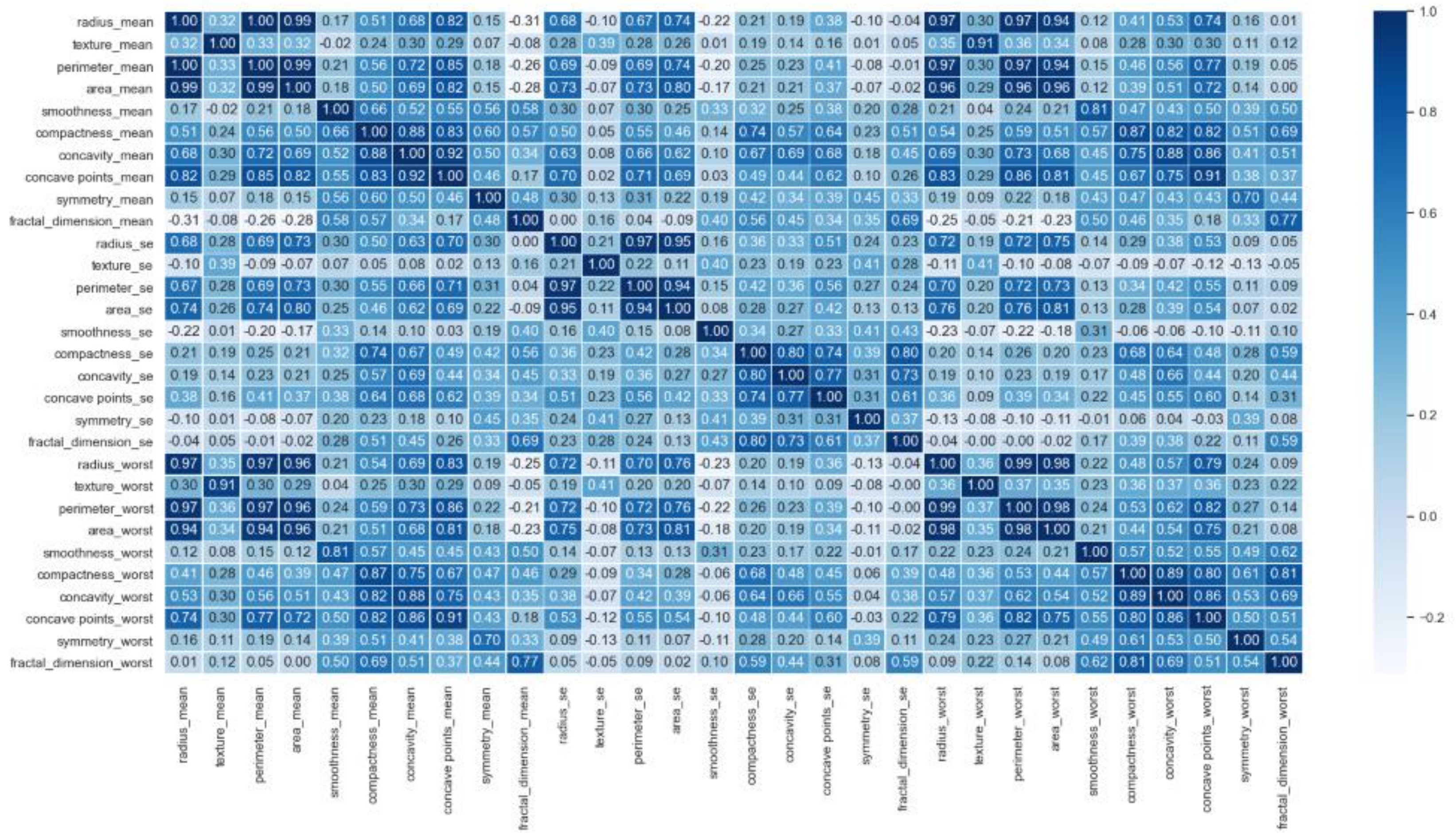
This plot shows the correlation between concavity worst and concave points worst. As we can see the overarching trend between points.



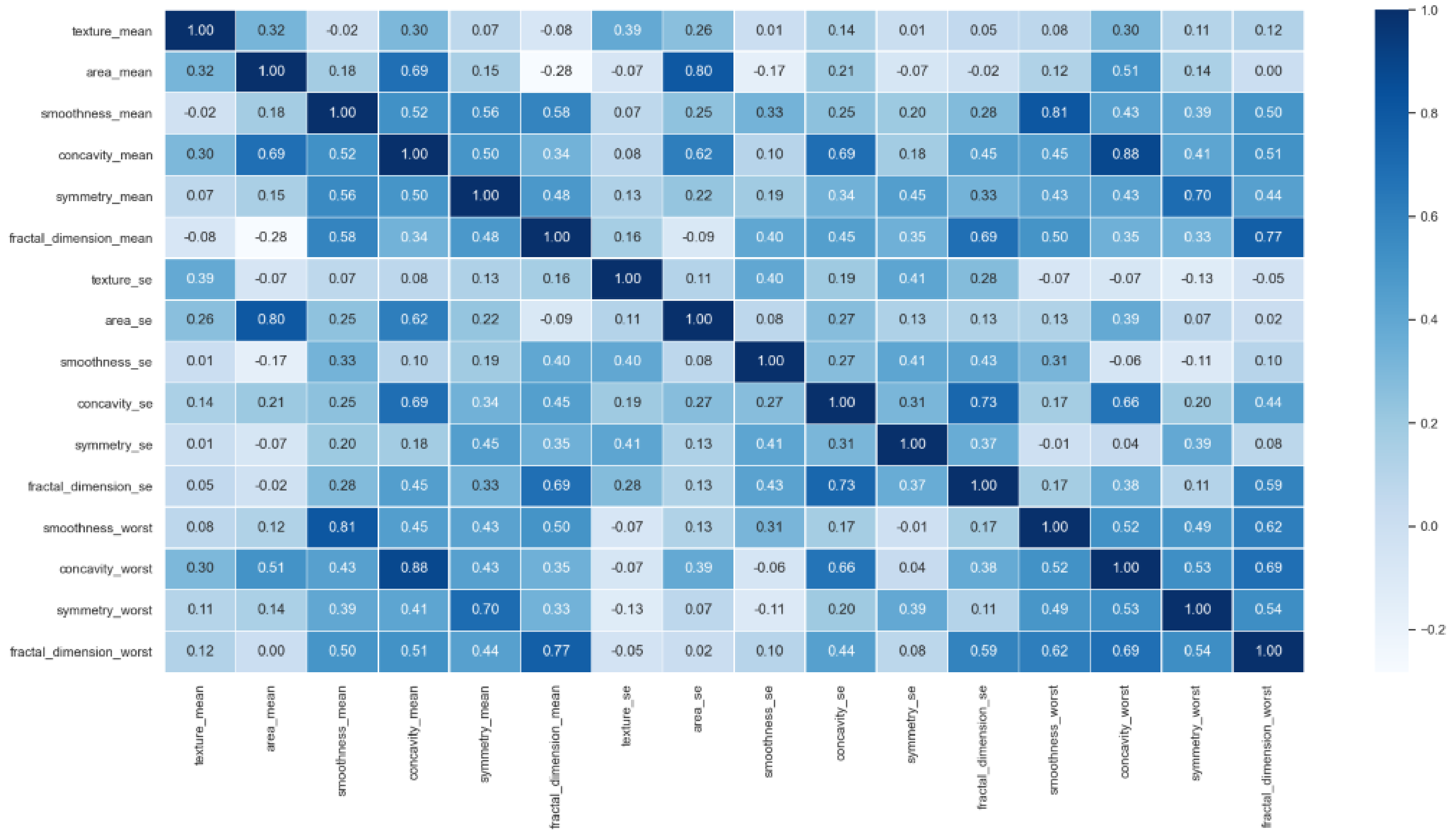
Pairs plot

A pairs plot is useful to analyse 3 or more attributes. We can see that the 3 given attributes are pretty similar to each other. This would be use full in feature selection as near identical attributes can be removed as they could be problematic

correlation coefficient



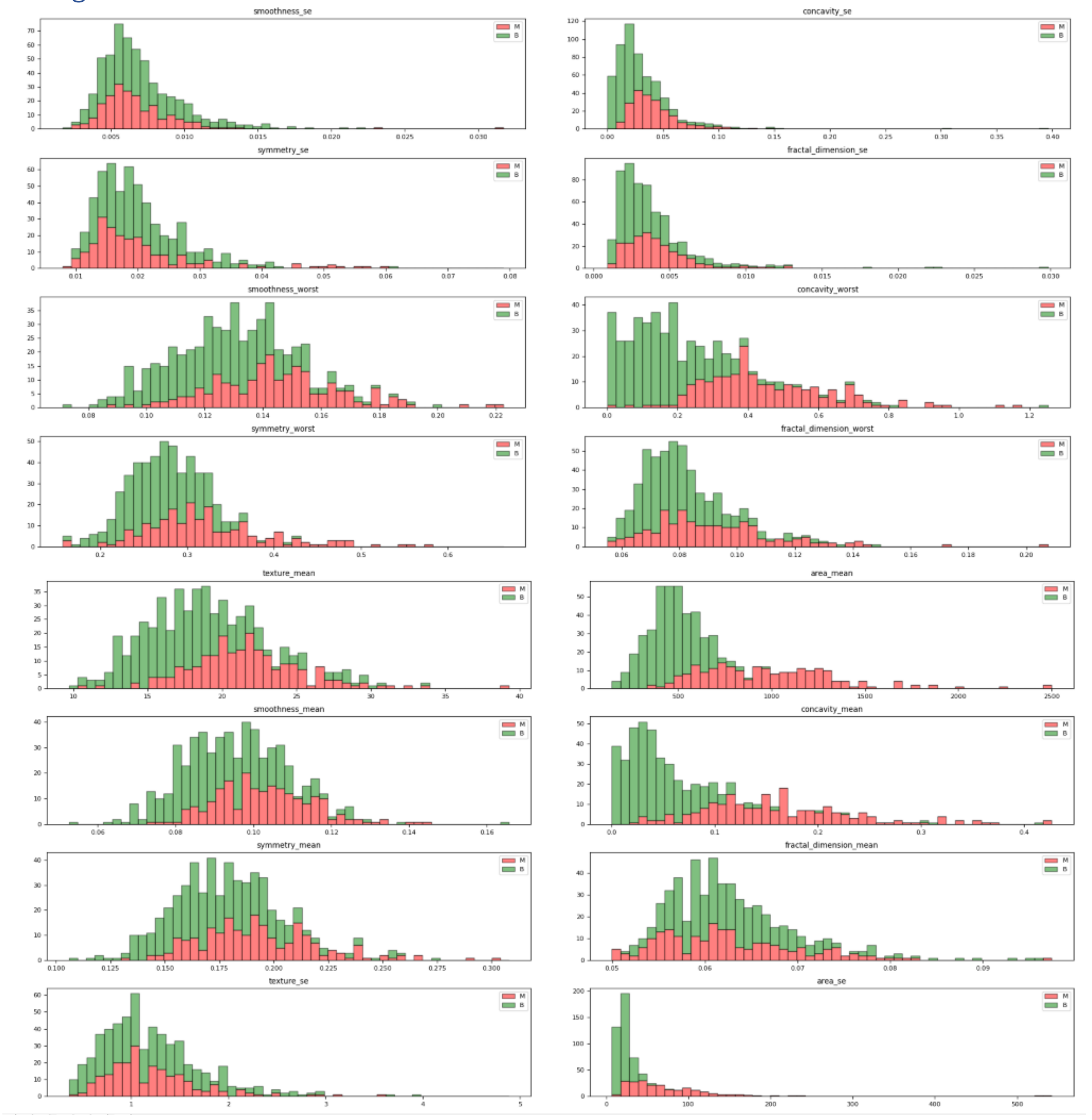
Next, we used a correlation coefficient to see if the attributes were closely correlated, this would allow us to predict one variable from another. With this early relationship between attributes and the chosen label we identified, this would hopefully lead to better processing of the data in upcoming tasks such as classification. In this matrix we used Pearson’s correlation coefficient.



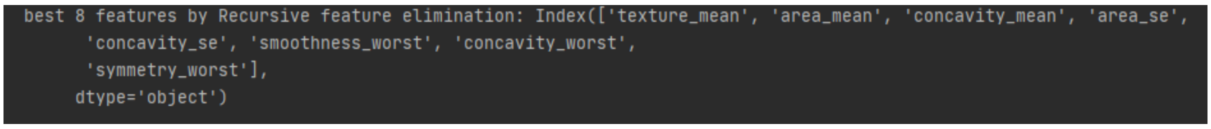
	texture_mean	area_mean	smoothness_mean	concavity_mean	symmetry_mean	fractal_dimension_mean	texture_se	area_se	smoothness_se	concavity_se	symmetry_se	fractal_dimension_se	smoothness_worst	concavity_worst	symmetry_worst	fractal_dimension_worst
texture_mean	1.00	0.32	-0.02	0.30	0.07	-0.08	0.39	0.26	0.01	0.14	0.01	0.05	0.08	0.30	0.11	0.12
area_mean	0.32	1.00	0.18	0.69	0.15	-0.28	-0.07	0.80	-0.17	0.21	-0.07	-0.02	0.12	0.51	0.14	0.00
smoothness_mean	-0.02	0.18	1.00	0.52	0.56	0.58	0.07	0.25	0.33	0.25	0.20	0.28	0.81	0.43	0.39	0.50
concavity_mean	0.30	0.69	0.52	1.00	0.50	0.34	0.08	0.62	0.10	0.69	0.18	0.45	0.45	0.88	0.41	0.51
symmetry_mean	0.07	0.15	0.56	0.50	1.00	0.48	0.13	0.22	0.19	0.34	0.45	0.33	0.43	0.43	0.70	0.44
fractal_dimension_mean	-0.08	-0.28	0.58	0.34	0.48	1.00	0.16	-0.09	0.40	0.45	0.35	0.69	0.50	0.35	0.33	0.77
texture_se	0.39	-0.07	0.07	0.08	0.13	0.16	1.00	0.11	0.40	0.19	0.41	0.28	-0.07	-0.07	-0.13	-0.05
area_se	0.26	0.80	0.25	0.62	0.22	-0.09	0.11	1.00	0.08	0.27	0.13	0.13	0.13	0.39	0.07	0.02
smoothness_se	0.01	-0.17	0.33	0.10	0.19	0.40	0.40	0.08	1.00	0.27	0.41	0.43	0.31	-0.06	-0.11	0.10
concavity_se	0.14	0.21	0.25	0.69	0.34	0.45	0.19	0.27	0.27	1.00	0.31	0.73	0.17	0.66	0.20	0.44
symmetry_se	0.01	-0.07	0.20	0.18	0.45	0.35	0.41	0.13	0.41	0.31	1.00	0.37	-0.01	0.04	0.39	0.08
fractal_dimension_se	0.05	-0.02	0.28	0.45	0.33	0.69	0.28	0.13	0.43	0.73	0.37	1.00	0.17	0.38	0.11	0.59
smoothness_worst	0.08	0.12	0.81	0.45	0.43	0.50	-0.07	0.13	0.31	0.17	-0.01	0.17	1.00	0.52	0.49	0.62
concavity_worst	0.30	0.51	0.43	0.88	0.43	0.35	-0.07	0.39	-0.06	0.66	0.04	0.38	0.52	1.00	0.53	0.69
symmetry_worst	0.11	0.14	0.39	0.41	0.70	0.33	-0.13	0.07	-0.11	0.20	0.39	0.11	0.49	0.53	1.00	0.54
fractal_dimension_worst	0.12	0.00	0.50	0.51	0.44	0.77	-0.05	0.02	0.10	0.44	0.08	0.59	0.62	0.69	0.54	1.00

In this step with narrowed down the 16 most correlated values with the label and this is what was given.

Histograms



Next from the reduced data set histograms showing two groups against each other were made to see the actual difference in values against diagnosis. For example, concavity mean. We can see that the vast majority between of cases between 0.0 and 0.1 are benign and as the get considerably larger more of these cases are malignant. This is also seen in area mean where practically zero benign cancers exist outside the 1000 range.



```
best 8 features by Recursive feature elimination: Index(['texture_mean', 'area_mean', 'concavity_mean', 'area_se',
       'concavity_se', 'smoothness_worst', 'concavity_worst',
       'symmetry_worst'],
      dtype='object')
```

Lastly, we used recursive feature elimination do find out the 8 most correlated attributes with the label. This will be used in the classification later on.

Conclusion of data analysis

This process was used to learn about the data set through the use of visuals, in this we found out without Appling any models how the data was and the good aspects such as which were correlated with the label and the bad such as once which were redundant. However, coloration does not necessarily mean causation. Next applying feature reduction allows to use the best possible data attributes for the upcoming classification task. Lastly it would have been better to visualise all attributes against each other, but this would have meant a lengthy document.

Task C practical work

Use of weka

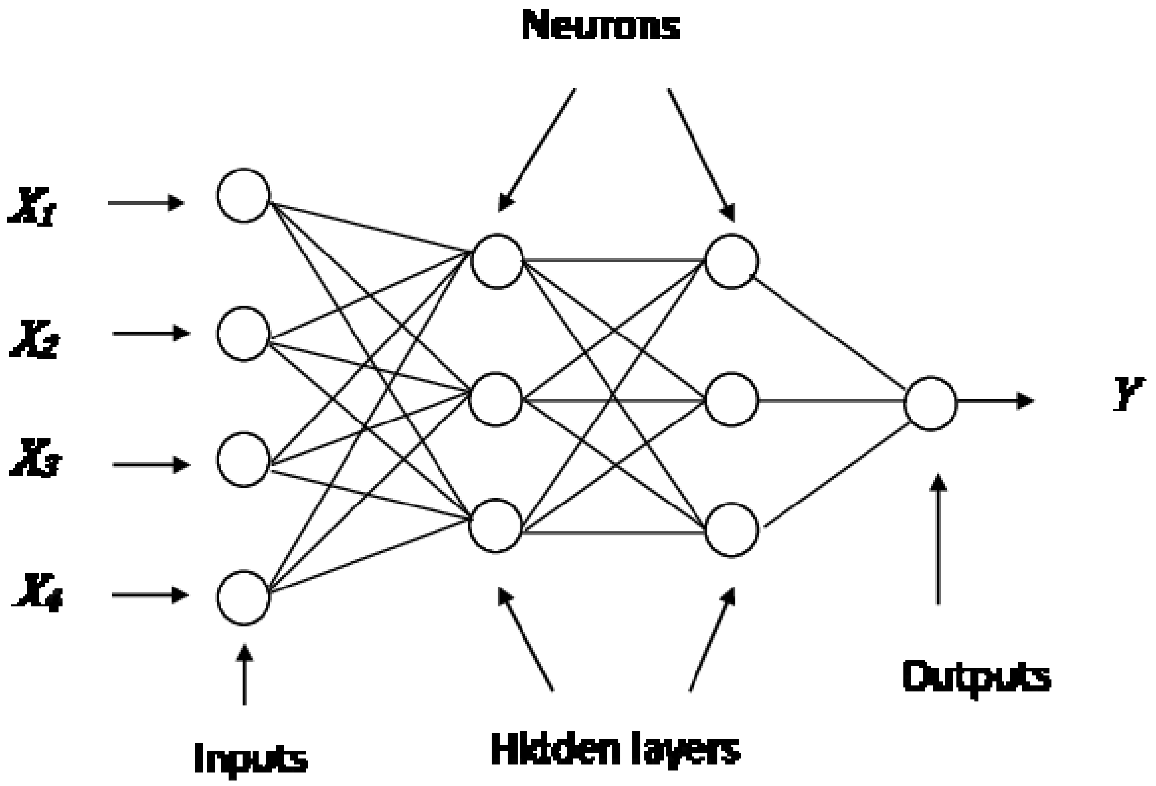
In order to achieve our goal of predicting whether a tumour is M or B we would need to apply machine learning algorithms for this. A way in which this could be done is to make them from scratch using either the python frameworks or the R frameworks. This would allow us total control of what is happening to our algorithms. however, these can take time to create and often only be good for one specific algorithm for example a random Forrest.

In order to bypass the need to create specific algorithms, weka can be used as it a tool that contains many machine learning algorithms for either clustering or classification. Furthermore, it allows for full control of the algorithm in terms of input such as the attributes selected, the process such as the split on training and testing data and the output in terms of suitably producing all the key results such as MAE and RMSE

Next weka has suitable visualisation tools for its applications such as being able to display a neural network or a random Forrest. Lastly weka allows us to change certain parameters for our desired algorithm such as learning rate. (5)

Learning algorithm

As we are choosing a supervised learning task, we would require an algorithm which would correctly classify our instances with the given previous data. For this task we will be using a multi layered perceptron. This is a powerful network which allows binary label classification to be performed. Furthermore, they can be applied to many types of data, for example image processing. Next the adaptive learning via weight change makes it viable option this leads to handling of imprecise data better. Lastly, they do not make assumptions regarding underlying probability destiny functions such as Bayesian linear regression.



Topology of a MLP
(6)
A MLP consists of an input layer, one or many hidden layers and an output layer. This algorithm handles mini batches and goes through the full training data set multiple times these are called epochs.

Each batch is given to the input layer which sends it to the input layer. Next the algorithm computes the sum of all the neurons in this layer. This is than passed onto the next layer and the sum is produced, also known as the forward pass.

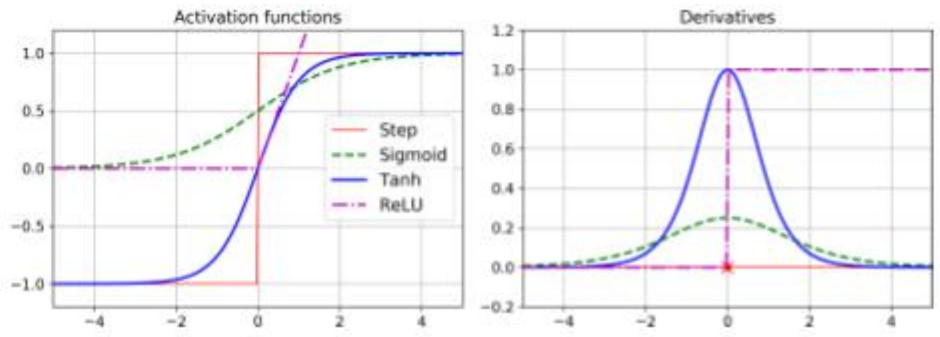
This occurs until the output layer and along the way it is making predictions, next as all the intermediate result are kept due to being needed for the backwards pass. Furthermore, an output error is measured this is done via a loss function to compare the given result to the desired output.

With the results from each layer and the error generated by each full epoch. The algorithm can compute how much error is given by each output. This is done via the chain rule. lastly the algorithm works backwards layer by layer to the input layer figuring out which of the connections cause the greatest error.

Lastly gradient decent is performed to modify the connection weights in the network using the errors it computed and stored. As for the weight they are randomly assigned as they are used “learning”. For example, if they are the same then the neurons in the hidden layer will be identical and this would result in backwards propagation to affect them in the same way rendering it useless.

Activation function

The sum of the weighted inputs is passed to an activation function, which processes the input and gives an output based on the function itself. They are many different functions such as ReLU function, this is continuous but as z equals 0 it is not differentiable. Next this function is widely used for its compute time and its lack of maximum output value. primarily they are two types of functions linear and nonlinear. Afore mentioned in a linear function and an example of a nonlinear would be the sigmoid function



Various activation functions and their derivatives
(4)

Hyper parameters

Learning rate
This controls the rate in which the neural network is updated. It is times the estimated weight error. this controls the rate of which the model learns through the hidden layers. A larger learning rate causes the model to learn faster at the cost of fining the optimal set of weights. A lower learning rate would mean the model will more likely to find optimal set of weights but the time taken could greatly increase. This goes without saying that the best learning rate for a model would be found with trial and error.

Batch size

Batch size affects the learning estimate and time taken for an iteration. A smaller batch size would mean a smaller iteration and less precise result. A larger batch would mean more precise result for the gradient and a larger time taken for the iteration.

Momentum

The use of history on a weigh update can be useful as it introduces inertia to the procedure, this cause change to head in one direction and make future updates easier resulting in less iterations.
Now that we have seen how a Multi layered perceptron work and we have established the parameters which can affect it. We will see what it produces in terms of results

Metrics

RMSE

Root mean square error this tells us the average of the
squared differences between prediction and actual observation. This also means that larger errors have a higher weight. This method uses the Euclidian distance between instances (A straight path).

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

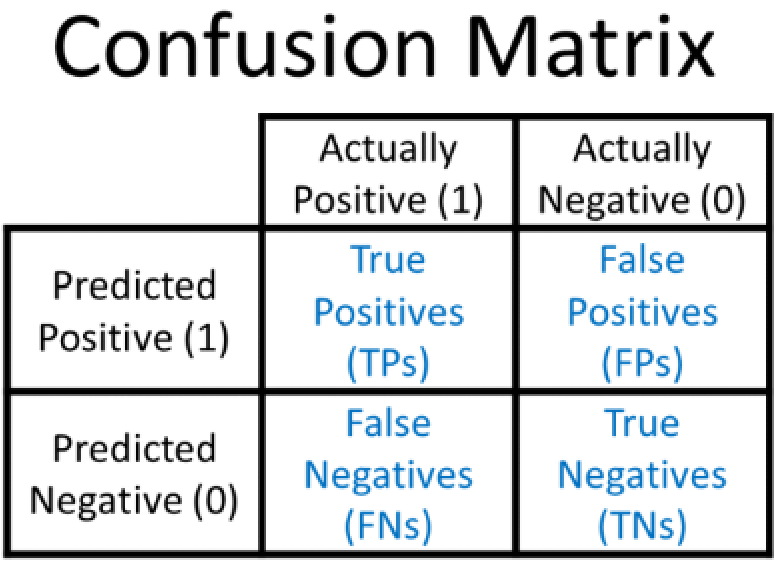
MAE

This is the average over the given test sample of
the absolute differences between actual observation and the prediction. Given
all individual differences have equal weight. This method uses the Manhattan distance between instances (given path along the designated routes). With both measures the lower the better.

$$\mathrm{MAE}(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m}\left|h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right|$$

Confusion matrix

An alternative way to evaluate the performance of a classifier is to look at the confusion matrix. This is done by giving the model a set of predictions so the model can compare between them.
Each row of the confusion matrix represents a class and
each column represents a predicted class.



Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

- True positive: positive instances that were **correctly** classified as **positive.**
- False positive: negative examples which were **incorrectly** classified as **positive.**
- Also known as type 1 error.
- False negative: positive instances which were **incorrectly** classified at **negative**
- Also known as type 2 error.
- True negative: negative examples which were **correctly** classified as **negative**
- They are many derivatives which can be made by this matrix these are the two we are going to use for our testing.
- Sensitivity or recall rate is measured by. sensitivity = TP/TP+FN.
- This is the total correctly predicted positives.
- Specificity rate is measured by. Specificity = TN/TN+FP
- This is how exact the assignment is to the positive class
- Accuracy rate is TP+TN/TP+TN+FP+F.
- This tells us how likely the model is to make a correct prediction.

Testing

A lot of pre-processing was done via python's data manipulation libraries. In this the optimal attributes were found using Pearson's correlation coefficient. The

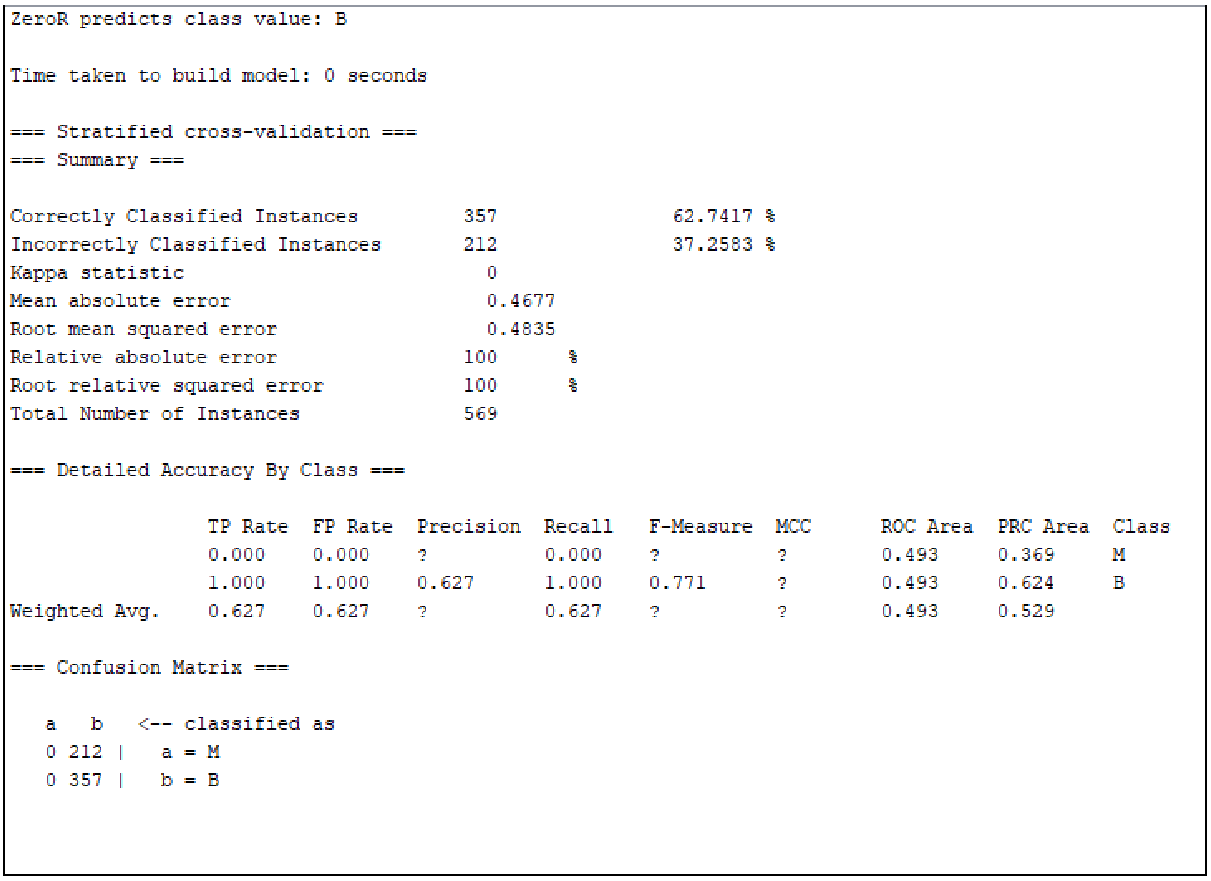
First, we will see if the size of the training set and testing set affects the MLP ability to predict the correct instance. For this all the remaining hyperparameters will be untouched to ensure a fair test.

The next test would be to test the various hyper parameters to ensure the optimal learning process.

Now that we have described what we are looking for we can begin testing. As we are only using the MLP we will have to change certain parameters to look at our model under different values.

Test 0- baseline

Baseline zeroR				
Test number	Batch size	Hidden layers	Learning rate	Training
0	100	a	0.3	500

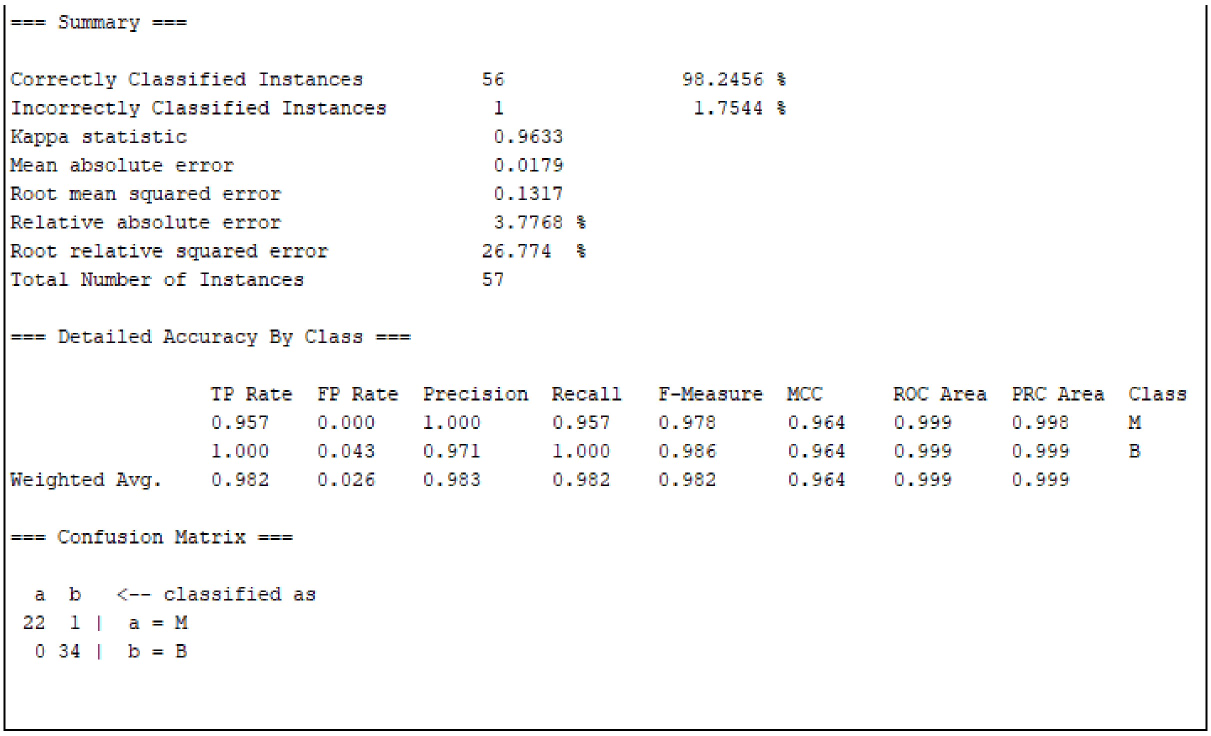


```
ZeroR predicts class value: B

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         357               62.7417 %
Incorrectly Classified Instances       212               37.2583 %
Kappa statistic                          0
Mean absolute error                      0.4677
Root mean squared error                  0.4835
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances              569

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.000    0.000    ?          0.000    ?          ?        0.493     0.369     M
                 1.000    1.000    0.627      1.000    0.771      ?        0.493     0.624     B
Weighted Avg.    0.627    0.627    ?          0.627    ?          ?        0.493     0.529

=== Confusion Matrix ===

   a   b   <-- classified as
   0 212 |   a = M
   0 357 |   b = B
```

Test 1 percentage split

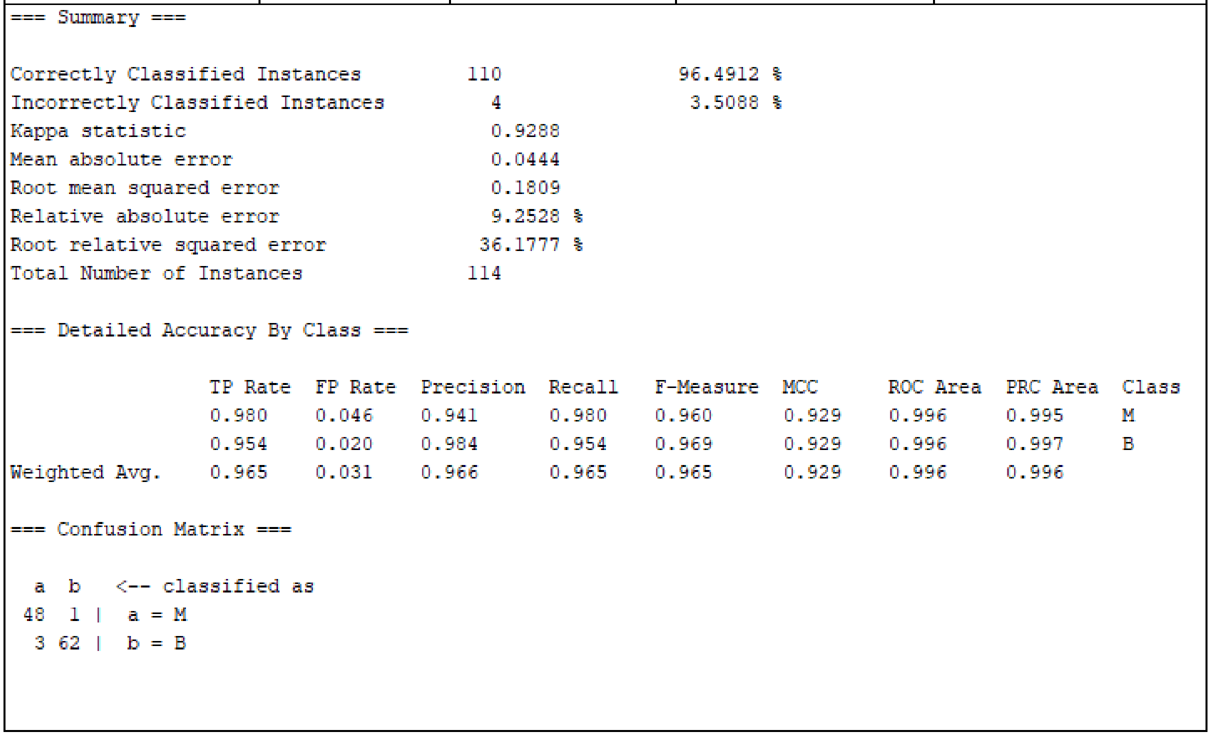
<u>The first set of tests will observe if changing the percentage split between of training and testing data will affect its output.</u>

Percentage split 90/10				
Test number	Batch size	Hidden layers	Learning rate	Training
1.1	100	a	0.3	513



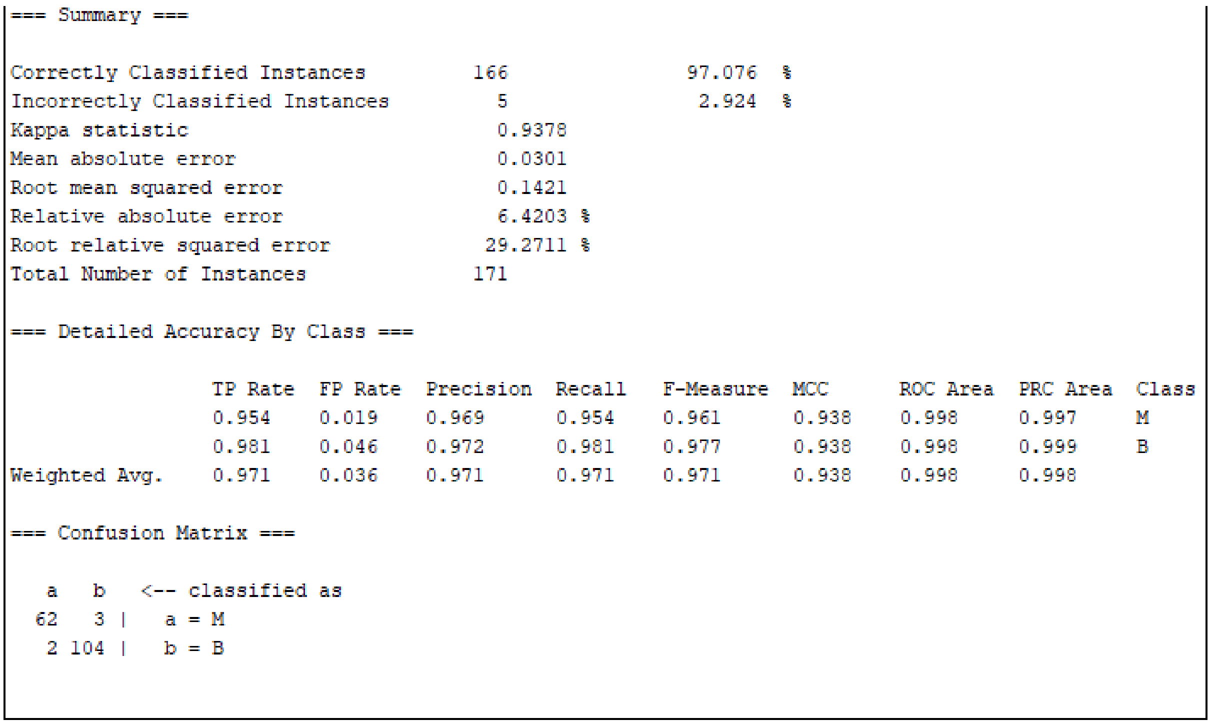
```
=== Summary ===

Correctly Classified Instances          56               98.2456 %
Incorrectly Classified Instances         1                1.7544 %
Kappa statistic                          0.9633
Mean absolute error                      0.0179
Root mean squared error                  0.1317
Relative absolute error                  3.7768 %
Root relative squared error             26.774  %
Total Number of Instances               57

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.957    0.000    1.000      0.957    0.978      0.964    0.999     0.998     M
                 1.000    0.043    0.971      1.000    0.986      0.964    0.999     0.999     B
Weighted Avg.    0.982    0.026    0.983      0.982    0.982      0.964    0.999     0.999

=== Confusion Matrix ===

  a  b   <-- classified as
 22  1 |  a = M
  0 34 |  b = B
```

Percentage split 80/20				
Test number	Batch size	Hidden layers	Learning rate	Training
1.2	100	a	0.3	456



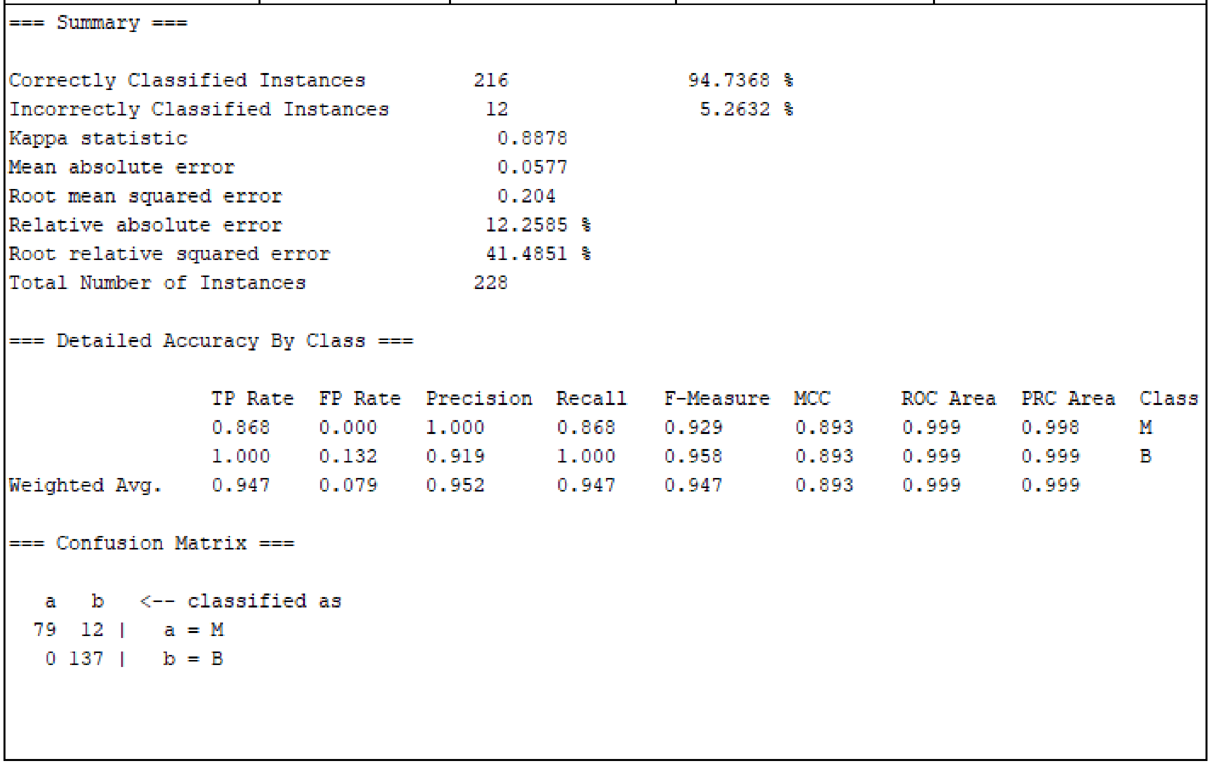
```
=== Summary ===

Correctly Classified Instances         110               96.4912 %
Incorrectly Classified Instances         4                3.5088 %
Kappa statistic                          0.9288
Mean absolute error                      0.0444
Root mean squared error                  0.1809
Relative absolute error                  9.2528 %
Root relative squared error             36.1777 %
Total Number of Instances              114

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.980    0.046    0.941      0.980    0.960      0.929    0.996     0.995     M
                 0.954    0.020    0.984      0.954    0.969      0.929    0.996     0.997     B
Weighted Avg.    0.965    0.031    0.966      0.965    0.965      0.929    0.996     0.996

=== Confusion Matrix ===

  a  b   <-- classified as
 48  1 |  a = M
  3 62 |  b = B
```

Percentage split70/30				
Test number	Batch size	Hidden layers	Learning rate	Training
1.3	100	a	0.3	399



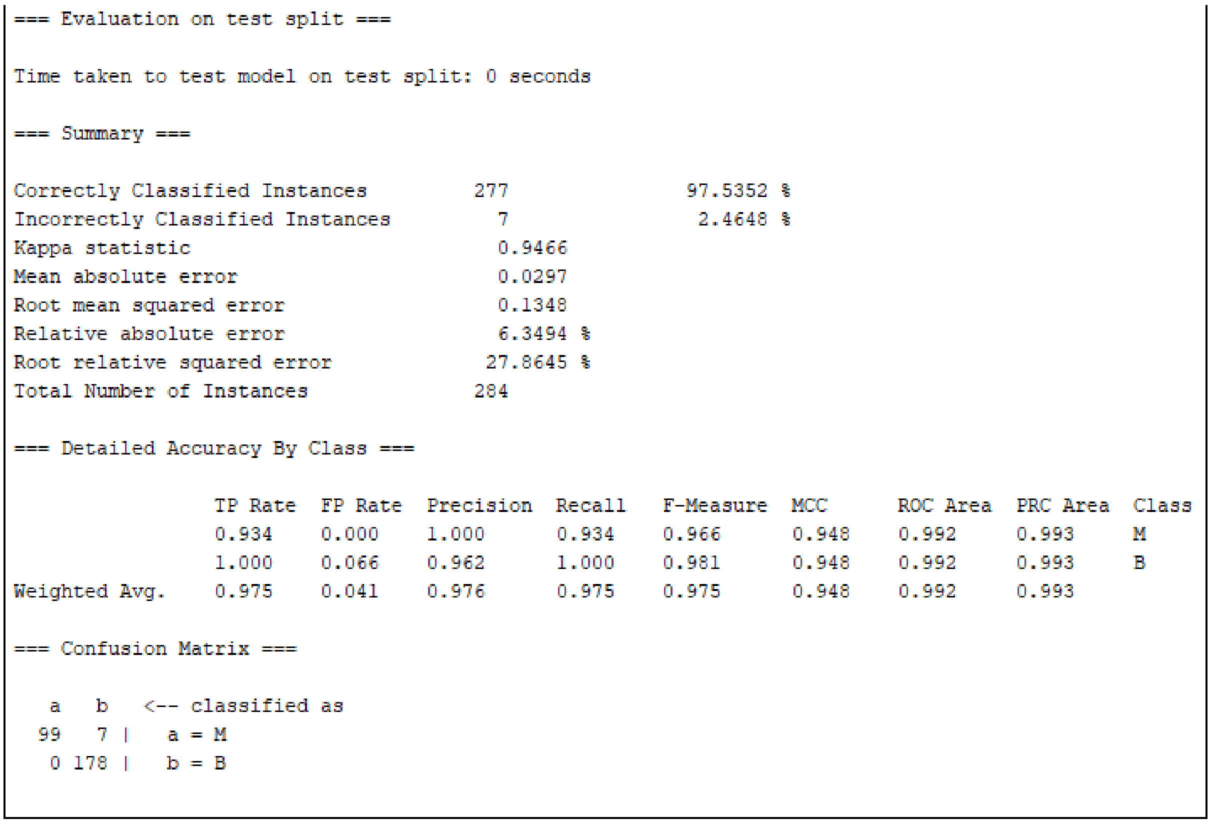
```
=== Summary ===

Correctly Classified Instances         166               97.076  %
Incorrectly Classified Instances         5                2.924  %
Kappa statistic                          0.9378
Mean absolute error                      0.0301
Root mean squared error                  0.1421
Relative absolute error                  6.4203 %
Root relative squared error             29.2711 %
Total Number of Instances              171

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.954    0.019    0.969      0.954    0.961      0.938    0.998     0.997     M
                 0.981    0.046    0.972      0.981    0.977      0.938    0.998     0.999     B
Weighted Avg.    0.971    0.036    0.971      0.971    0.971      0.938    0.998     0.998

=== Confusion Matrix ===

   a   b   <-- classified as
  62   3 |   a = M
   2 104 |   b = B
```

Percentage split 60/40				
Test number	Batch size	Hidden layers	Learning rate	Training
1. 4	100	a	0.3	342



```
=== Summary ===

Correctly Classified Instances         216               94.7368 %
Incorrectly Classified Instances        12                5.2632 %
Kappa statistic                          0.8878
Mean absolute error                      0.0577
Root mean squared error                  0.204
Relative absolute error                 12.2585 %
Root relative squared error             41.4851 %
Total Number of Instances              228

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.868    0.000    1.000      0.868    0.929      0.893    0.999     0.998     M
                 1.000    0.132    0.919      1.000    0.958      0.893    0.999     0.999     B
Weighted Avg.    0.947    0.079    0.952      0.947    0.947      0.893    0.999     0.999

=== Confusion Matrix ===

   a   b   <-- classified as
  79  12 |   a = M
   0 137 |   b = B
```

Percentage split 50/50				
Test number	Batch size	Hidden layers	Learning rate	Training
1. 5	100	a	0.3	285



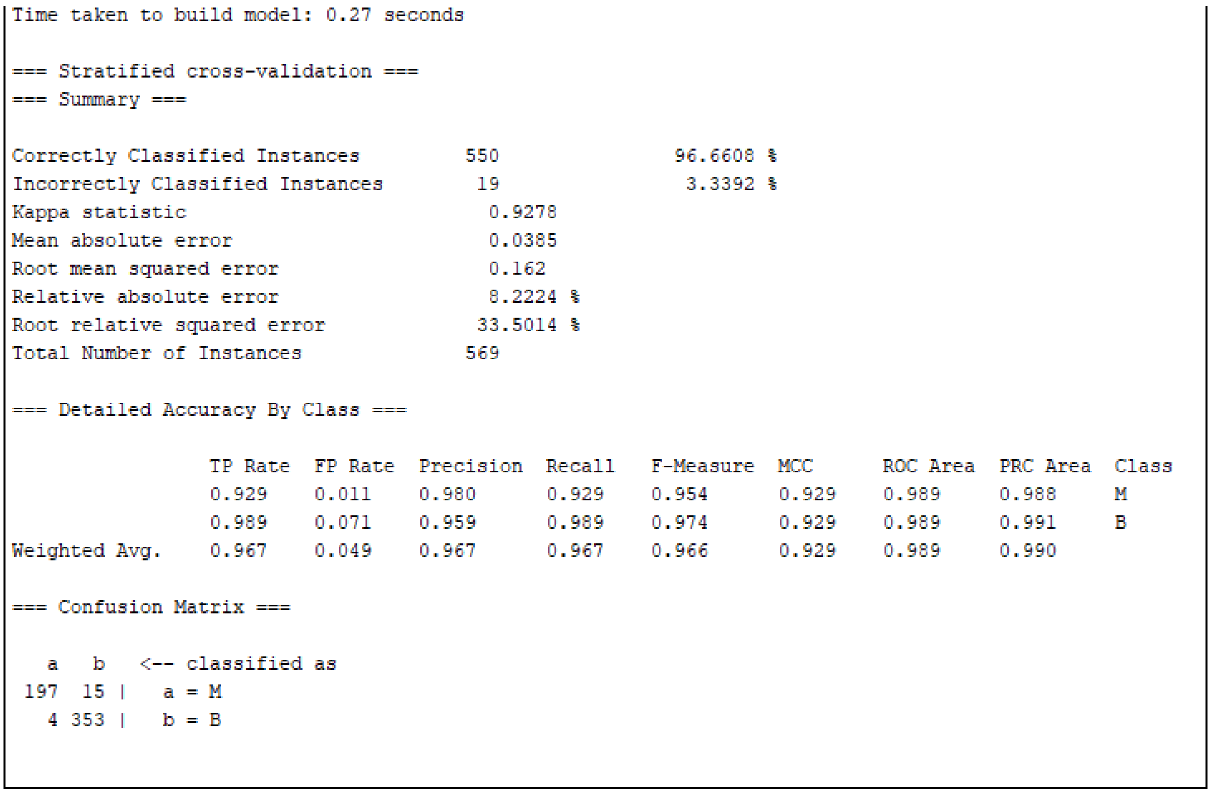
```
=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances         277               97.5352 %
Incorrectly Classified Instances         7                2.4648 %
Kappa statistic                          0.9466
Mean absolute error                      0.0297
Root mean squared error                  0.1348
Relative absolute error                  6.3494 %
Root relative squared error             27.8645 %
Total Number of Instances              284

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.934    0.000    1.000      0.934    0.966      0.948    0.992     0.993     M
                 1.000    0.066    0.962      1.000    0.981      0.948    0.992     0.993     B
Weighted Avg.    0.975    0.041    0.976      0.975    0.975      0.948    0.992     0.993

=== Confusion Matrix ===

   a   b   <-- classified as
  99   7 |   a = M
   0 178 |   b = B
```

As we can see from the results during testing if we increase the test size then the overall accuracy decreases with an outlier of 50/50 split. This is because of the model having less data to “learn” from. Next the RMSE and the MAE do differentiate much from each test. with the MAE being between 0.01 and 0.057. Next the RMSE was between 0.13 and 0.21. these all were big changes as in some cases 8% of the prediction was wrong.

Test 2 batch size

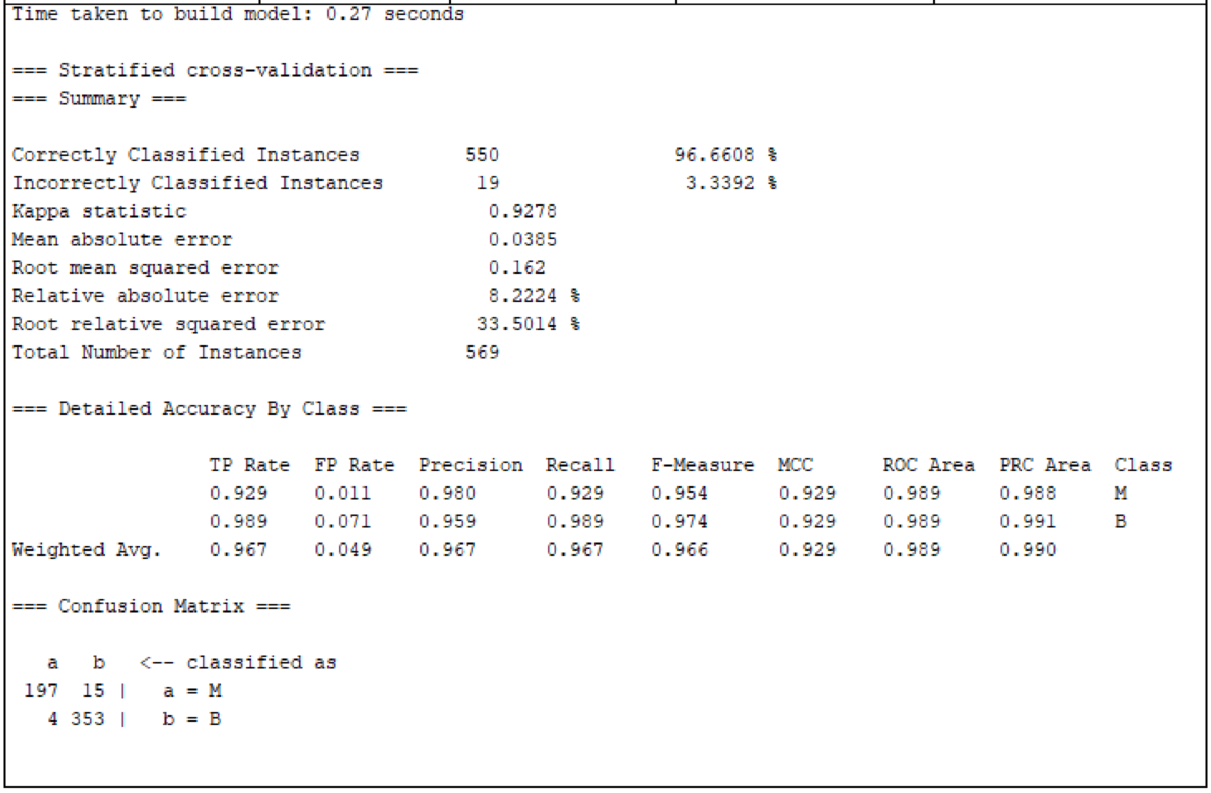
<u>For these tests we will use cross validation as it will encompass the whole data set.</u>

Cross validation with batch size changes				
Test number	Batch size	Hidden layers	Learning rate	Training
2.1	50	a	0.3	569



```
Time taken to build model: 0.27 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         550               96.6608 %
Incorrectly Classified Instances        19                3.3392 %
Kappa statistic                          0.9278
Mean absolute error                      0.0385
Root mean squared error                  0.162
Relative absolute error                  8.2224 %
Root relative squared error             33.5014 %
Total Number of Instances              569

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.929    0.011    0.980      0.929    0.954      0.929    0.989     0.988     M
                 0.989    0.071    0.959      0.989    0.974      0.929    0.989     0.991     B
Weighted Avg.    0.967    0.049    0.967      0.967    0.966      0.929    0.989     0.990

=== Confusion Matrix ===

   a   b   <-- classified as
 197  15 |   a = M
   4 353 |   b = B
```

Cross validation with batch size changes				
Test number	Batch size	Hidden layers	Learning rate	Training
2.2	100	a	0.3	500



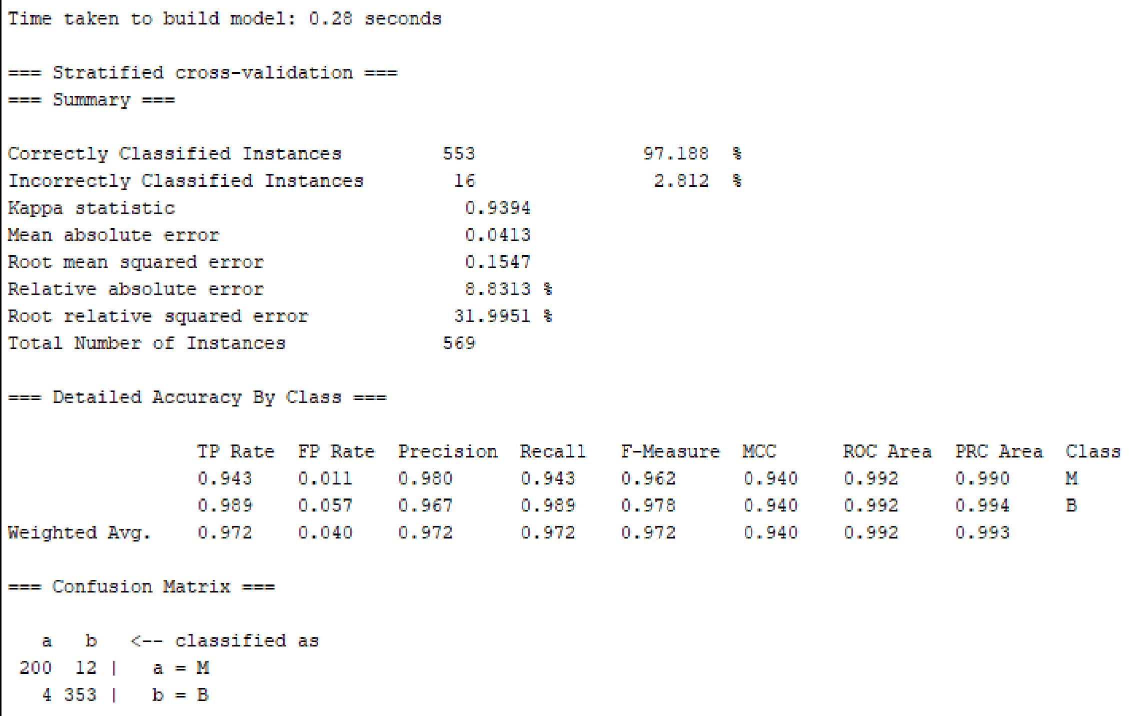
```
Time taken to build model: 0.27 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         550               96.6608 %
Incorrectly Classified Instances        19                3.3392 %
Kappa statistic                          0.9278
Mean absolute error                      0.0385
Root mean squared error                  0.162
Relative absolute error                  8.2224 %
Root relative squared error             33.5014 %
Total Number of Instances              569

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.929    0.011    0.980      0.929    0.954      0.929    0.989     0.988     M
                 0.989    0.071    0.959      0.989    0.974      0.929    0.989     0.991     B
Weighted Avg.    0.967    0.049    0.967      0.967    0.966      0.929    0.989     0.990

=== Confusion Matrix ===

   a   b   <-- classified as
 197  15 |   a = M
   4 353 |   b = B
```

In this series of test, we change the batch sizes. From the results we quickly learned that the made no difference to the metrics we are trying to measure as we found out the same results exist for a batch size of 50 or a batch size of 200. So testing was abandoned.

Test 3 learning changes

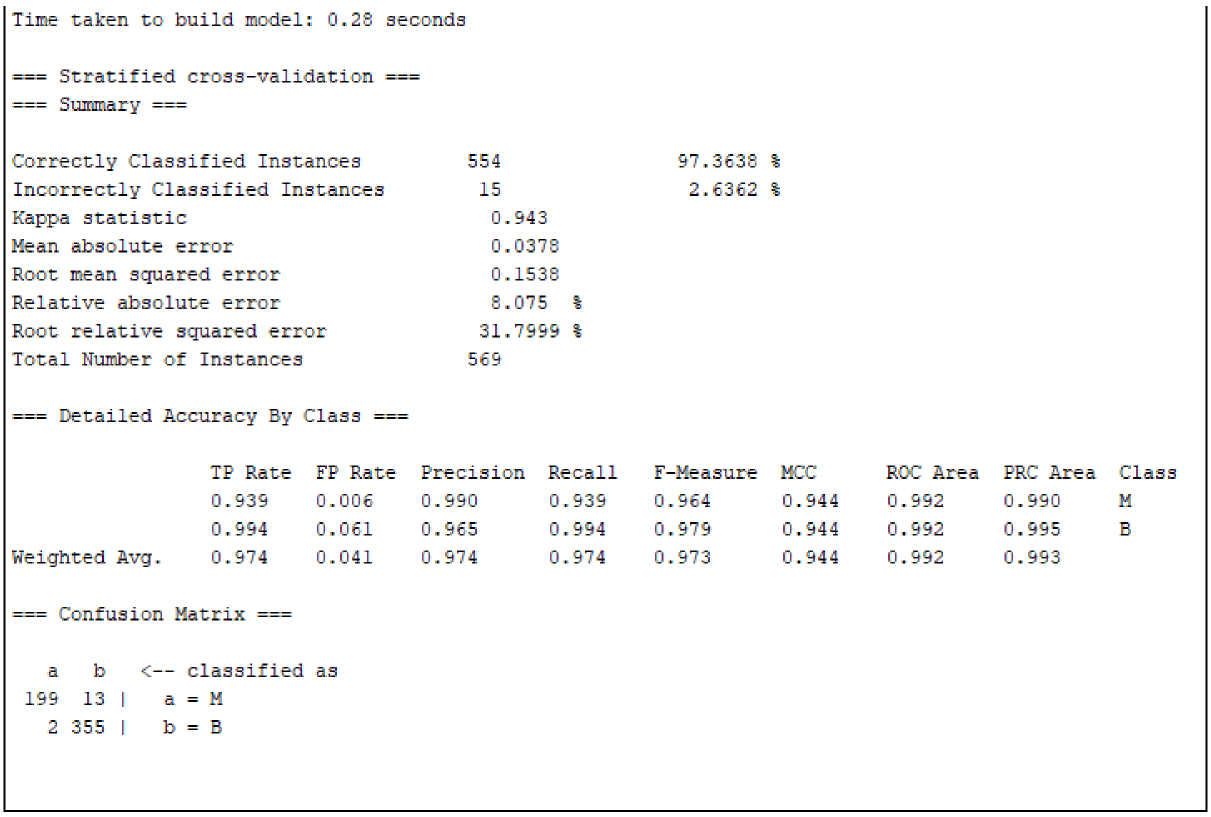
Cross validation with learning changes

Cross validation with learning rate changes				
Test number	Batch size	Hidden layers	Learning rate	Training
3.1	100	a	0.1	569



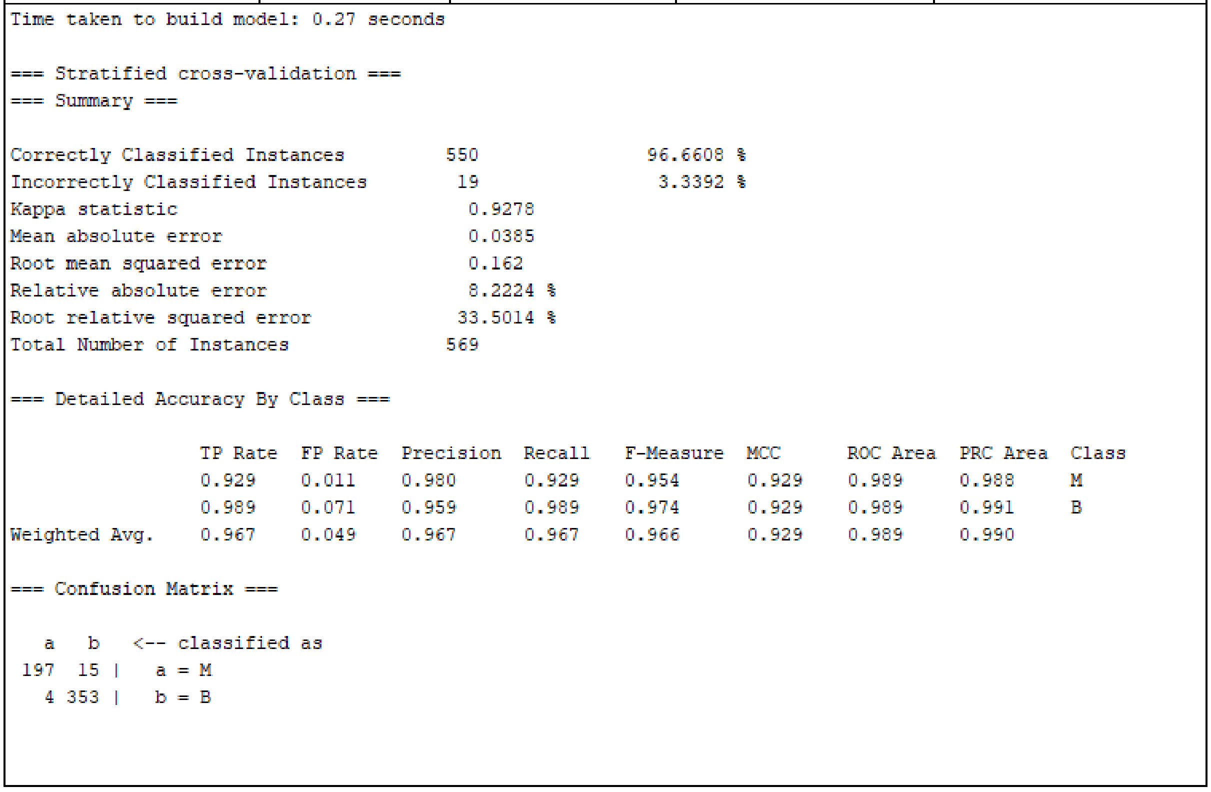
```
Time taken to build model: 0.28 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         553               97.188  %
Incorrectly Classified Instances        16                2.812  %
Kappa statistic                          0.9394
Mean absolute error                      0.0413
Root mean squared error                  0.1547
Relative absolute error                  8.8313 %
Root relative squared error             31.9951 %
Total Number of Instances              569

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.943    0.011    0.980      0.943    0.962      0.940    0.992     0.990     M
                 0.989    0.057    0.967      0.989    0.978      0.940    0.992     0.994     B
Weighted Avg.    0.972    0.040    0.972      0.972    0.972      0.940    0.992     0.993

=== Confusion Matrix ===

   a   b   <-- classified as
 200  12 |   a = M
   4 353 |   b = B
```

Cross validation with learning rate changes				
Test number	Batch size	Hidden layers	Learning rate	Training
3.2	100	a	0.2	569



```
Time taken to build model: 0.28 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         554               97.3638 %
Incorrectly Classified Instances        15                2.6362 %
Kappa statistic                          0.943
Mean absolute error                      0.0378
Root mean squared error                  0.1538
Relative absolute error                  8.075  %
Root relative squared error             31.7999 %
Total Number of Instances              569

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.939    0.006    0.990      0.939    0.964      0.944    0.992     0.990     M
                 0.994    0.061    0.965      0.994    0.979      0.944    0.992     0.995     B
Weighted Avg.    0.974    0.041    0.974      0.974    0.973      0.944    0.992     0.993

=== Confusion Matrix ===

   a   b   <-- classified as
 199  13 |   a = M
   2 355 |   b = B
```

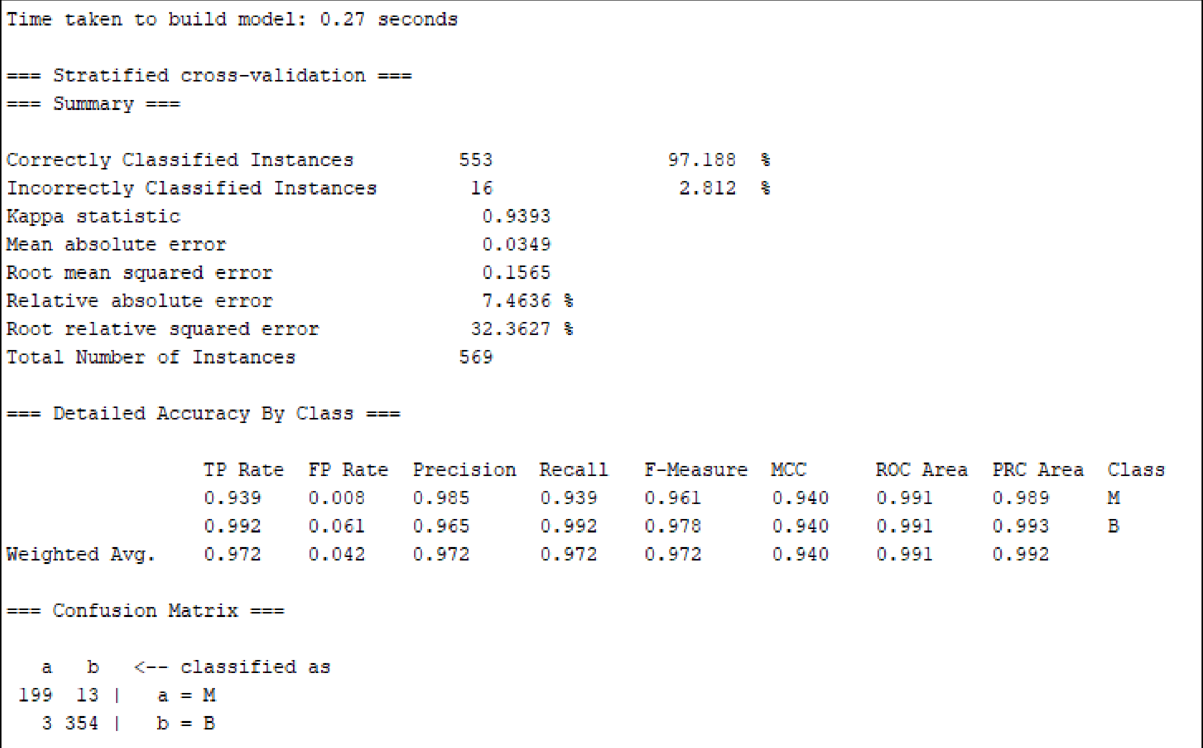
Cross validation with learning rate changes				
Test number	Batch size	Hidden layers	Learning rate	Training
3.3	100	a	0.3	569



```
Time taken to build model: 0.27 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         550               96.6608 %
Incorrectly Classified Instances        19                3.3392 %
Kappa statistic                          0.9278
Mean absolute error                      0.0385
Root mean squared error                  0.162
Relative absolute error                  8.2224 %
Root relative squared error             33.5014 %
Total Number of Instances              569

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.929    0.011    0.980      0.929    0.954      0.929    0.989     0.988     M
                 0.989    0.071    0.959      0.989    0.974      0.929    0.989     0.991     B
Weighted Avg.    0.967    0.049    0.967      0.967    0.966      0.929    0.989     0.990

=== Confusion Matrix ===

   a   b   <-- classified as
 197  15 |   a = M
   4 353 |   b = B
```

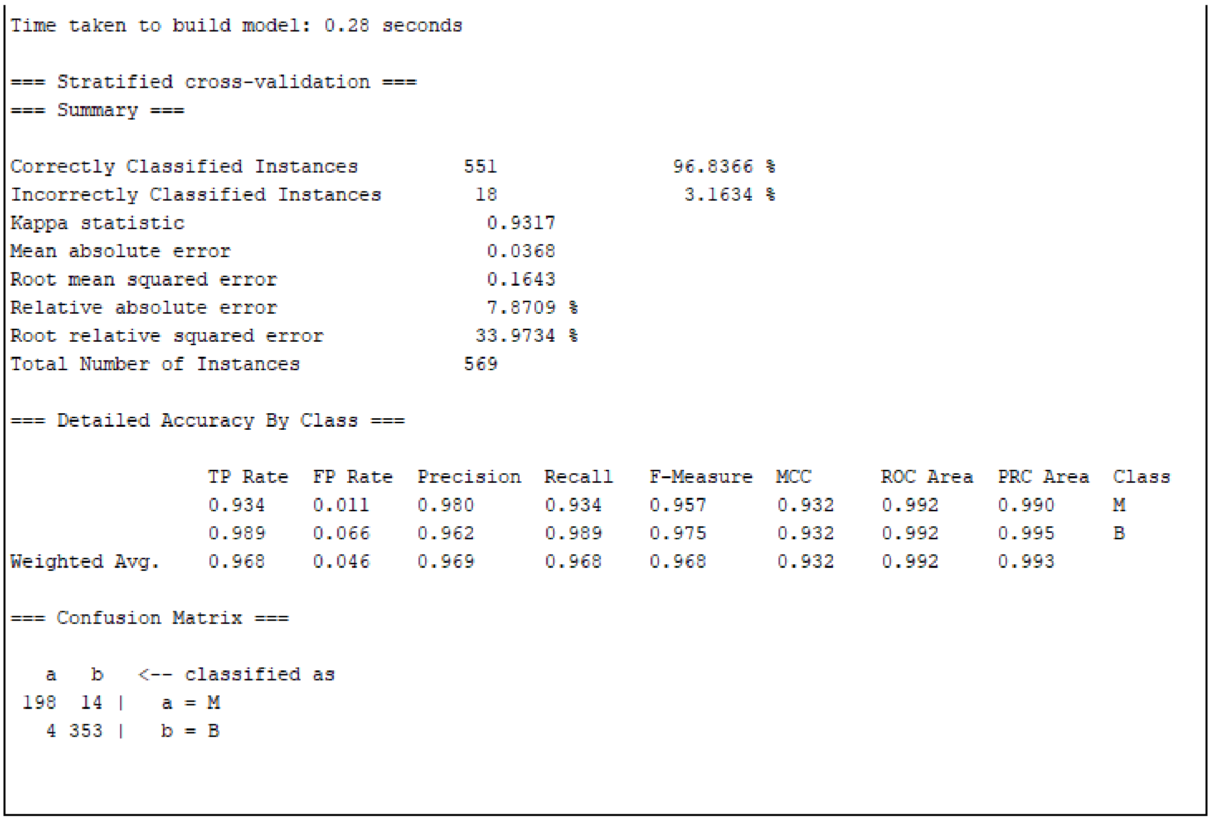
Cross validation with learning rate changes

Test number	Batch size	Hidden layers	Learning rate	Training
3.4	100	a	0.4	569



```
Time taken to build model: 0.27 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         553               97.188  %
Incorrectly Classified Instances        16                2.812  %
Kappa statistic                          0.9393
Mean absolute error                      0.0349
Root mean squared error                  0.1565
Relative absolute error                  7.4636 %
Root relative squared error             32.3627 %
Total Number of Instances              569

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.939    0.008    0.985      0.939    0.961      0.940    0.991     0.989     M
                 0.992    0.061    0.965      0.992    0.978      0.940    0.991     0.993     B
Weighted Avg.    0.972    0.042    0.972      0.972    0.972      0.940    0.991     0.992

=== Confusion Matrix ===

   a   b   <-- classified as
 199  13 |   a = M
   3 354 |   b = B
```

Cross validation with learning rate changes				
Test number	Batch size	Hidden layers	Learning rate	Training
3.5	100	a	0.5	569



```
Time taken to build model: 0.28 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         551               96.8366 %
Incorrectly Classified Instances        18                3.1634 %
Kappa statistic                          0.9317
Mean absolute error                      0.0368
Root mean squared error                  0.1643
Relative absolute error                  7.8709 %
Root relative squared error             33.9734 %
Total Number of Instances              569

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.934    0.011    0.980      0.934    0.957      0.932    0.992     0.990     M
                 0.989    0.066    0.962      0.989    0.975      0.932    0.992     0.995     B
Weighted Avg.    0.968    0.046    0.969      0.968    0.968      0.932    0.992     0.993

=== Confusion Matrix ===

   a   b   <-- classified as
 198  14 |   a = M
   4 353 |   b = B
```

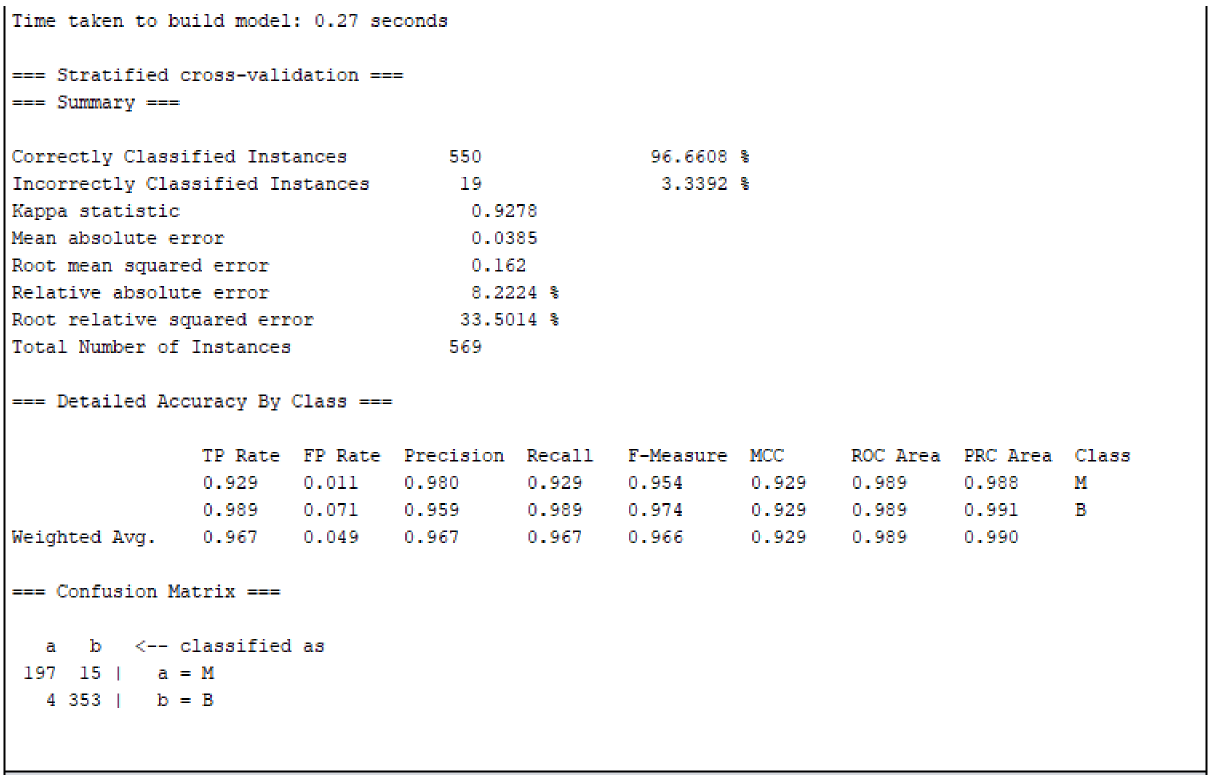
For this test we used 10-fold cross validation. This is to ensure that all the data can appear in both the testing and training sets. The hyper parameter we changed for this was the learning rate. This was to see if a higher or lower rate would lead to errors in classification.
In the results we found out that this hyper parameter
did very little to   affect accuracy of the model as it was between 0.96 and 0.98 with all the tests. Next the MAE did not go change beyond 0.03 and 0.04.

This was the same for the RMSE which was between 0.15 and 0.17. This could have been due to the sizing of the data set and that it was only 569 records or the fact that not much variety existed between records in an attribute.

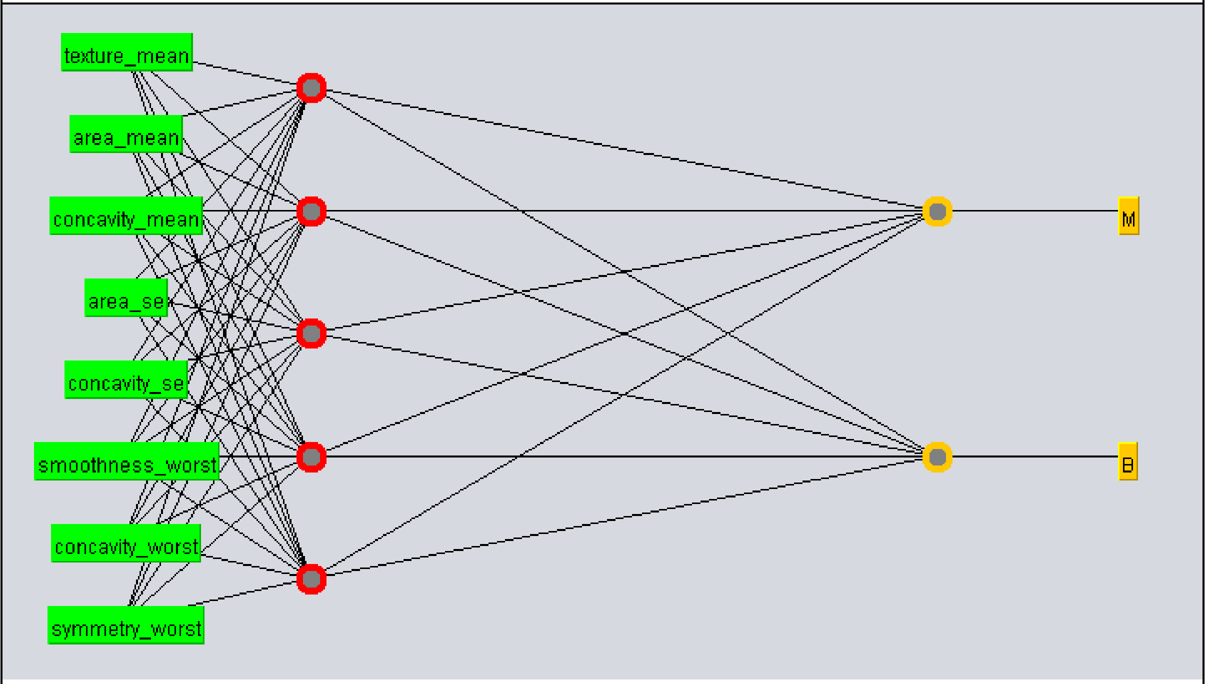
Test 4 hidden layer

<u>Changing the hidden layer</u>

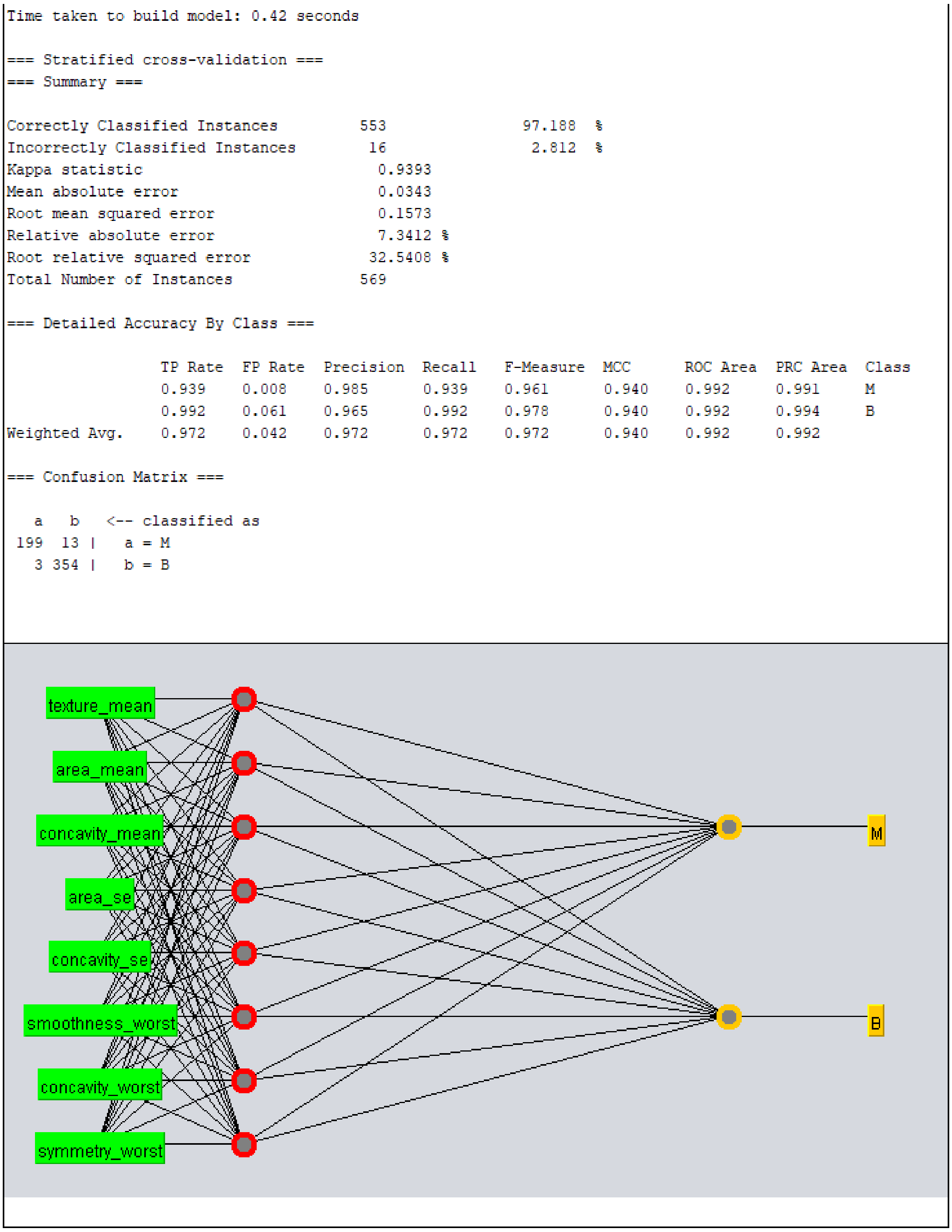
Cross validation with hidden layer changes				
Test number	Batch size	Hidden layers	Learning rate	Training
4.1	100	A =(attributes +classes) /2	0.3	569



```
Time taken to build model: 0.27 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         550               96.6608 %
Incorrectly Classified Instances        19                3.3392 %
Kappa statistic                          0.9278
Mean absolute error                      0.0385
Root mean squared error                  0.162
Relative absolute error                  8.2224 %
Root relative squared error             33.5014 %
Total Number of Instances              569

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.929    0.011    0.980      0.929    0.954      0.929    0.989     0.988     M
                 0.989    0.071    0.959      0.989    0.974      0.929    0.989     0.991     B
Weighted Avg.    0.967    0.049    0.967      0.967    0.966      0.929    0.989     0.990

=== Confusion Matrix ===

   a   b   <-- classified as
 197  15 |   a = M
   4 353 |   b = B
```

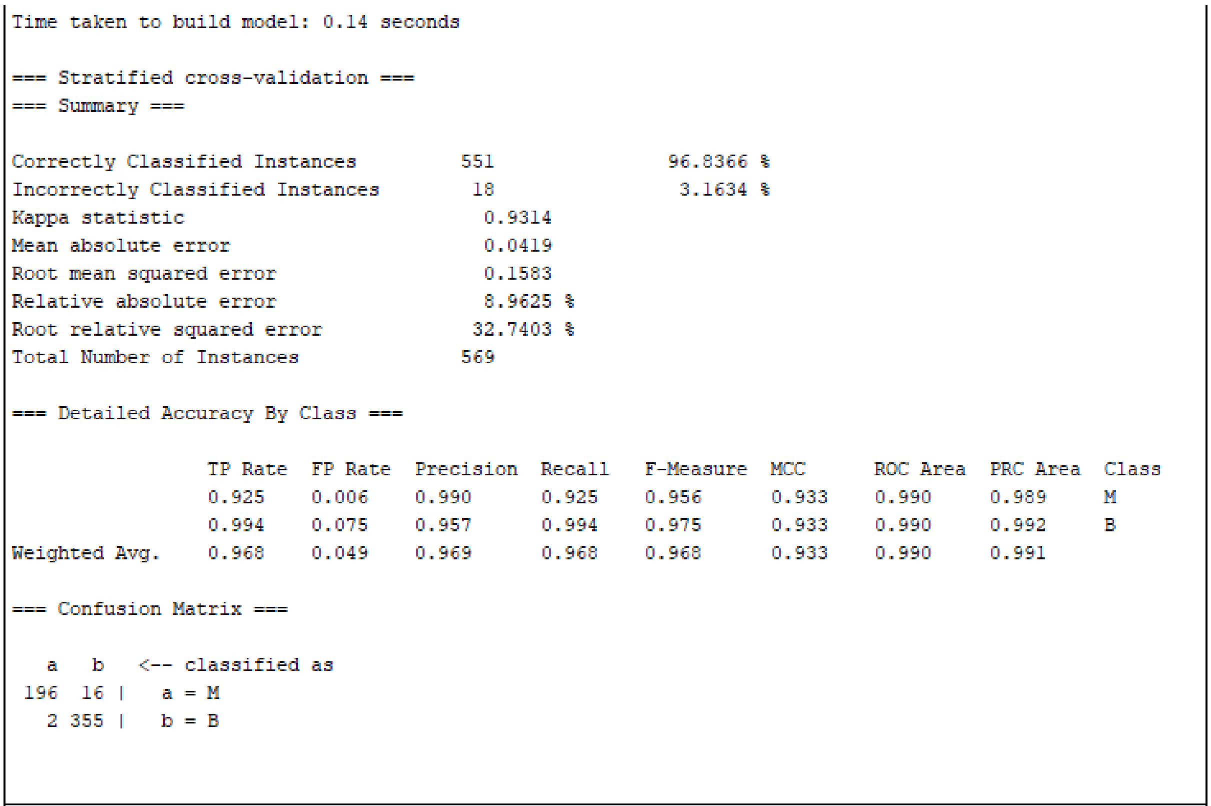


Cross validation with hidden layer changes				
Test number	Batch size	Hidden layers	Learning rate	Training
4.2	100	I =attributes	0.3	569

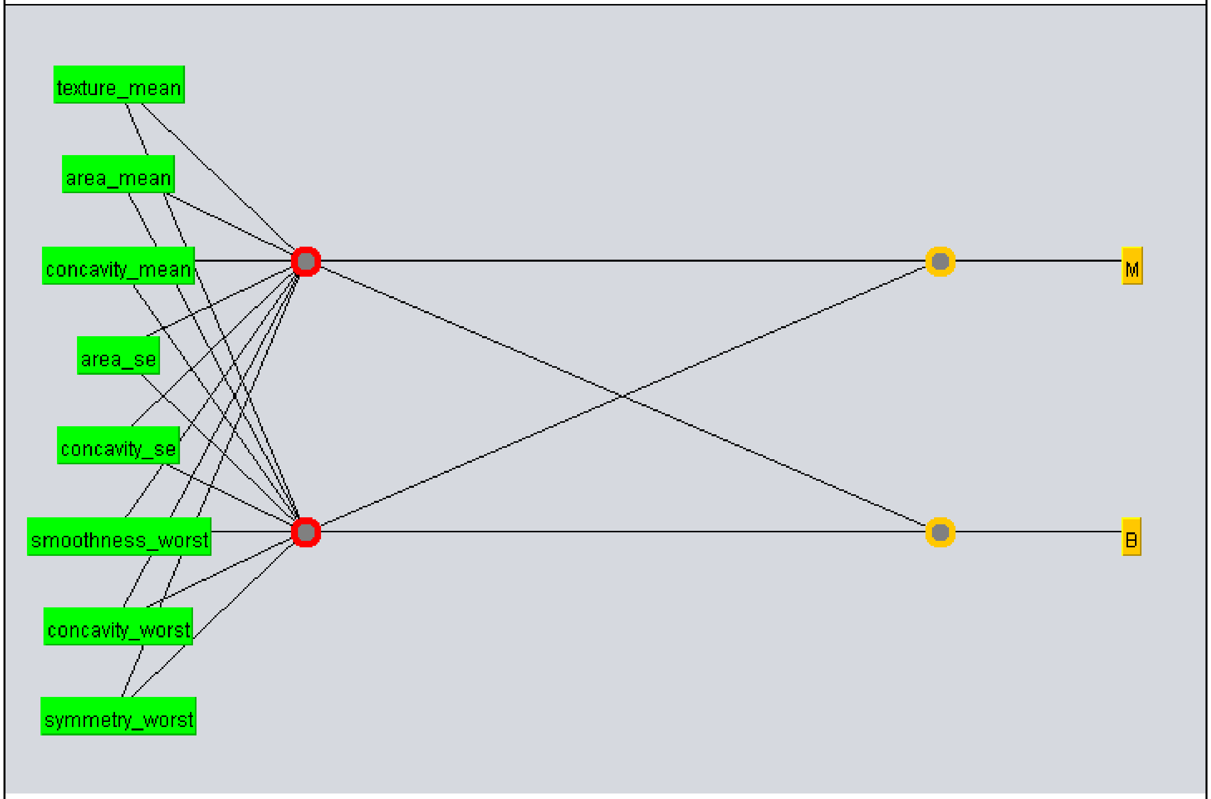


```
Time taken to build model: 0.42 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         553               97.188  %
Incorrectly Classified Instances        16                2.812  %
Kappa statistic                          0.9393
Mean absolute error                      0.0343
Root mean squared error                  0.1573
Relative absolute error                  7.3412 %
Root relative squared error             32.5408 %
Total Number of Instances              569

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.939    0.008    0.985      0.939    0.961      0.940    0.992     0.991     M
                 0.992    0.061    0.965      0.992    0.978      0.940    0.992     0.994     B
Weighted Avg.    0.972    0.042    0.972      0.972    0.972      0.940    0.992     0.992

=== Confusion Matrix ===

   a   b   <-- classified as
 199  13 |   a = M
   3 354 |   b = B
```

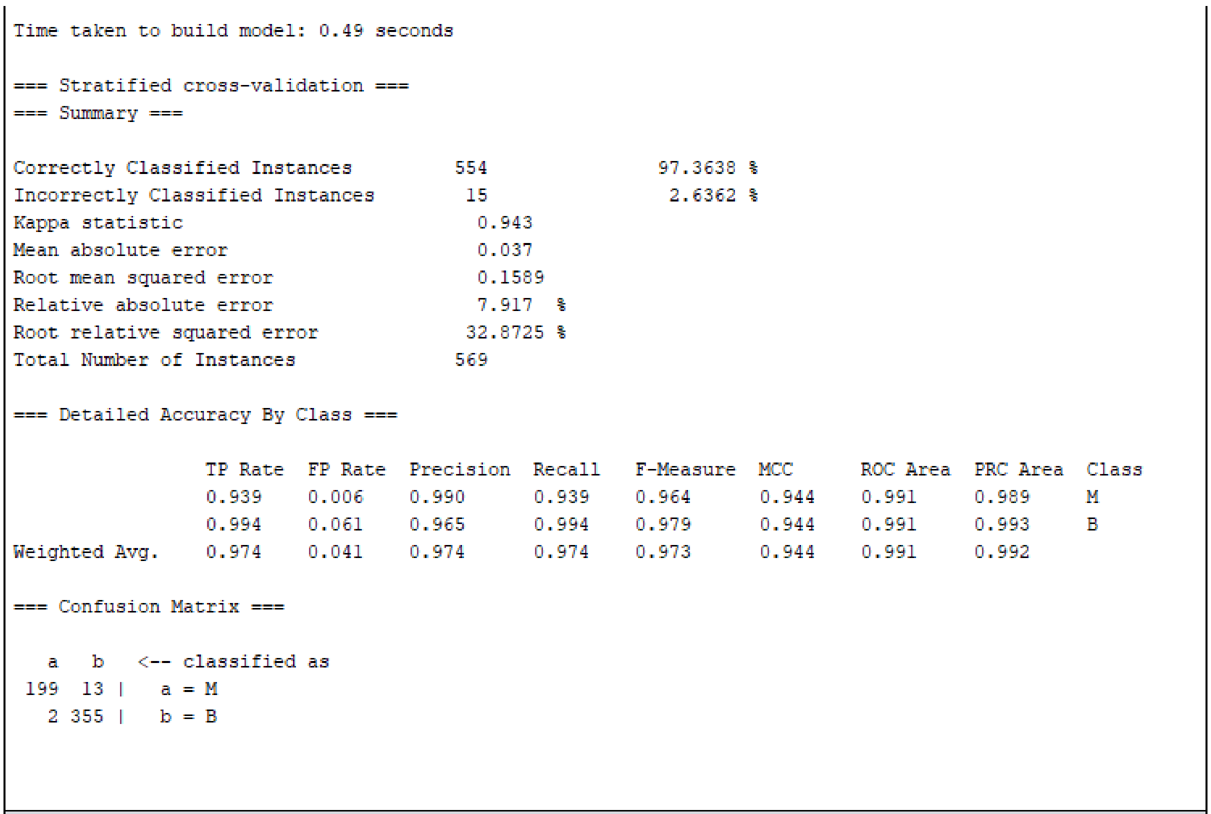
Cross validation with hidden layer changes				
Test number	Batch size	Hidden layers	Learning rate	Training
4.3	100	O = classes	0.3	569



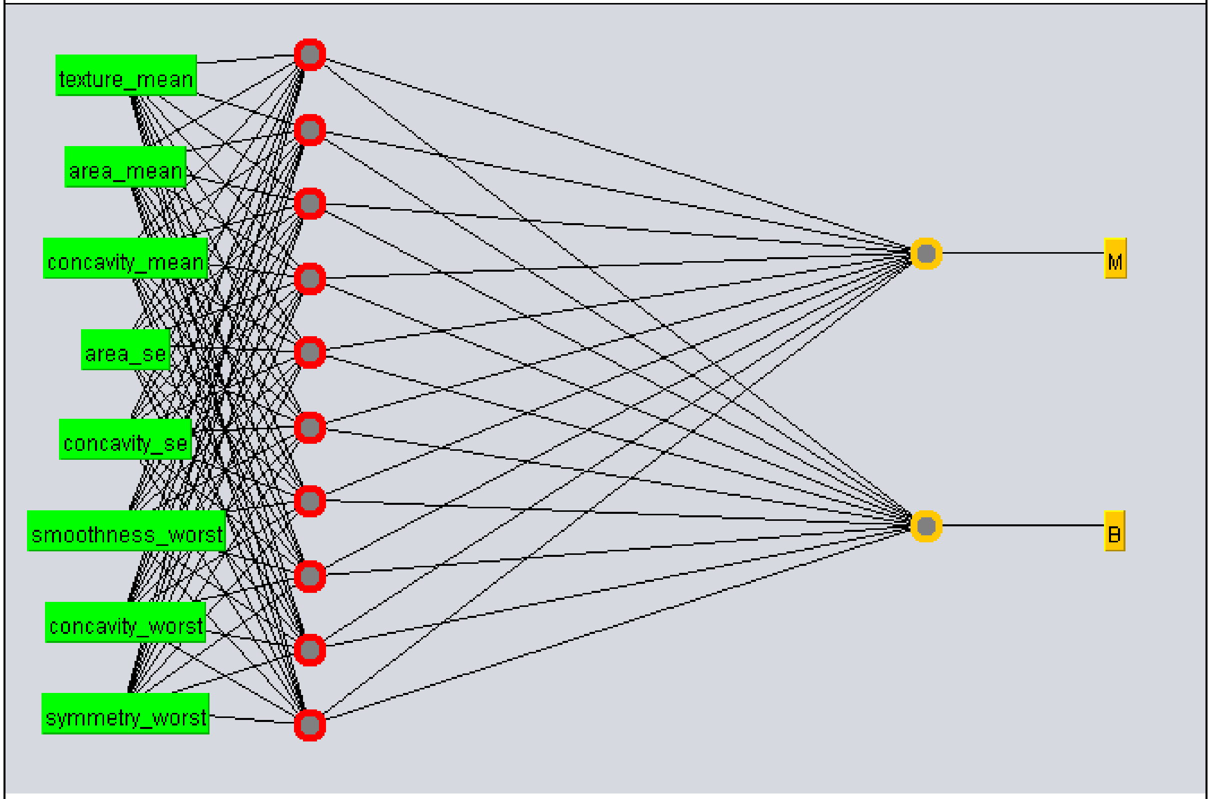
```
Time taken to build model: 0.14 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         551               96.8366 %
Incorrectly Classified Instances        18                3.1634 %
Kappa statistic                          0.9314
Mean absolute error                      0.0419
Root mean squared error                  0.1583
Relative absolute error                  8.9625 %
Root relative squared error             32.7403 %
Total Number of Instances              569

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.925    0.006    0.990      0.925    0.956      0.933    0.990     0.989     M
                 0.994    0.075    0.957      0.994    0.975      0.933    0.990     0.992     B
Weighted Avg.    0.968    0.049    0.969      0.968    0.968      0.933    0.990     0.991

=== Confusion Matrix ===

   a   b   <-- classified as
 196  16 |   a = M
   2 355 |   b = B
```



Cross validation with hidden layer changes				
Test number	Batch size	Hidden layers	Learning rate	Training
4.4	100	T= attributes +classes	0.3	569



```
Time taken to build model: 0.49 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         554               97.3638 %
Incorrectly Classified Instances        15                2.6362 %
Kappa statistic                          0.943
Mean absolute error                      0.037
Root mean squared error                  0.1589
Relative absolute error                  7.917  %
Root relative squared error             32.8725 %
Total Number of Instances              569

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.939    0.006    0.990      0.939    0.964      0.944    0.991     0.989     M
                 0.994    0.061    0.965      0.994    0.979      0.944    0.991     0.993     B
Weighted Avg.    0.974    0.041    0.974      0.974    0.973      0.944    0.991     0.992

=== Confusion Matrix ===

   a   b   <-- classified as
 199  13 |   a = M
   2 355 |   b = B
```



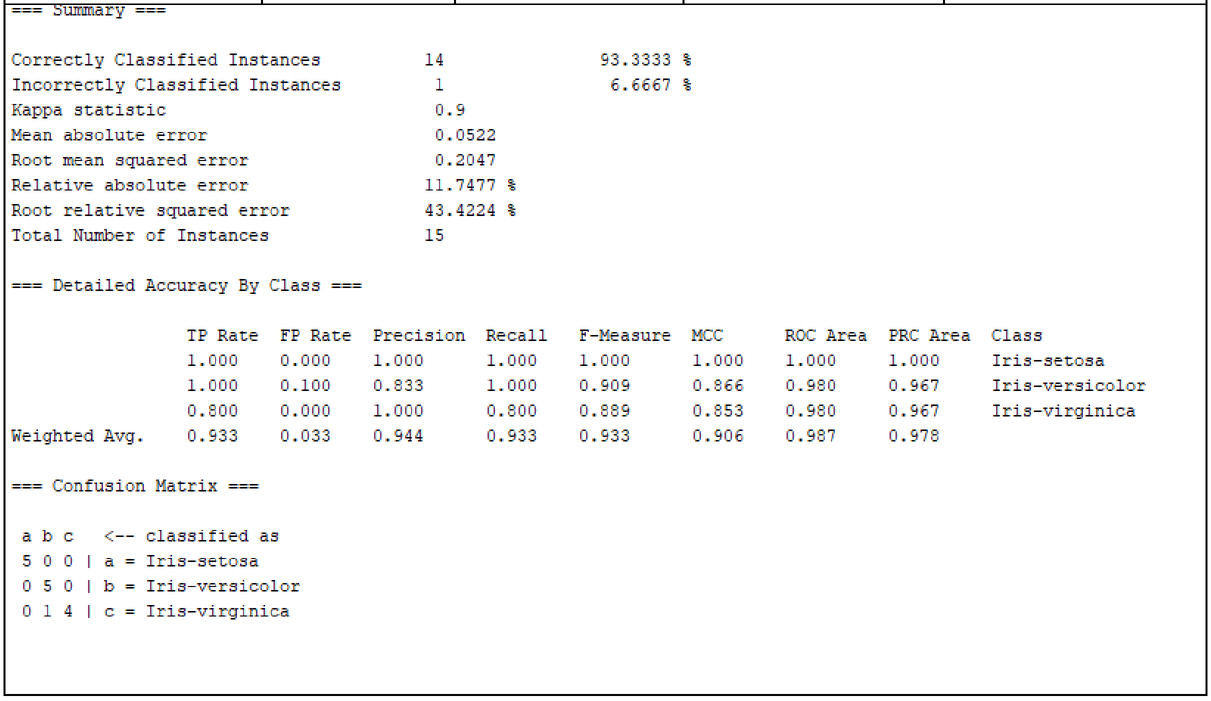
Lastly, we changed the hidden layer in the model by adding or decreasing the number of nodes. again, the accuracy did not change as much and was between 0.96 and 0.98. next the MAE was between 0.03 and 0.045 which is a very small change. This could have been due to the data size or the variety of data.

Test 5 Bench mark test

In order to see if what we did was Comparison with the iris dataset. This is widely known as a benchmark data set. In these tests we will be running some of tests we conducted on our data to see if they were viable or something was wrong with our method. we cannot replicate all the tests as they would be unproductive. So, we will replace the percentage split portion of the tests.

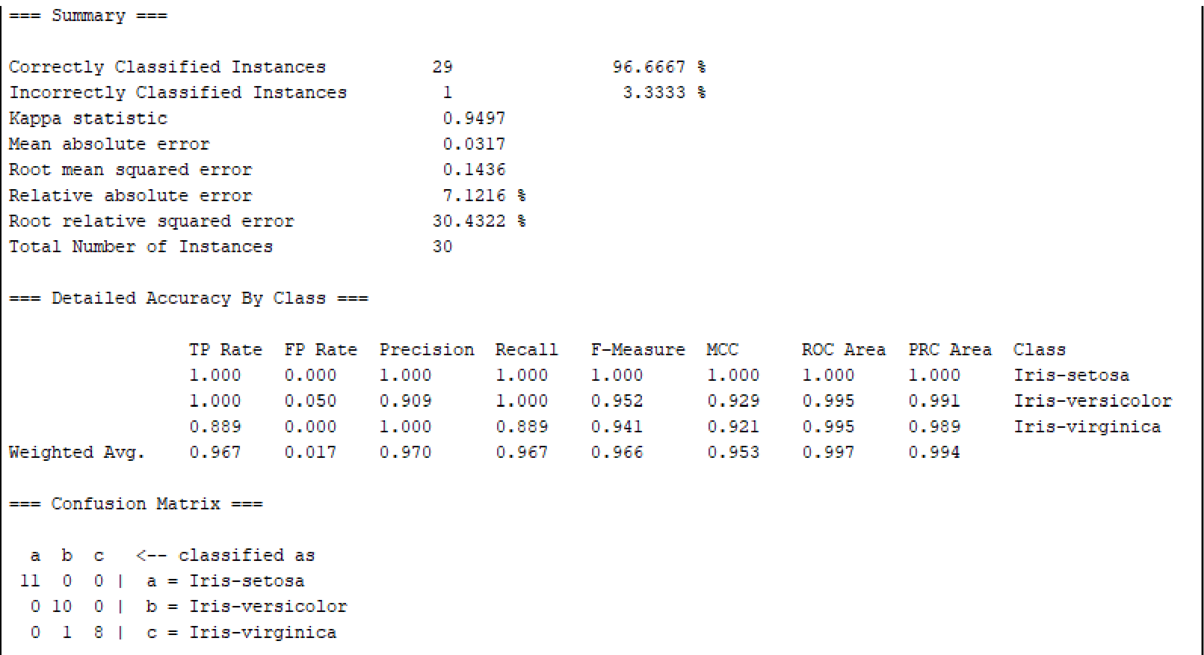
Iris bench mark data set

Percentage split 90/10				
Test number	Batch size	Hidden layers	Learning rate	Training
1.1	100	a	0.3	513



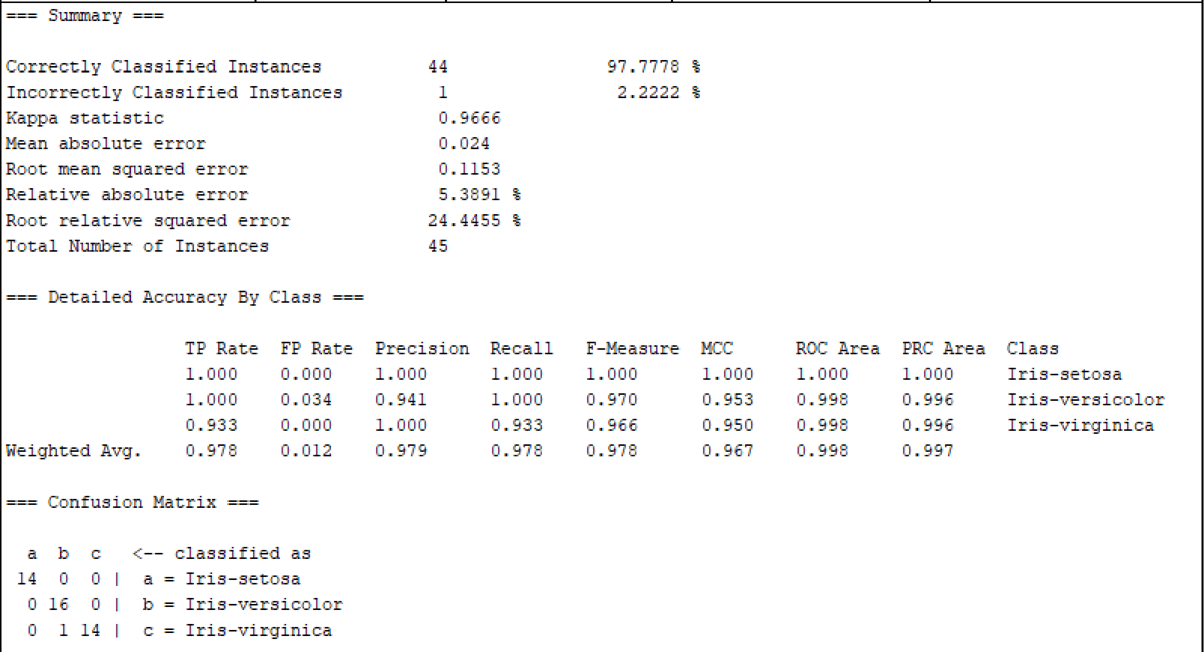
```
=== Summary ===

Correctly Classified Instances          14               93.3333 %
Incorrectly Classified Instances         1                6.6667 %
Kappa statistic                          0.9
Mean absolute error                      0.0522
Root mean squared error                  0.2047
Relative absolute error                 11.7477 %
Root relative squared error             43.4224 %
Total Number of Instances               15

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 1.000    0.100    0.833      1.000    0.909      0.866    0.980     0.967     Iris-versicolor
                 0.800    0.000    1.000      0.800    0.889      0.853    0.980     0.967     Iris-virginica
Weighted Avg.    0.933    0.033    0.944      0.933    0.933      0.906    0.987     0.978

=== Confusion Matrix ===

 a b c   <-- classified as
 5 0 0 | a = Iris-setosa
 0 5 0 | b = Iris-versicolor
 0 1 4 | c = Iris-virginica
```

Percentage split 80/20				
Test number	Batch size	Hidden layers	Learning rate	Training
1.2	100	a	0.3	456



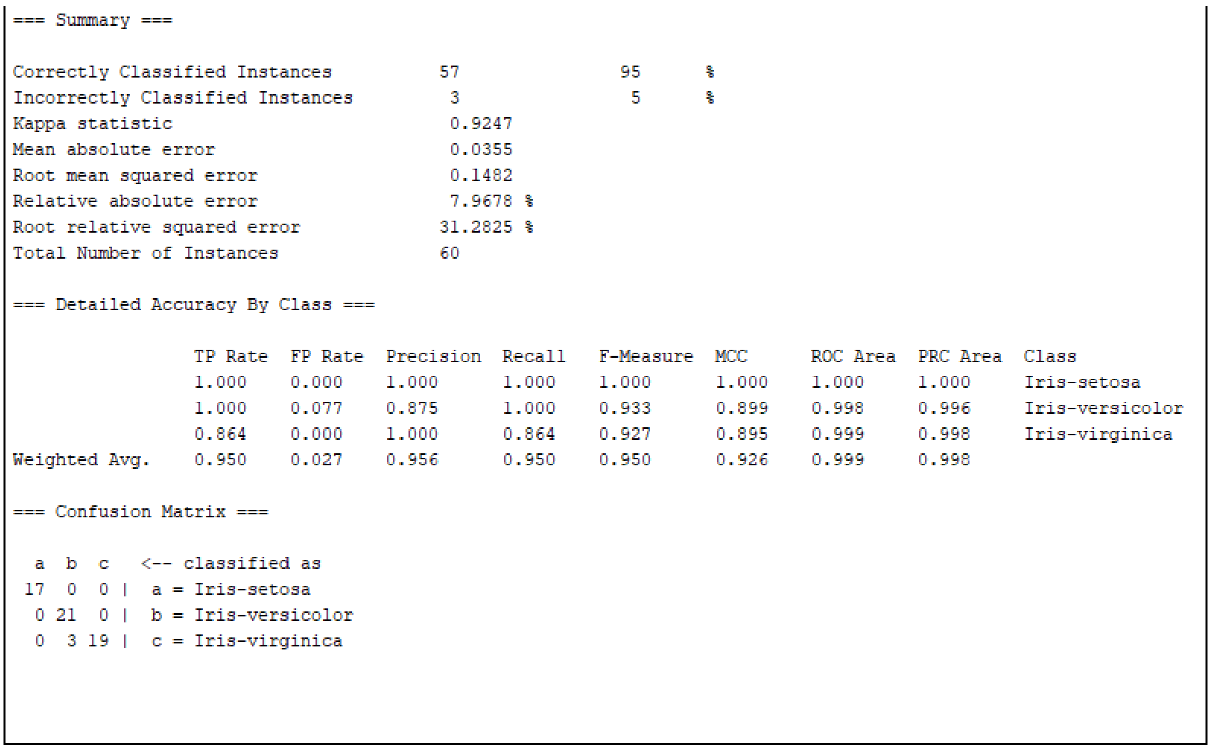
```
=== Summary ===

Correctly Classified Instances          29               96.6667 %
Incorrectly Classified Instances         1                3.3333 %
Kappa statistic                          0.9497
Mean absolute error                      0.0317
Root mean squared error                  0.1436
Relative absolute error                  7.1216 %
Root relative squared error             30.4322 %
Total Number of Instances               30

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 1.000    0.050    0.909      1.000    0.952      0.929    0.995     0.991     Iris-versicolor
                 0.889    0.000    1.000      0.889    0.941      0.921    0.995     0.989     Iris-virginica
Weighted Avg.    0.967    0.017    0.970      0.967    0.966      0.953    0.997     0.994

=== Confusion Matrix ===

  a  b  c   <-- classified as
 11  0  0 |  a = Iris-setosa
  0 10  0 |  b = Iris-versicolor
  0  1  8 |  c = Iris-virginica
```

Percentage split70/30				
Test number	Batch size	Hidden layers	Learning rate	Training
1.3	100	a	0.3	399



```
=== Summary ===

Correctly Classified Instances          44               97.7778 %
Incorrectly Classified Instances         1                2.2222 %
Kappa statistic                          0.9666
Mean absolute error                      0.024
Root mean squared error                  0.1153
Relative absolute error                  5.3891 %
Root relative squared error             24.4455 %
Total Number of Instances               45

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 1.000    0.034    0.941      1.000    0.970      0.953    0.998     0.996     Iris-versicolor
                 0.933    0.000    1.000      0.933    0.966      0.950    0.998     0.996     Iris-virginica
Weighted Avg.    0.978    0.012    0.979      0.978    0.978      0.967    0.998     0.997

=== Confusion Matrix ===

  a  b  c   <-- classified as
 14  0  0 |  a = Iris-setosa
  0 16  0 |  b = Iris-versicolor
  0  1 14 |  c = Iris-virginica
```

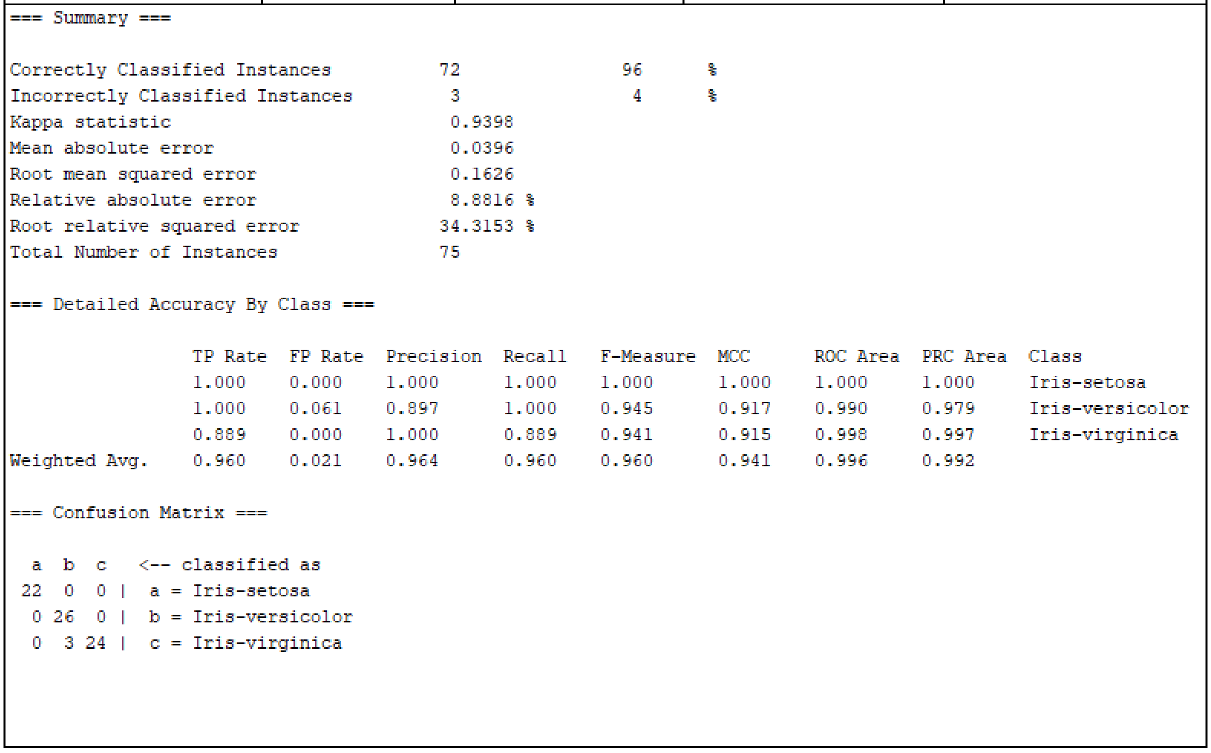
Percentage split 60/40				
Test number	Batch size	Hidden layers	Learning rate	Training
1. 4	100	a	0.3	342



```
=== Summary ===

Correctly Classified Instances          57               95      %
Incorrectly Classified Instances         3                5      %
Kappa statistic                          0.9247
Mean absolute error                      0.0355
Root mean squared error                  0.1482
Relative absolute error                  7.9678 %
Root relative squared error             31.2825 %
Total Number of Instances               60

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 1.000    0.077    0.875      1.000    0.933      0.899    0.998     0.996     Iris-versicolor
                 0.864    0.000    1.000      0.864    0.927      0.895    0.999     0.998     Iris-virginica
Weighted Avg.    0.950    0.027    0.956      0.950    0.950      0.926    0.999     0.998

=== Confusion Matrix ===

  a  b  c   <-- classified as
 17  0  0 |  a = Iris-setosa
  0 21  0 |  b = Iris-versicolor
  0  3 19 |  c = Iris-virginica
```

Percentage split 50/50

Test number	Batch size	Hidden layers	Learning rate	Training
1. 5	100	a	0.3	285



```
=== Summary ===

Correctly Classified Instances          72               96      %
Incorrectly Classified Instances         3                4      %
Kappa statistic                          0.9398
Mean absolute error                      0.0396
Root mean squared error                  0.1626
Relative absolute error                  8.8816 %
Root relative squared error             34.3153 %
Total Number of Instances               75

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 1.000    0.061    0.897      1.000    0.945      0.917    0.990     0.979     Iris-versicolor
                 0.889    0.000    1.000      0.889    0.941      0.915    0.998     0.997     Iris-virginica
Weighted Avg.    0.960    0.021    0.964      0.960    0.960      0.941    0.996     0.992

=== Confusion Matrix ===

  a  b  c   <-- classified as
 22  0  0 |  a = Iris-setosa
  0 26  0 |  b = Iris-versicolor
  0  3 24 |  c = Iris-virginica
```

From this we can see that it matches the results that we generate for the same test with a variance of 0.10 and 0.20 for the RMSE. The MAE was between 0.024 and 0.039. next the accuracy was between 0.94 and 0.97.

This shows even with a greater percentage split we can see that a change does occur but a small one at that. This again might be due to small data set size and the lack of variety in the data form.

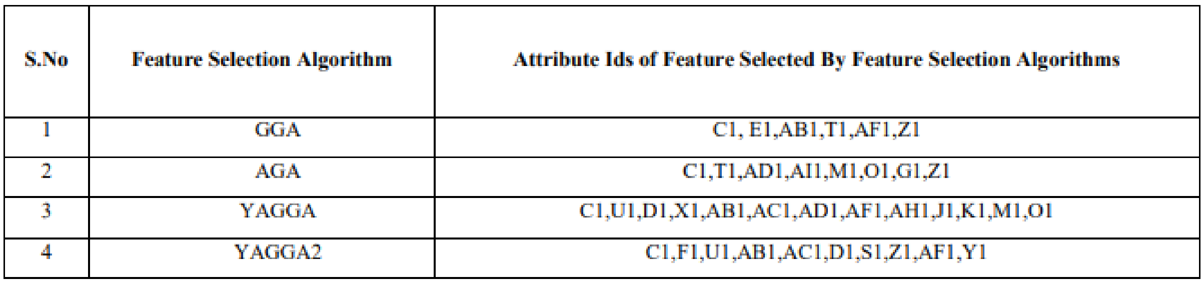
Conclusion of results

As mentioned before the changes in the hyper parameters do not affect the classification performance much, in fact there is an underlying base of results which none of the tests fell under. This was roughly 96% accuracy, 0.05 % MAE and 0.20 RMSE. Again, this could be due to the small data size, which means there isn't much deviation in records or variety in the records such as vast outliers which may challenge current weight assignment.

Discussion or results

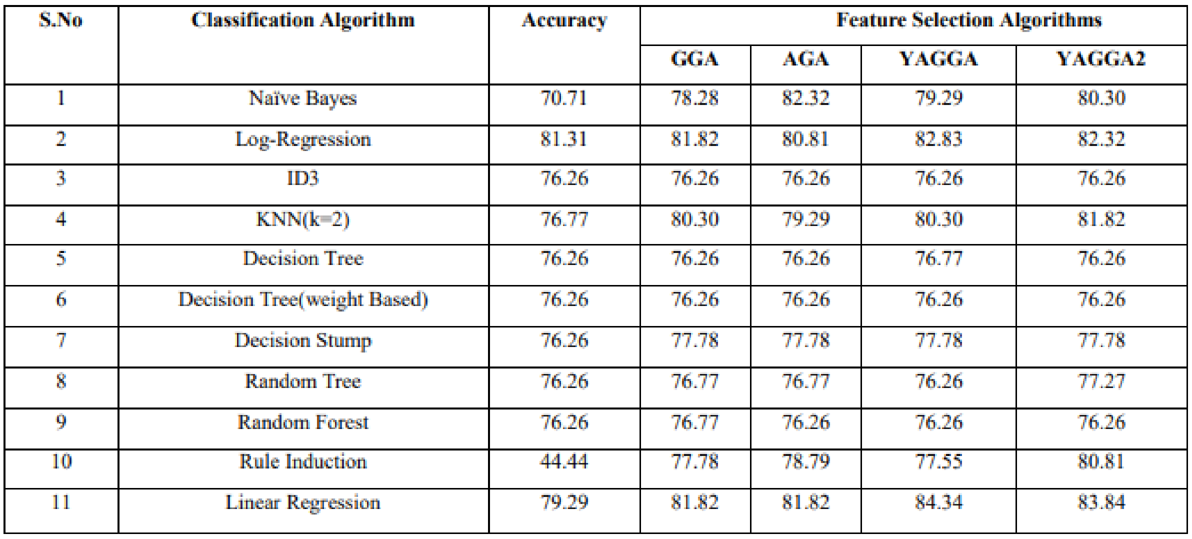
Discussion of results. First, we used the ZeroR classifier to establish a baseline for the predictability of a tumour. This yielded poor result in accuracy with 62% and a MAE and RMSE of 0.48. Next we used our MLP to see if we could beat those metrics and we did with an overarching accuracy of over 96%. The MAE did not move beyond 0.01 and 0.05. and the RMSE was not beyond 0.13 and 0.21. These were only in the percentage split tests.

The cross validation yielded a tighter spread as seen in the analysis. Lastly, we can rule out problems such as overfitting to our data as we saw using the iris data set that we gained similar results for the given test. This is also supported by paper 2 as some of their feature count was higher than ours.



S.No	Feature Selection Algorithm	Attribute Ids of Feature Selected By Feature Selection Algorithms
1	GGA	C1, E1,AB1,T1,AF1,Z1
2	AGA	C1,T1,AD1,AI1,M1,O1,G1,Z1
3	YAGGA	C1,U1,D1,X1,AB1,AC1,AD1,AF1,AH1,J1,K1,M1,O1
4	YAGGA2	C1,F1,U1,AB1,AC1,D1,S1,Z1,AF1,Y1

Overall, we found out that no matter what hyper-parameters we change we still achieved a better result in terms of classification. This was surprising as some parameters were pushed to extremes such as batch size and the percentage split. (Table 1)



S.No	Classification Algorithm	Accuracy	Feature Selection Algorithms			
			GGA	AGA	YAGGA	YAGGA2
1	Naïve Bayes	70.71	78.28	82.32	79.29	80.30
2	Log-Regression	81.31	81.82	80.81	82.83	82.32
3	ID3	76.26	76.26	76.26	76.26	76.26
4	KNN(k=2)	76.77	80.30	79.29	80.30	81.82
5	Decision Tree	76.26	76.26	76.26	76.77	76.26
6	Decision Tree(weight Based)	76.26	76.26	76.26	76.26	76.26
7	Decision Stump	76.26	77.78	77.78	77.78	77.78
8	Random Tree	76.26	76.77	76.77	76.26	77.27
9	Random Forest	76.26	76.77	76.26	76.26	76.26
10	Rule Induction	44.44	77.78	78.79	77.55	80.81
11	Linear Regression	79.29	81.82	81.82	84.34	83.84

Comparing this to research paper 2. They used 11 different classifiers to see how well they could predict the tumour. However, these results faired different to ours, as they used different feature selection methods. (3)
 This is further supposed by table two in which ANNs were used and generally yielded a higher accuracy. However, both papers did not display other metrics we used such as RMSE and MAE, so a comparison of this is not possible.

cross validation using naïve Bayes				
Test number	Batch size	Hidden layers	Learning rate	Training
2.1	100	n/a	n/a	all

```
Correctly Classified Instances         530               93.1459 %
Incorrectly Classified Instances        39                6.8541 %
Kappa statistic                          0.8521
Mean absolute error                      0.0683
Root mean squared error                  0.2351
Relative absolute error                 14.6103 %
Root relative squared error             48.6156 %
Total Number of Instances              569

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.887    0.042    0.926      0.887    0.906      0.853    0.981     0.960     M
                 0.958    0.113    0.934      0.958    0.946      0.853    0.981     0.989     B
Weighted Avg.    0.931    0.087    0.931      0.931    0.931      0.853    0.981     0.979

=== Confusion Matrix ===

   a   b   <-- classified as
 188  24 |   a = M
  15 342 |   b = B
```

Even if we try to use the naïve Bayes on our data selection, we still get a higher accuracy rate than them.

Task D

Use of tools

The tools and techniques used in this project were not difficult as Weka allowed us not to have to code the full algorithm along with constant change to the parameters. Its GUI allowed also allowed us to visualise the model. moreover, we could establish other models such as the benchmark and the naïve Bayes which were used in validating our results.

However, the hardest part was raw data analysis and feature selection. This was due to having to use python to find out which features correlate with each other using various methods. Next the visualisation was not simple as they had to be made from scratch with full understanding of the process.

Overall observations

Overall, we set out to see if a tumour could be predicted as benign or malignant. First, we gained a background in the data set, then we use basic data analysis methods to try to see if any correlations exist in doing so, we found 8 of the most corelated. Next, we chose our model and we rand extensive tests to see if we could maximise our data's potential in answering the given problem using a multi layered perceptron. In doing so we found out that the results were extremely good. This was a cause for concern as parameters had been exhausted. But when we used paper 2 to compare, we saw that some of their tests had higher features which was the main cause for concern (over fitting).
However, different subsets of data may provide better results or alternative ways which weren't conducted in the analysis. Next additional data mining tools such as clustering may have benefitted analysis for instance removing records which could be ambiguous in the clusters.

Future work, If the images were available in the data set then software such as openDX and xming would have allowed numerical data to be extracted via an algorithm instead of being reliant on hand drawn borders. This would have provided a sounder data set to analyse. Additional parameters such as weight and age would also have allowed the classification to be applicable in real life.

Appendix

References

(1)
Kaggle.com. (2020). Breast Cancer Wisconsin (Prognostic) Data Set. [online] Available at: https://www.kaggle.com/sarahvch/breast-cancer-wisconsin-prognostic-data-set [Accessed 2 Feb 2020].

(2)

Wolberg, W. (1995). Computerized Breast Cancer Diagnosis and Prognosis From Fine-Needle Aspirates. Archives of Surgery, [online] 130(5), p.30. Available at: https://www.researchgate.net/publication/15451281_Computerized_breast_cancer_diagnosis_and_progno sis_from_fine-needle_aspirates/citation/download [Accessed 1 Feb 2020].

(3)
AlamKhan, R., Ahmad, N. and Minallah, N., 2013. Classification and Regression Analysis of the Prognostic Breast Cancer using Generation Optimizing Algorithms. *International Journal of Computer Applications*, 68(25), pp.42-47.

(4)
Géron, A., 2019. *Hands-On Machine Learning With Scikit-Learn, Keras, And Tensorflow*. Beijing [etc.]: O'Reilly. pp.289

(5)
https://www.cs.waikato.ac.nz/ml/weka/

(6)

Wang, S., 2020. *Recent Developments On Statistical And Neural Network Tools Focusing On Biodiesel Quality*. [online] Issuu. Available at: <https://issuu.com/sep2011--now/docs/1_8afe4a324f8b31> [Accessed 1 May 2020].pg 100

(7)

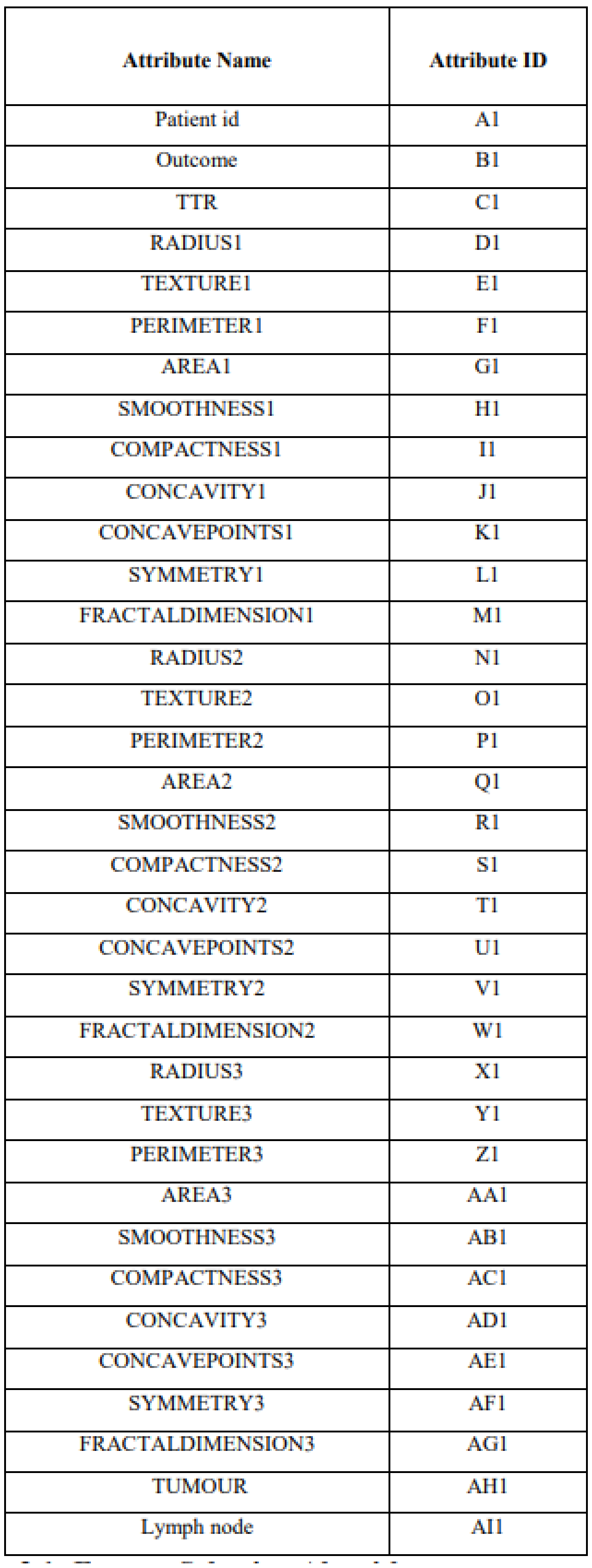
Beg, M. and Jain, M., 2012. An Analysis of The Methods Employed for Breast Cancer Diagnosis. *International Journal of Research in Computer Science*, 2(3), pp.25-29.

(8)

Abbass, H., 2002. An evolutionary artificial neural networks approach for breast cancer diagnosis. *Artificial Intelligence in Medicine*, 25(3), pp.265-281.

Tables

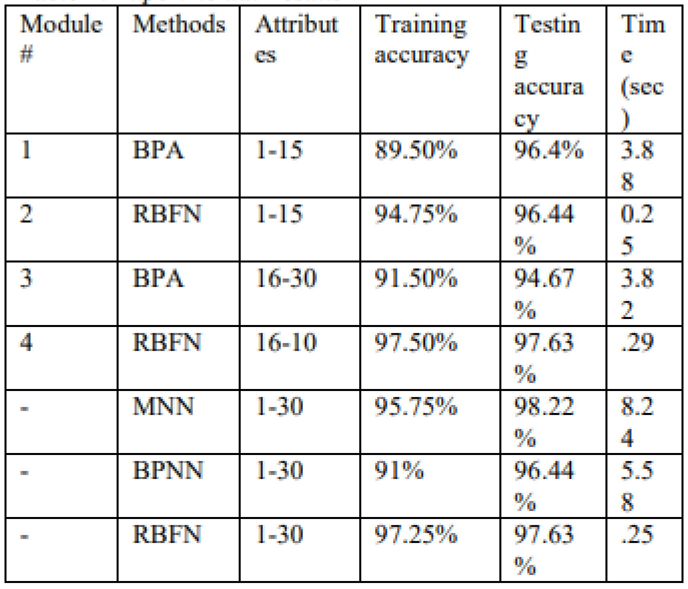
Table 1 Key of feature values in resource 3 comparison. (Taken from paper 2 reference 4)



Attribute Name	Attribute ID
Patient id	A1
Outcome	B1
TTR	C1
RADIUS1	D1
TEXTURE1	E1
PERIMETER1	F1
AREA1	G1
SMOOTHNESS1	H1
COMPACTNESS1	I1
CONCAVITY1	J1
CONCAVEPOINTS1	K1
SYMMETRY1	L1
FRACTALDIMENSION1	M1
RADIUS2	N1
TEXTURE2	O1
PERIMETER2	P1
AREA2	Q1
SMOOTHNESS2	R1
COMPACTNESS2	S1
CONCAVITY2	T1
CONCAVEPOINTS2	U1
SYMMETRY2	V1
FRACTALDIMENSION2	W1
RADIUS3	X1
TEXTURE3	Y1
PERIMETER3	Z1
AREA3	AA1
SMOOTHNESS3	AB1
COMPACTNESS3	AC1
CONCAVITY3	AD1
CONCAVEPOINTS3	AE1
SYMMETRY3	AF1
FRACTALDIMENSION3	AG1
TUMOUR	AH1
Lymph node	AI1

Table 2 (7) – table of results derived from paper 3 (reference 7)

Table 1: Experimental Results



Module #	Methods	Attributes	Training accuracy	Testing accuracy	Time (sec)
1	BPA	1-15	89.50%	96.4%	3.88
2	RBFN	1-15	94.75%	96.44 %	0.25
3	BPA	16-30	91.50%	94.67 %	3.82
4	RBFN	16-10	97.50%	97.63 %	.29
-	MNN	1-30	95.75%	98.22 %	8.24
-	BPNN	1-30	91%	96.44 %	5.58
-	RBFN	1-30	97.25%	97.63 %	.25