Data Mining and Neural Networks Computational Task 1

Luke Emmins – 189025797

**Task 1**

In the article “Computer-derived nuclear features distinguish malignant from benign breast cytology” the authors looked to classify breast cancer cell samples into Malignant or Benign. This classification was not as simple as choosing which class a cell belongs to with 100% accuracy which was why numerous methodologies were used to try and have the highest possible accuracy for classification. Before speaking about each method used individually, it is worth mentioning that the cell samples were classified based upon their features, Radius, Area, etc. The two methods used within this report were a logistic regression validation and MSM-T classification. For both methods, due to a multi-dimensional data set, we need to cross validate to ensure accuracy. The cross validation is achieved by applying a method to 9/10 data sets and seeing how accurate those results are for the 10th data set.

The cells were gathered using fine needle aspiration and then a computer defines nuclear size, and texture features.[[1]](#footnote-1) The features were defined based upon the location of each cell nucleus and typically the higher the value for each feature, the more like the cell sample is Malignant.

For the logistics regression, the article briefly mentions the use of SAS software as well as Systat software. Furthermore, the linear regression was also able to classify the data. This was done by a stepwise logistics regression selection consisting of standard error of the radius, worst radius, worst texture, and worst concave point.[[2]](#footnote-2) The model that was produced from these features meant that they were able to class the cell samples with only 9 Benign and 12.4 Malignant being misclassified.

Then, for the MSM-T method, this used all 30 dimensions of the data. The process involved trying to place a plane in the data set that effectively separated Benign data from Malignant data. If this plane is not possible then MSM-T will try to place a plane so that the misclassified cell samples are minimised. If a plane is produced, then future data can be classified using the plane. The results found in the report were that worst Area, mean Texture and worse smoothness gave the most accurate plane for classification. This method produced only 7 misclassified cell samples for each class.

The article mentions that accuracy of the classification was measured in two ways, by fraction of the number of correctly classified samples out of the total cases, or by using the method mentioned above, classifying for several attributes, and then testing it on the rest to determine how many were correctly classified.

**Task 2**

For task 2, I produced 2 tables that include total, malignant, and benign data in each one. The difference between the tables is that in the second the data has been standardised and normalised to have unit variance and zero mean. As a result of the computers precision, the mean and variance for some data will be close to the desired value. The variations are small enough to ignore and set the values to their respective values. Also, I decided to standardise and normalise all dimensions of data.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Total Data | | | Malignant | | | Benign | | |
| Mean | Std | Var | Mean | Std | Var | Mean | Std | Var |
| Radius | 14.13 | 3.524 | 12.4189 | 17.4628 | 3.2040 | 10.2654 | 12.146 | 1.7805 | 3.170221 |
| Area | 19.29 | 4.301 | 18.4989 | 21.6049 | 3.7795 | 14.2844 | 17.91 | 3.9951 | 15.961 |
| Perimeter | 91.97 | 24.299 | 590.4405 | 115.3654 | 21.8547 | 477.626 | 78.08 | 11.8074 | 139.4156 |
| Texture | 654.9 | 351.91 | 123843.5 | 978.3764 | 367.983 | 135378.3 | 462.79 | 134.2871 | 18033.03 |
| Smoothness | 0.096 | 0.0141 | 0.0001977 | 0.1029 | 0.0126 | 0.000158 | 0.0925 | 0.0134 | 0.000180 |
| Compactness | 0.104 | 0.0528 | 0.0027891 | 0.1452 | 0.054 | 0.002914 | 0.0801 | 0.0337 | 0.0011 |
| Concavity | 0.089 | 0.0797 | 0.0063552 | 0.1608 | 0.075 | 0.00563 | 0.0461 | 0.0434 | 0.0019 |
| Mean Concave Points | 0.049 | 0.0388 | 0.0015056 | 0.0880 | 0.0344 | 0.0012 | 0.0257 | 0.0159 | 0.000253 |
| Symmetry | 0.181 | 0.0274 | 0.0007515 | 0.1929 | 0.0276 | 0.000763 | 0.1742 | 0.0248 | 0.000615 |
| Mean Fractal Dimension | 0.063 | 0.0071 | 0.0000498 | 0.0627 | 0.0076 | 5.735510 | 0.0629 | 0.0067 | 4.552663 |

*Table 1 – Showing the means, standard deviation, and variance of all data before data transformation.*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Total Data | | | Malignant | | | Benign | | |
| Mean | Std | Var | Mean | Std | Var | Mean | Std | Var |
| Radius | 0 | 1 | 1 | 0.9465 | 0.9092 | 0.8266 | -0.562 | 0.505 | 0.2553 |
| Area | 0 | 1 | 1 | 0.5383 | 0.8787 | 0.7721 | -0.319 | 0.9288 | 0.8628 |
| Perimeter | 0 | 1 | 1 | 0.9629 | 0.8994 | 0.8089 | -0.517 | 0.4859 | 0.2361 |
| Texture | 0 | 1 | 1 | 0.9192 | 1.0455 | 1.0931 | -0.546 | 0.3816 | 0.1456 |
| Smoothness | 0 | 1 | 1 | 0.4649 | 0.8965 | 0.8037 | -0.276 | 0.9561 | 0.9140 |
| Compactness | 0 | 1 | 1 | 0.7734 | 1.0222 | 1.04498 | -0.459 | 0.6390 | 0.4084 |
| Concavity | 0 | 1 | 1 | 0.9029 | 0.9410 | 0.8856 | -0.536 | 0.5449 | 0.2969 |
| Mean Concave Points | 0 | 1 | 1 | 1.0069 | 0.8858 | 0.7847 | -0.597 | 0.4099 | 0.1681 |
| Symmetry | 0 | 1 | 1 | 0.4285 | 1.0082 | 1.0164 | -0.254 | 0.9048 | 0.8188 |
| Mean Fractal Dimension | 0 | 1 | 1 | -0.017 | 1.0727 | 1.1506 | 0.0098 | 0.9556 | 0.9132 |

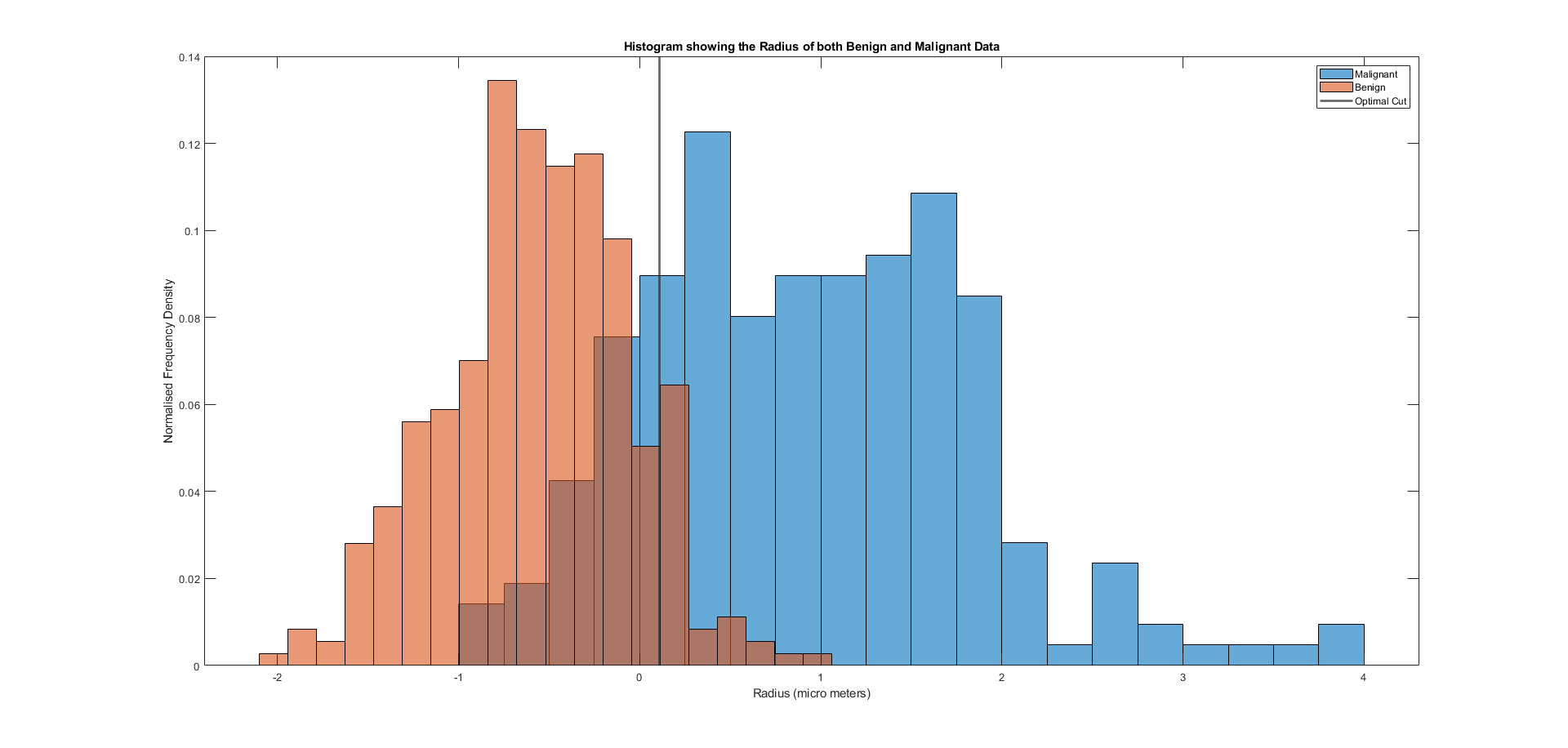
*Table 2 – Showing the means, standard deviation, and variance of all data after data transformation.*

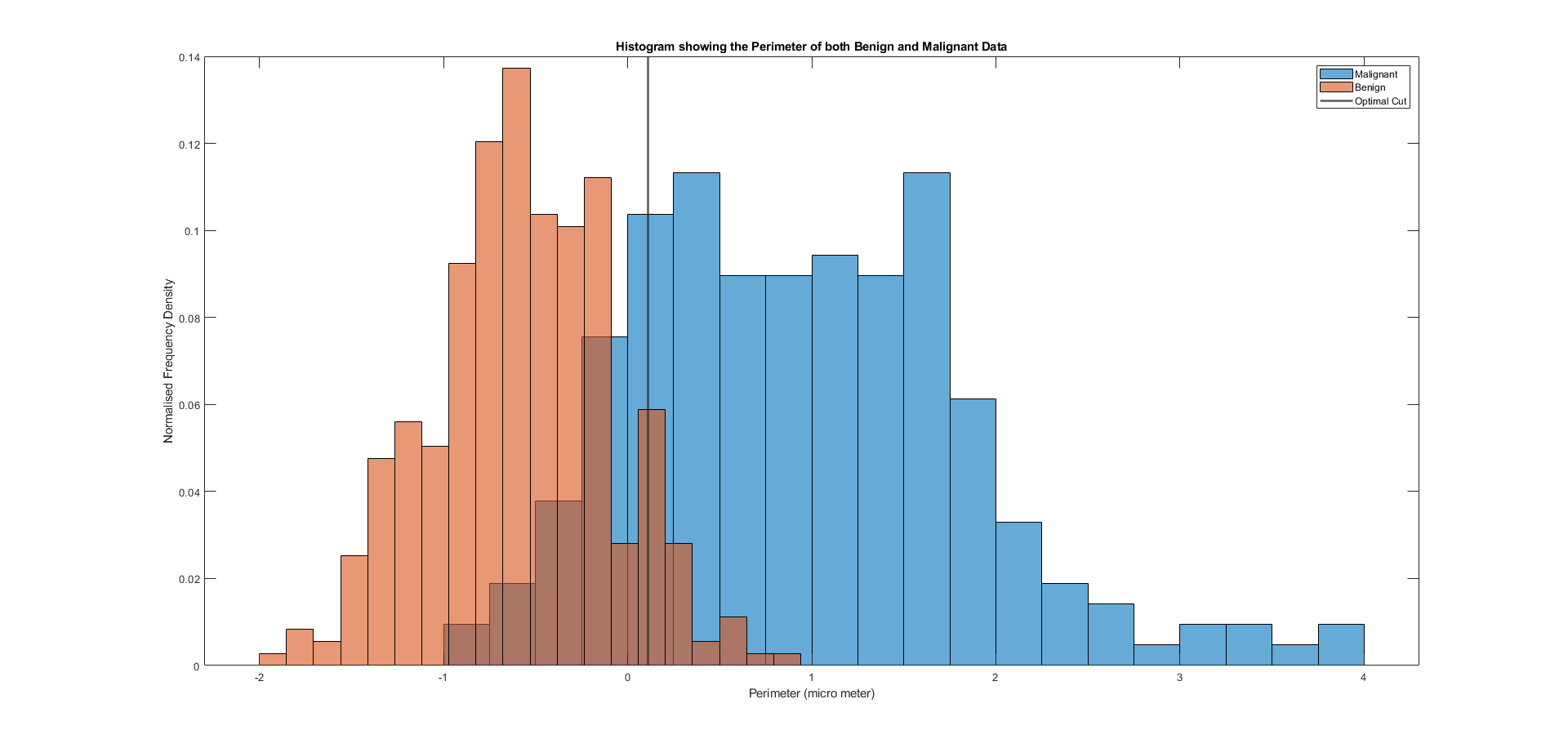
In the data set[[3]](#footnote-3), there was 569 different cases for which there was 30 different attributes. The 30 attributes were made up of 10 real attributes, the standard error of these 10 real attributes and the worst of them. I chose to use the real attributes and exclude the standard error and worst of each attribute. Within the 569 cases there were 357 Benign cases and 212 Malignant cases.

The code that was used to produce the data for these tables will be attached in the appendix of this report.

**Task 3**

In this section, the aim was to create a predictor that was able to find the optimal cut on a histogram so that you could classify the different samples. The method in finding the optimal cut was to find the value such that the misclassification was at its lowest. Below, are the 10 histograms for all real attributes with the optimal cut shown by the vertical line. The histograms will use the normalised/standardised data because the histograms will best represent a distribution because of the property that comes with it. The sum of all data sums to 1 which represents a probability distribution.

Chart, histogram

Description automatically generatedChart, histogram

Description automatically generatedChart, histogram

Description automatically generatedChart, histogram

Description automatically generatedChart, histogram

Description automatically generatedChart, histogram

Description automatically generatedChart, histogram

Description automatically generatedChart, histogram

Description automatically generated

Now that all histograms have been presented, it was time to test the accuracy of this optimal cut for each attribute. The optimal cut was found by finding the point along the x-axis such that the misclassification was at its lowest. The method used to find this was to count the number of samples that were correctly classified within each bin of the histogram.

Below is the table that shows the optimal cut and then the percentages of correctly classified values for both the Malignant and Benign cases.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Optimal cut value | Malignant Accuracy | Benign Accuracy |
| Radius | 0.1107 | 81.13% | 90.48% |
| Area | 0.3673 | 55.19% | 81.51% |
| Perimeter | 0.2671 | 75.47% | 97.2% |
| Texture | 0.1090 | 76.42% | 96.92% |
| Smoothness | 0.1072 | 63.68% | 67.23% |
| Compactness | 0.3234 | 64.15% | 89.36% |
| Concavity | 0.1968 | 77.36% | 94.12% |
| Mean Concave Points | 0.1142 | 87.26% | 93.0% |
| Symmetry | 0.0861 | 59.43% | 67.23% |
| Mean Fractal Dimension | 0.3111 | 33.02% | 71.15% |

*Table 3 – shows the optimal cuts and the classification accuracy for each of the classes.*

From the table, by quick inspection, the Mean Concave points attribute classified the data the best whilst the Mean Fractal Dimension attribute miss classified the data the most.

Quick inspection is not enough to determine how well this method can classify the data, so I have arranged the attributes in order of their classification success. I do this by taking an average of the percentages for each attribute.

|  |  |
| --- | --- |
| Average Classification % | Attribute |
| 90.13% | Mean Concave Points |
| 86.67% | Texture |
| 86.34% | Perimeter |
| 85.8% | Radius |
| 85.74% | Compactness |
| 76.75% | Compactness |
| 68.35% | Area |
| 65.45% | Smoothness |
| 63.33% | Symmetry |
| 52.08% | Mean Fractal Dimension |

The article in the first task suggests that classification that is under 70% is suspicious, therefore because of this there are four attributes that are deemed suspicious under this definition.

**Task 4**

In task 4, the k-th nearest neighbour algorithm was needed in attempt to better classify the data. 1NN and 3NN were the rules that needed to be tested. The 1NN classified the test sample using the piece of data that was closest to it where as the 3NN looked at the 3 closest pieces of data to it and took the most common as the classification. The distance was calculated using the square root of all attributes squared, as seen in the code attached. The code for both 1NN and 3NN are very similar and both calculate the sensitivity.

For the 1NN rule, the code produced a success rate of 91.21% and the 3NN rule produced a success rate of 93.67%. This result is interesting because one can conclude that in the data somewhere there is a plane that can separate the data with high accuracy. This is because if more than 90% of data has its closest neighbour being in the same class, then the data from each class must be clumped together. This will prove to be true for most data, and not all, .

The kth nearest neighbour algorithm is one that can be very reliable with large datasets. This is especially apparent in the 1NN rule because it guarantees an error of no worse than twice the Bayes error rate.[[4]](#footnote-4)

Furthermore, this method yields a better outcome in comparison to the optimal cut method and would only get better if I were to exclude the data that had poor classification in the previous task.

**Task 5**

The main idea behind Fisher’s Linear Discriminant is to have a D-dimensional data set and to project it in such as was to a lower dimension, D’, such that a separation can be made. This separation is known as the threshold.

Fisher’s Linear Discriminant handles K number of classes and has different methodologies for K = 2 and K > 2. In this report I will explain how Fisher’s Linear Discriminant is used for 2 classes because this is what is used for the data I used.

The algorithm works as follows; you are given as full dataset, complete with labels, and you need to calculate the projection matrix, the direction of the projection, and the threshold that allows you to best distinguish the differences. To better explain the importance of the direction, the diagram below best shows this:

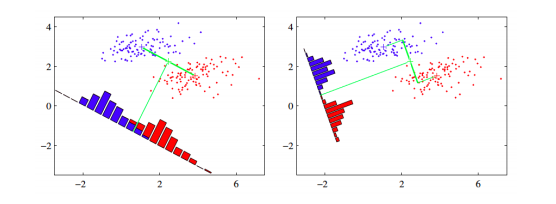


Figure 1: Pattern Recognition and Machine Learning (Christopher M. Bishop)

The calculations that are needed are the mean vectors for each attribute, the variances for each attribute and the covariance matrices. These are needed for the full data, Malignant data, and Benign data. This is just for the example of this report. The initial idea is to find a vector that links the means of each class together so that they can be projected into one dimension along this vector.[[5]](#footnote-5)

The projection needs to be selected in such a way that the class separation is maximised. Fisher’s Linear Discriminant does this by maximising the ratio between the between-class variance to the within-class variance.[[6]](#footnote-6)

To calculate the projection matrix, you must find the inverse of the covariance matrix for all of the data and times it by the difference in class means.

Where: covariance matrix, = mean, and projection matrix.

Then, threshold value that best separates the classes can be found by the following equation:

This leads you to the final step of the algorithm, to project the original data into one dimension so that it can be classified using the threshold value. Again, this was done using the following equation:

Where: the one-dimensional value after projection, the transpose of the projection matrix, and the original data set.

Once you have the final values, if they are then they belong to one class, and if they are then they belong to the other.

One final remark on this, a clear separation will not always be found. In the event of this the linear discriminant will find the best separation.

**Task 6**

In the final task, I had to apply this to the data set that was used in previous tasks. After using Matlab to find the projection matrix as well as the threshold value, I took the original data and projected into singular values so that they could be compared to the threshold value. As previously stated, if the value is higher than it tends to be a Malignant sample thus, if the data was then it was Malignant and Benign for any other value. The method produced a correct classification rate of 93.67%, misclassifying only 36 samples out of the 569 samples.

Finally, the accuracy for both the 3NN and Fisher’s Linear Discriminant methods provided the same percentage of 93.67%. This reinforces the idea previously mentioned that there was a plane that could separate majority of data so that misclassification was small. Fishers’ Linear Discriminant was able to project the data into one dimension and find that plane which was why, to a much higher decimal place, the accuracies were so close. However, Fisher’s Linear Discriminant provided a more accurate classification than the 1NN method, but only marginally. If the data set were to be significantly larger, the 1NN method would likely have proved a more accurate result. This is because when the data is projected using FLD, with a sufficiently large dataset, it would be harder because there will be potentially a lot more overlaps.

**References**

Wolberg, W., Street, W., Heisey, D. and Mangasarian, O., 1995. Computer-derived nuclear features distinguish malignant from benign breast cytology. *Human Pathology*. Available at: <https://dollar.biz.uiowa.edu/~nstreet/research/hu\_path95/hp95.pdf> [Accessed 10 February 2021].

Donald Bren School of Information and Computer sciences, Breast cancer dataset. Available at <<http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancerwisconsin/wdbc.data>> [Accessed 12 February]

En.wikipedia.org. 2021. K-nearest neighbours algorithm. Available at: <https://en.wikipedia.org/wiki/K-nearest\_neighbors\_algorithm> [Accessed 14 February 2021].

Silva, T., 2021. An illustrative introduction to Fisher's Linear Discriminant - Thalles' blog. Available at: <https://sthalles.github.io/fisher-linear-discriminant/> [Accessed 14 February 2021].

BISHOP, C., 2016. PATTERN RECOGNITION AND MACHINE LEARNING textbook. SPRINGER-VERLAG NEW YORK, pp.186-189. Available at < http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Recognition%20And%20Machine%20Learning%20-%20Springer%20%202006.pdf> [Accessed 15 February 2021]

En.wikipedia.org. 2021. Linear discriminant analysis. Available at: <https://en.wikipedia.org/wiki/Linear\_discriminant\_analysis> [Accessed 16 February 2021].

**Appendix**

(File name: TotalDataNormalisation.m)

load('Malignant.mat');

load('Benign.mat');

load('FullData.mat');

%Removal of ID number

FullData(:,1) = [];

%Changing M -> 1 and B -> 2

label = table2array(FullData(:,1));

for i = 1:size(FullData(:,1),1)

if label(i) == 'M'

label2(i) = 1;

else

label2(i) = 2;

end

end

label = label2(:);

FullData(:,1) = [];

A = array2table(label);

FullData = [A FullData];

%normalise all REAL attributes

for i = 2:11

for j = 1:size(FullData,1)

NewFullData(i-1,j) = (FullData{j,i} - min(FullData{:,i}))/(max(FullData{:,i}) - min(FullData{:,i}));

end

end

NewFullData = transpose(NewFullData);

%Standardise all REAL attributes

for i = 2:11

for k = 1:size(FullData,1)

FinalFullData(k,i-1) = (NewFullData(k,i-1) - mean(NewFullData(:,i-1)))/std(NewFullData(:,i-1));

end

end

FinalFullData = [label FinalFullData];

%some weird issue caused FullData to be a 569 square matrix

%Separate the Benign data and Malignant Data

MalignantData = [];

BenignData = [];

for i = 1:length(label)

if FinalFullData(i,1) == 1

MalignantData = [MalignantData; FinalFullData(i,:)];

else

BenignData = [BenignData; FinalFullData(i,:)];

end

end

%mean and std for each attribute after data transform

for i = 2:11

MeanBenign(i-1) = mean(BenignData(:,i));

MeanMalignant(i-1) = mean(MalignantData(:,i));

StdBenign(i-1) = std(BenignData(:,i));

StdMalignant(i-1) = std(MalignantData(:,i));

end

%mean and std for each attribute before data transform

for i = 2:11

MeanBenign(2,i-1) = mean(Benign{:,i+1});

MeanMalignant(2,i-1) = mean(Benign{:,i+1});

StdBenign(2,i-1) = std(Malignant{:,i+1});

StdMalignant(2,i-1) = std(Malignant{:,i+1});

end

(File name: NormalisedHistograms.m)

%Histogram for each attribute, normalised

load('BenignData.mat')

load('MalignantData.mat')

[N, edges] = histcounts(BenignData(:,2),50,'Normalization','probability');

for i = 1:length(edges)

Count = 0;

for j = 1:length(BenignData(:,2))

if BenignData(j,2) > edges(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

for k = 1:length(MalignantData(:,2))

if MalignantData(k,2) <= edges(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

Misclassification(i) = Count/569;

end

a1 = min(Misclassification);

figure(1)

histogram(MalignantData(:,2),20,'Normalization','probability')

hold on

histogram(BenignData(:,2),20,'Normalization','probability')

legend('Malignant','Benign')

xline(a1, 'LineWidth',2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[N, edges2] = histcounts(BenignData(:,3),50,'Normalization','probability');

for i = 1:length(edges2)

Count = 0;

for j = 1:length(BenignData(:,3))

if BenignData(j,3) > edges2(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

for k = 1:length(MalignantData(:,3))

if MalignantData(k,3) <= edges2(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

Misclassification2(i) = Count/569;

end

a2 = min(Misclassification2);

figure(2)

histogram(MalignantData(:,3),20,'Normalization','probability')

hold on

histogram(BenignData(:,3),20,'Normalization','probability')

legend('Malignant','Benign')

xline(a2, 'LineWidth',2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[N, edges3] = histcounts(BenignData(:,4),50,'Normalization','probability');

for i = 1:length(edges3)

Count = 0;

for j = 1:length(BenignData(:,4))

if BenignData(j,4) > edges3(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

for k = 1:length(MalignantData(:,4))

if MalignantData(k,4) <= edges3(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

Misclassification3(i) = Count/569;

end

a3 = min(Misclassification3);

figure(3)

histogram(MalignantData(:,4),20,'Normalization','probability')

hold on

histogram(BenignData(:,4),20,'Normalization','probability')

legend('Malignant','Benign')

xline(a3, 'LineWidth',2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[N, edges4] = histcounts(BenignData(:,5),50,'Normalization','probability');

for i = 1:length(edges4)

Count = 0;

for j = 1:length(BenignData(:,5))

if BenignData(j,5) > edges4(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

for k = 1:length(MalignantData(:,5))

if MalignantData(k,5) <= edges4(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

Misclassification4(i) = Count/569;

end

a4 = min(Misclassification4);

figure(4)

histogram(MalignantData(:,5),20,'Normalization','probability')

hold on

histogram(BenignData(:,5),20,'Normalization','probability')

legend('Malignant','Benign')

xline(a4, 'LineWidth',2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[N, edges5] = histcounts(BenignData(:,6),50,'Normalization','probability');

for i = 1:length(edges5)

Count = 0;

for j = 1:length(BenignData(:,6))

if BenignData(j,6) > edges5(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

for k = 1:length(MalignantData(:,6))

if MalignantData(k,6) <= edges5(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

Misclassification5(i) = Count/569;

end

a5 = min(Misclassification5);

figure(5)

histogram(MalignantData(:,6),20,'Normalization','probability')

hold on

histogram(BenignData(:,6),20,'Normalization','probability')

legend('Malignant','Benign')

xline(a5, 'LineWidth',2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[N, edges6] = histcounts(BenignData(:,7),50,'Normalization','probability');

for i = 1:length(edges6)

Count = 0;

for j = 1:length(BenignData(:,7))

if BenignData(j,7) > edges6(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

for k = 1:length(MalignantData(:,7))

if MalignantData(k,7) <= edges6(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

Misclassification6(i) = Count/569;

end

a6 = min(Misclassification6);

figure(6)

histogram(MalignantData(:,7),20,'Normalization','probability')

hold on

histogram(BenignData(:,7),20,'Normalization','probability')

legend('Malignant','Benign')

xline(a6, 'LineWidth',2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[N, edges7] = histcounts(BenignData(:,8),50,'Normalization','probability');

for i = 1:length(edges7)

Count = 0;

for j = 1:length(BenignData(:,8))

if BenignData(j,8) > edges7(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

for k = 1:length(MalignantData(:,8))

if MalignantData(k,8) <= edges7(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

Misclassification7(i) = Count/569;

end

a7 = min(Misclassification7);

figure(7)

histogram(MalignantData(:,8),20,'Normalization','probability')

hold on

histogram(BenignData(:,8),20,'Normalization','probability')

legend('Malignant','Benign')

xline(a7, 'LineWidth',2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[N, edges8] = histcounts(BenignData(:,9),50,'Normalization','probability');

for i = 1:length(edges8)

Count = 0;

for j = 1:length(BenignData(:,9))

if BenignData(j,9) > edges8(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

for k = 1:length(MalignantData(:,9))

if MalignantData(k,9) <= edges8(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

Misclassification8(i) = Count/569;

end

a8 = min(Misclassification8);

figure(8)

histogram(MalignantData(:,9),20,'Normalization','probability')

hold on

histogram(BenignData(:,9),20,'Normalization','probability')

legend('Malignant','Benign')

xline(a8, 'LineWidth',2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[N, edges9] = histcounts(BenignData(:,10),50,'Normalization','probability');

for i = 1:length(edges9)

Count = 0;

for j = 1:length(BenignData(:,10))

if BenignData(j,10) > edges9(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

for k = 1:length(MalignantData(:,10))

if MalignantData(k,10) <= edges9(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

Misclassification9(i) = Count/569;

end

a9 = min(Misclassification9);

figure(9)

histogram(MalignantData(:,10),20,'Normalization','probability')

hold on

histogram(BenignData(:,10),20,'Normalization','probability')

legend('Malignant','Benign')

xline(a9, 'LineWidth',2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[N, edges10] = histcounts(BenignData(:,11),50,'Normalization','probability');

for i = 1:length(edges10)

Count = 0;

for j = 1:length(BenignData(:,11))

if BenignData(j,11) > edges10(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

for k = 1:length(MalignantData(:,11))

if MalignantData(k,11) <= edges10(i)

Count = Count + 1;

else

Count = Count + 0;

end

end

Misclassification10(i) = Count/569;

end

a10 = min(Misclassification10);

figure(10)

histogram(MalignantData(:,11),20,'Normalization','probability')

hold on

histogram(BenignData(:,11),20,'Normalization','probability')

legend('Malignant','Benign')

xline(a10, 'LineWidth',2)

(File Name: ClassificationError.m)

%This is using the normalised Data.

load('MalignantData.mat')

load('BenignData.mat')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Radius %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Radius\_acc = sum(MalignantData(:,2) > 0.1107)/length(MalignantData(:,2));

B\_Radius\_acc = sum(BenignData(:,2) <= 0.1107)/length(BenignData(:,2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Area %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Area\_acc = sum(MalignantData(:,3) > 0.3673)/length(MalignantData(:,3));

B\_Area\_acc = sum(BenignData(:,3) <= 0.3673)/length(BenignData(:,3));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Perimeter %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Perimeter\_acc = sum(MalignantData(:,4) > 0.2671)/length(MalignantData(:,4));

B\_Perimeter\_acc = sum(BenignData(:,4) <= 0.2671)/length(BenignData(:,4));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Texture %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Texture\_acc = sum(MalignantData(:,5) > 0.1090)/length(MalignantData(:,5));

B\_Texture\_acc = sum(BenignData(:,5) <= 0.1090)/length(BenignData(:,5));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Smoothness %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Smoothness\_acc = sum(MalignantData(:,6) > 0.1072)/length(MalignantData(:,7));

B\_Smoothness\_acc = sum(BenignData(:,6) <= 0.1072)/length(BenignData(:,7));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Compactness %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Compactness\_acc = sum(MalignantData(:,7) > 0.3234)/length(MalignantData(:,8));

B\_Compactness\_acc = sum(BenignData(:,7) <= 0.3234)/length(BenignData(:,8));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Concavity %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Concavity\_acc = sum(MalignantData(:,8) > 0.1968)/length(MalignantData(:,9));

B\_Concavity\_acc = sum(BenignData(:,8) <= 0.1968)/length(BenignData(:,9));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Concave %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Concave\_acc = sum(MalignantData(:,9) > 0.1142)/length(MalignantData(:,9));

B\_Concave\_acc = sum(BenignData(:,9) <= 0.1142)/length(BenignData(:,9));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Symmetry %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Symmetry\_acc = sum(MalignantData(:,10) > 0.0861)/length(MalignantData(:,9));

B\_Symmetry\_acc = sum(BenignData(:,10) <= 0.0861)/length(BenignData(:,9));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Fractal %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Fractal\_acc = sum(MalignantData(:,11) > 0.3111)/length(MalignantData(:,9));

B\_Fractal\_acc = sum(BenignData(:,11) <= 0.3111)/length(BenignData(:,9));

Malignant\_Data\_acc = [M\_Radius\_acc;M\_Area\_acc;M\_Perimeter\_acc;M\_Texture\_acc;M\_Smoothness\_acc;M\_Compactness\_acc;M\_Concavity\_acc;M\_Concave\_acc;M\_Symmetry\_acc;M\_Fractal\_acc];

Benign\_Data\_acc = [B\_Radius\_acc;B\_Area\_acc;B\_Perimeter\_acc;B\_Texture\_acc;B\_Smoothness\_acc;B\_Compactness\_acc;B\_Concavity\_acc;B\_Concave\_acc;B\_Symmetry\_acc;B\_Fractal\_acc];

Averages\_Class = {'Radius', 'Area', 'Perimeter', 'Texture', 'Smoothness', 'Compactness', 'Concavity', 'Mean Concave Points','Symmetry', 'Mean Fractal Dimension'};

Classification\_Error\_Table = table(Malignant\_Data\_acc,Benign\_Data\_acc,'RowNames', Averages\_Class);

(File Name: code1NN.m)

%This is using the normalised Data.

load('MalignantData.mat')

load('BenignData.mat')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Radius %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Radius\_acc = sum(MalignantData(:,2) > 0.1107)/length(MalignantData(:,2));

B\_Radius\_acc = sum(BenignData(:,2) <= 0.1107)/length(BenignData(:,2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Area %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Area\_acc = sum(MalignantData(:,3) > 0.3673)/length(MalignantData(:,3));

B\_Area\_acc = sum(BenignData(:,3) <= 0.3673)/length(BenignData(:,3));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Perimeter %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Perimeter\_acc = sum(MalignantData(:,4) > 0.2671)/length(MalignantData(:,4));

B\_Perimeter\_acc = sum(BenignData(:,4) <= 0.2671)/length(BenignData(:,4));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Texture %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Texture\_acc = sum(MalignantData(:,5) > 0.1090)/length(MalignantData(:,5));

B\_Texture\_acc = sum(BenignData(:,5) <= 0.1090)/length(BenignData(:,5));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Smoothness %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Smoothness\_acc = sum(MalignantData(:,6) > 0.1072)/length(MalignantData(:,7));

B\_Smoothness\_acc = sum(BenignData(:,6) <= 0.1072)/length(BenignData(:,7));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Compactness %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Compactness\_acc = sum(MalignantData(:,7) > 0.3234)/length(MalignantData(:,8));

B\_Compactness\_acc = sum(BenignData(:,7) <= 0.3234)/length(BenignData(:,8));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Concavity %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Concavity\_acc = sum(MalignantData(:,8) > 0.1968)/length(MalignantData(:,9));

B\_Concavity\_acc = sum(BenignData(:,8) <= 0.1968)/length(BenignData(:,9));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Concave %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Concave\_acc = sum(MalignantData(:,9) > 0.1142)/length(MalignantData(:,9));

B\_Concave\_acc = sum(BenignData(:,9) <= 0.1142)/length(BenignData(:,9));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Symmetry %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Symmetry\_acc = sum(MalignantData(:,10) > 0.0861)/length(MalignantData(:,9));

B\_Symmetry\_acc = sum(BenignData(:,10) <= 0.0861)/length(BenignData(:,9));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Fractal %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_Fractal\_acc = sum(MalignantData(:,11) > 0.3111)/length(MalignantData(:,9));

B\_Fractal\_acc = sum(BenignData(:,11) <= 0.3111)/length(BenignData(:,9));

Malignant\_Data\_acc = [M\_Radius\_acc;M\_Area\_acc;M\_Perimeter\_acc;M\_Texture\_acc;M\_Smoothness\_acc;M\_Compactness\_acc;M\_Concavity\_acc;M\_Concave\_acc;M\_Symmetry\_acc;M\_Fractal\_acc];

Benign\_Data\_acc = [B\_Radius\_acc;B\_Area\_acc;B\_Perimeter\_acc;B\_Texture\_acc;B\_Smoothness\_acc;B\_Compactness\_acc;B\_Concavity\_acc;B\_Concave\_acc;B\_Symmetry\_acc;B\_Fractal\_acc];

Averages\_Class = {'Radius', 'Area', 'Perimeter', 'Texture', 'Smoothness', 'Compactness', 'Concavity', 'Mean Concave Points','Symmetry', 'Mean Fractal Dimension'};

Classification\_Error\_Table = table(Malignant\_Data\_acc,Benign\_Data\_acc,'RowNames', Averages\_Class);

(File Name: code3NN.m)

clear

load('FinalFullData.mat');

DataSet = FinalFullData;

X = DataSet(:,2:11);

Y = DataSet(:,1);

%X = FullData{:,2:11};

%Y = FullData{:,1};

N = size(X,1);

for i = 1:N

x = X(i,:);

dist = sum((X-x).^2,2);

[~,ind] = sort(dist);

Classification1(i) = mode(Y(ind(2:4)));

end

Actual = Y(:);

Classification1 = Classification1(:);

out = table(Actual, Classification1);

%percentage of correctly classified

Correct = 0;

for i = 1:N

if out{i,1} == out{i,2}

Correct = Correct + 1;

else

Correct = Correct + 0;

end

end

Correctly\_Classified\_3NN = Correct/N

(File Name: FishersDiscriminant.m)

load('MalignantData.mat')

load('BenignData.mat')

load('FinalFullData.mat')

%mu0 is malignant and mu1 is benign

m0 = mean(MalignantData(:,2:11));

m1 = mean(BenignData(:,2:11));

S0 = cov(MalignantData(:,2:11));

S1 = cov(BenignData(:,2:11));

w = (S0+S1)\transpose((m0+m1));

%threshold

c = w' \* 1/2 \* (m0 + m1)';

x = FinalFullData(:,2:11);

for n = 1:569

y = x(n,:)\*w;

if y >= c

Classification(n) = 1; %Malignant

else

Classification(n) = 2; %Benign

end

end

Comparison = [FinalFullData(:,1) transpose(Classification)];

%error

Correct = 0;

for i = 1:569

if Comparison(i,1) == Comparison(i,2)

Correct = Correct + 1;

else

Correct = Correct + 0;

end

end

Correctly\_Classified\_Fisher = Correct/569

1. W.H. Wolberg, W.N. Street, D.M. Heisey, and O.L. Mangasarian Paper [↑](#footnote-ref-1)
2. W.H. Wolberg, W.N. Street, D.M. Heisey, and O.L. Mangasarian Paper [↑](#footnote-ref-2)
3. Archive Data Set [↑](#footnote-ref-3)
4. K-nearest neighbors algorithm [↑](#footnote-ref-4)
5. Fisher’s Linear Discriminant Thalles’ blog [↑](#footnote-ref-5)
6. Pattern Recognition and Machine Learning textbook [↑](#footnote-ref-6)