Machine Learning with Fetal Health dataset

Lee Kai Yang
kai.y.lee@ucdconnect.ie

October 29, 2023

1 Data Preparation / Cleaning

1.1 Normalization

Since the features in the dataset has different scales, I applied a *Normalize* filter with scale of 100, so every data falls on the range $[0, 100]$.

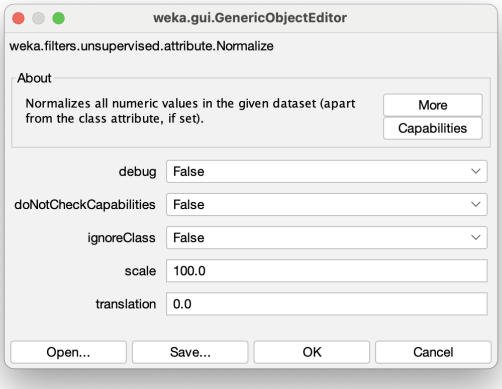

Figure 1: Normalization

1.2 Feature removal

Then I manually removed the feature *severe_decelerations* because it is mostly 0 except for once it is 0.001, and that could introduce noise.

2 Classify with k-NN, Naive Bayes and Decision Tree

2.1 Choosing the accuracy measures

The accuracy measures I chose is **F1-Measure**. This is due to the fact that F1-Measure is a combination of both precision and recall, it reflects the true positive rate and false positive rate of the classifiers' performance. Because of that this measure tells us a more complete picture of each classifier's pros and cons compared to other measures such as accuracy. Hence, it is easier for us to compare and determine the trade-offs between the classifiers.

To clarify, Weka labels it as `F-Measure` but it is the same as F1-Measure because it uses harmonic mean ($\beta = 1$) in the calculation of:

$$F = \frac{(1+\beta^2) \cdot Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$$

2.2 Discussion on the different accuracy values

By running the classifiers with default parameters by Weka, the results are as follows:

F1-Measure	Naive Bayes	k-NN	Decision Tree
Normal	0.900	0.955	0.958
Pathologic	0.603	0.860	0.936
Suspect	0.621	0.717	0.774
Weighted Average	0.837	0.915	0.931

Table 1: F1-Measure of each class and the weighted average

The **weighted average** is the average of the F1-Measure of each class weighted by the number of instances in each class. The formula is as follows:

$$F1_{weighted} = \frac{F1_{normal} \times n_{normal} + F1_{pathologic} \times n_{pathologic} + F1_{suspect} \times n_{suspect}}{n_{normal} + n_{pathologic} + n_{suspect}}$$

From the results above, Naive Bayes is the worst performing classifier and this is because the features in the dataset depends on each other for example, when there are lesser **accelerations**, there are also lesser **fetal_movement** whereas Decision Tree performed the best with 0.931 and k-NN being the close second with 0.915.

2.3 Finding the optimal value of k for k-NN

The table below shows the weighted average F1-Measure by running the k-NN classifier with odd values of k from 1 to 15.

k	F1-Measure (weighted average)
1	0.915
3	0.907
5	0.905
7	0.899
9	0.894
11	0.889
13	0.891
15	0.889

Table 2: F1-Measure of k-NN from $k = 1$ to $k = 15$

From the results above, we can observe a downtrend as k increases and the highest F1-Measure recorded is when $k = 1$. However, I would consider $k = 3$ to be the optimal value because generally overfits occur when $k = 1$. In addition, I purposely skipped even number of k to avoid ties.

2.4 Using 1/d weighting

Using only $k = 3$, I applied 1/d weighting to the k-NN classifier and the results are as follows:

F1-Measure (No distance weighting)	F1-Measure (Weight by 1/distance)
0.907	0.913

Table 3: F1-Measure of k-NN with 1/d weighting where $k = 3$

Note that the F1-Measure above is the weighted average. From the results above, we do see a slight improvement by applying 1/d weighting to the k-NN classifier.

2.5 Using $k = 3$ and $k = 5$ fold-cross validation

The table below shows the weighted average F1-Measure by running the k-NN classifier with $k = 10$ (default), $k = 5$ and $k = 3$ fold-cross validation.

k fold-cross validation	F1-Measure (Weight by 1/distance)
10	0.913
5	0.912
3	0.908

Table 4: F1-Measure of k-NN with varying k fold-cross validation

From the results above, we can observe that the results does change with different value k fold-cross validation because when k fold-cross validation decreases, F1-Measure also decreases.

3 Receiver operating characteristic (ROC) curves

3.1 ROC Curves for the classification models

3.1.1 Naive Bayes

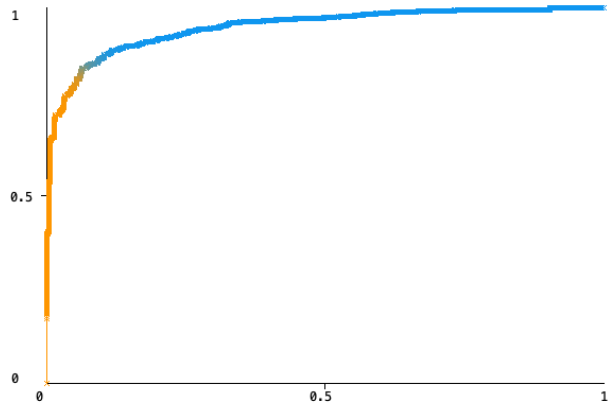

Figure 2: Naive Bayes (Normal)

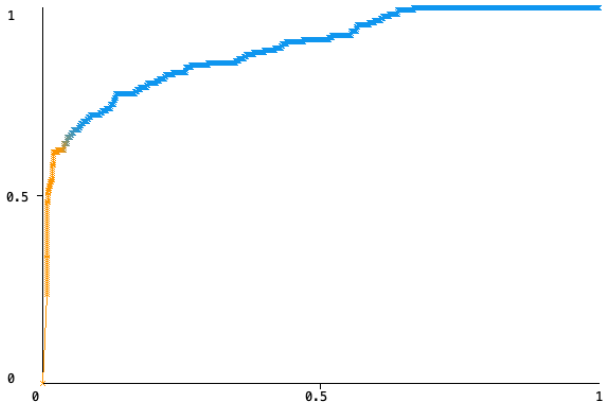

Figure 3: Naive Bayes (Pathological)

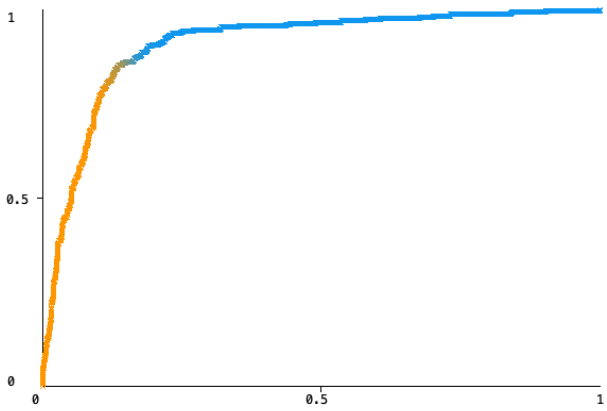

Figure 4: Naive Bayes (Suspect)

3.1.2 k-NN

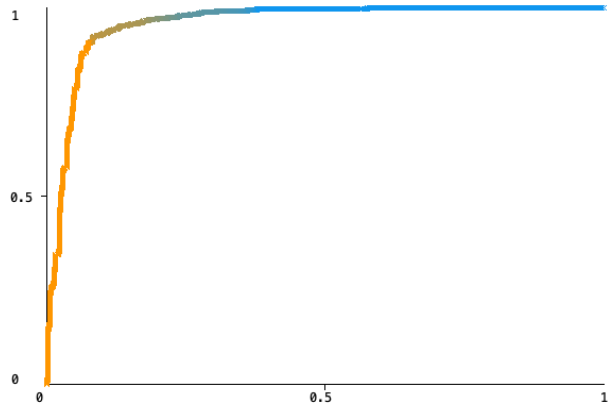

Figure 5: k-NN (Normal)

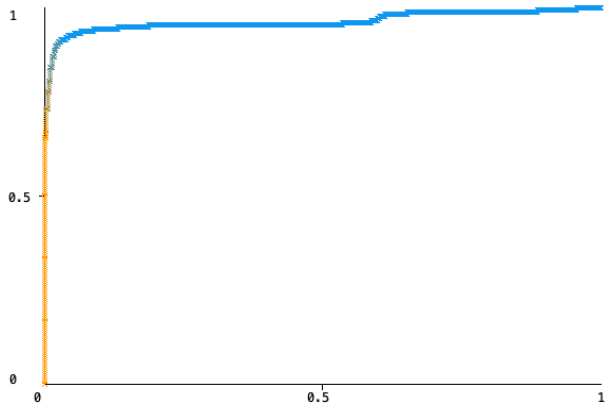

Figure 6: k-NN (Pathological)

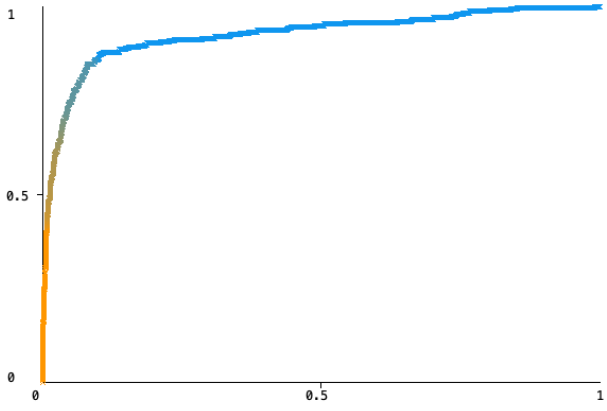

Figure 7: k-NN (Suspect)

3.1.3 Decision Tree

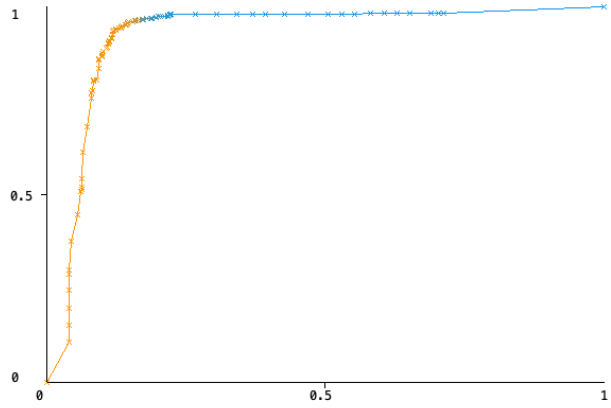

Figure 8: Decision Tree (Normal)

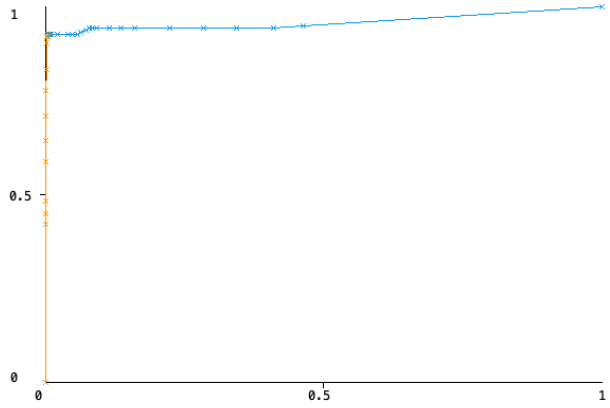

Figure 9: Decision Tree (Pathological)

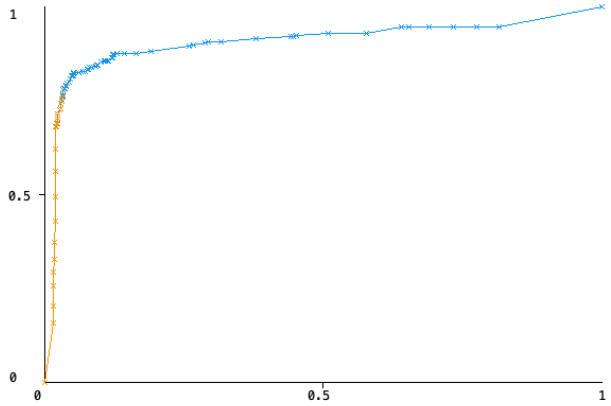

Figure 10: Decision Tree (Suspect)

3.2 Learnings from the ROC curves

Since ROC curves are plotted using the true positive rate (TPR) against the false positive rate (FPR), it helps us to visualize the performance of the classifiers at different thresholds and ideally we would want the curve to be above the "random" reference line and as close to the top left corner as possible because the top left corner represents the best case where $TPR = 1.0$ and $FPR = 0.0$ meaning the model has no false positive results. One way to measure this is by calculating the area under the ROC curve (AUC), the closer the AUC is to 1.0, the better the model is and for the curve to be above the "random" reference line, the AUC must be more than 0.5. From the ROC curves above, we can deduce the area under the ROC curve (AUC) for each classifier:

Classifier	Area under the ROC curve (AUC)
Naive Bayes	0.947 (Normal), 0.888 (Pathological), 0.907 (Suspect)
k-NN	0.957 (Normal), 0.962 (Pathological), 0.923 (Suspect)
Decision Tree	0.920 (Normal), 0.957 (Pathological), 0.903 (Suspect)

Table 5: Area under the ROC curve (AUC) for each classifier

3.3 Conclusion

Based on Table 5, we can see that the k-NN classifier has the highest AUC for all 3 classes and hence it is the best classifier among the 3. I am satisfied with the result because the k-NN classifier has AUC of more than 0.9 for all 3 classes and this means that the classifier is able to classify the instances correctly most of the time. Note that the k-NN classifier was ran with the parameters:

1. $k = 3$
2. k fold-cross validation $= 10$
3. 1/d weighting

4 Most discriminating features

To find the most discriminating fetaures, I applied the **AttributeSelection** filter with **InfoGainAttributeEval** as the evaluator and **Ranker** as the search method in the preprocess tab. This filter sorts the features by their information gain compared to the class and the result is as follows:

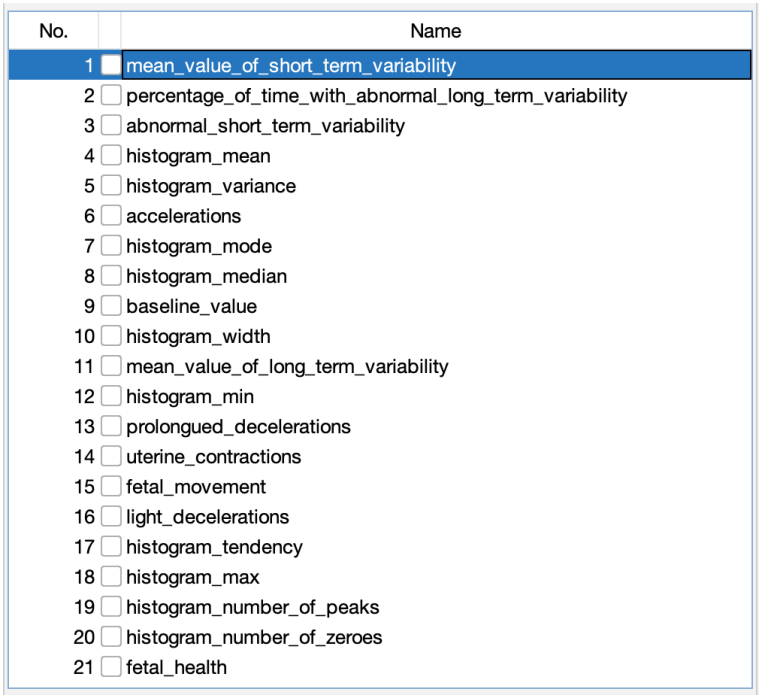

Figure 11: Features sorted by information gain

From the figure above, we can see that the top 5 most discriminating features are:

1. **mean_value_of_short_term_variability**
2. **percentage_of_time_with_abnormal_long_term_variability**

3. **abnormal_short_term_variability**
4. **histogram_mean**
5. **histogram_variance**

5 Finding the top 5 features

5.1 Wrapper + Forward Search

To achieve this I used the **WrapperSubsetEval** attribute evaluator with J48 as the classifier and the search method **GreedyStepwise** with `searchBackwards = false`. The result is shown below.

```
=== Attribute Selection on all input data ===

Search Method:
        Greedy Stepwise (forwards).
        Start set: no attributes
        Merit of best subset found:    0.936

Attribute Subset Evaluator (supervised, Class (nominal): 21 fetal_health):
        Wrapper Subset Evaluator
        Learning scheme: weka.classifiers.trees.J48
        Scheme options: -C 0.25 -M 2
        Subset evaluation: classification accuracy
        Number of folds for accuracy estimation: 5

Selected attributes: 2,4,7,8,9,11,14,17 : 8
                     accelerations
                     uterine_contractions
                     abnormal_short_term_variability
                     mean_value_of_short_term_variability
                     percentage_of_time_with_abnormal_long_term_variability
                     histogram_width
                     histogram_number_of_peaks
                     histogram_mean
```

Figure 12: Wrapper + Forward Search feature selection

5.2 Wrapper + Backward Search

To achieve this I used the **WrapperSubsetEval** attribute evaluator with J48 as the classifier and the search method **GreedyStepwise** with `searchBackwards = true`. The result is shown below.

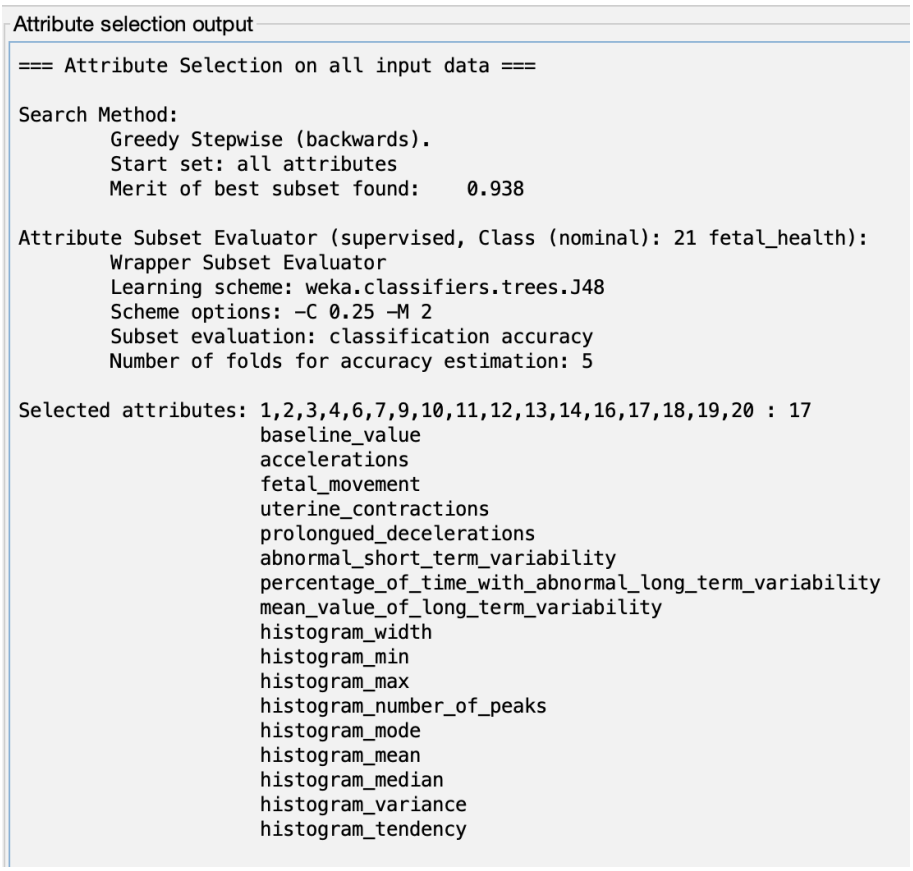

Figure 13: Wrapper + Backward Search feature selection

5.3 Information Gain

To achieve this I used the **InfoGainAttributeEval** attribute evaluator with the search method **Ranker**. The result is shown below.

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 21 fetal_health):
        Information Gain Ranking Filter

Ranked attributes:
 0.3026    8 mean_value_of_short_term_variability
 0.274     9 percentage_of_time_with_abnormal_long_term_variability
 0.2632    7 abnormal_short_term_variability
 0.2184   17 histogram_mean
 0.2122   19 histogram_variance
 0.1991    2 accelerations
 0.1882   16 histogram_mode
 0.1834   18 histogram_median
 0.1453    1 baseline_value
 0.1375   11 histogram_width
 0.1368   10 mean_value_of_long_term_variability
 0.128    12 histogram_min
 0.1261    6 prolongued_decelerations
 0.0859    4 uterine_contractions
 0.0696    3 fetal_movement
 0.0533    5 light_decelerations
 0.0309   20 histogram_tendency
 0.0294   13 histogram_max
 0.023    14 histogram_number_of_peaks
 0        15 histogram_number_of_zeroes

Selected attributes: 8,9,7,17,19,2,16,18,1,11,10,12,6,4,3,5,20,13,14,15 : 20
```

Figure 14: Information Gain feature selection

5.4 Discussion

By comparing the number of selected attributes from the three methods above, we can already see significant differences. *Wrapper + Forward Search* selected only 8 attributes, *Wrapper + Backward Search* selected 17 attributes and *Information Gain* selected all 20 attributes. The information gain method merely ranks the attributes based on their information gain and does not select a subset of attributes hence resulting in all attributes being selected whereas the wrapper method uses a classifier to evaluate the attributes and select only the subset of features which has the most impact on the classifier's performance. Since forward search starts with an empty set and only adds attributes iteratively, it made sense that it selected lesser attributes than backward search which starts with all attributes and removes attributes iteratively.

6 Classifier performance with the top 5 features

6.1 Wrapper + Forward Search

The results was obtained by running the classifiers with the following 5 features:

1. **accelerations**
2. **uterine_contractions**

3. **abnormal_short_term_variability**
4. **mean_value_of_short_term_variability**
5. **percentage_of_time_with_abnormal_long_term_variability**

F1-Measure	Naive Bayes	k-NN	Decision Tree
Normal	0.889	0.941	0.940
Pathologic	0.105	0.766	0.729
Suspect	0.602	0.727	0.759
Weighted Average	0.785	0.897	0.897

Table 6: F1-Measure of each class using the top 5 features from Wrapper + Forward Search

The average of the 3 weighted average F1-Measure is 0.860.

6.2 Wrapper + Backward Search

The results was obtained by running the classifiers with the following 5 features:

1. **baseline_value**
2. **accelerations**
3. **fetal_movement**
4. **uterine_contractions**
5. **prolongued_decelerations**

F1-Measure	Naive Bayes	k-NN	Decision Tree
Normal	0.901	0.928	0.925
Pathologic	0.579	0.728	0.706
Suspect	0.572	0.584	0.563
Weighted Average	0.829	0.864	0.857

Table 7: F1-Measure of each class using the top 5 features from Wrapper + Backward Search

The average of the 3 weighted average F1-Measure is 0.850.

6.3 Information Gain

The results was obtained by running the classifiers with the following 5 features:

1. **mean_value_of_short_term_variability**
2. **percentage_of_time_with_abnormal_long_term_variability**
3. **abnormal_short_term_variability**

4. **histogram_mean**
5. **histogram_variance**

F1-Measure	Naive Bayes	k-NN	Decision Tree
Normal	0.908	0.953	0.952
Pathologic	0.605	0.852	0.870
Suspect	0.584	0.712	0.768
Weighted Average	0.838	0.912	0.920

Table 8: F1-Measure of each class using the top 5 features from Wrapper + Backward Search

The average of the 3 weighted average F1-Measure is 0.890.

6.4 Discussion

By comparing the results above, we can see that the top 5 features from *Information Gain* performed the best and I think this is due to the fact that information gain uses filters and filters has no feature dependencies and no model bias. Looking at the two wrapper methods, forward search performed slightly better than backward search on average but it performed very poorly in the *Pathologic* class of *Naive Bayes*, I think the reason for this is that one of the dominating feature for this specific class was not included in the top 5 features.

7 Principal Components

7.1 Extracting Principal Components

I used the **PrincipalComponents** filter with `centerData = true` in the preprocess tab to combine existing features into principal components. I then removed the other features from the third ranked to the last ranked so I was left with two features. The result looks like this:

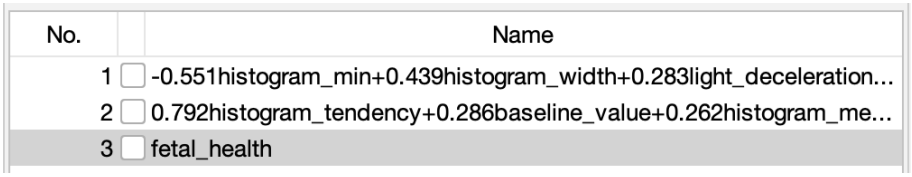

Figure 15: Principal components

7.2 Classifiers performance using the principal components

F1-Measure	Naive Bayes	k-NN	Decision Tree
Normal	0.902	0.907	0.914
Pathologic	0.344	0.680	0.685
Suspect	0.344	0.471	0.452
Weighted Average	0.779	0.828	0.831

Table 9: F1-Measure of each class and weighted average using principal components

The average of the 3 weighted average F1-Measure is 0.813. Comparing this to the results from the previous section, we can see that the performance of the classifiers dropped significantly. This is because although applying principal component analysis (PCA) reduces the dimensionality of the dataset, it also reduces the variance in the dataset. Therefore, the classifiers are not able to perform as well as before.

7.3 Principal Components Visualisation

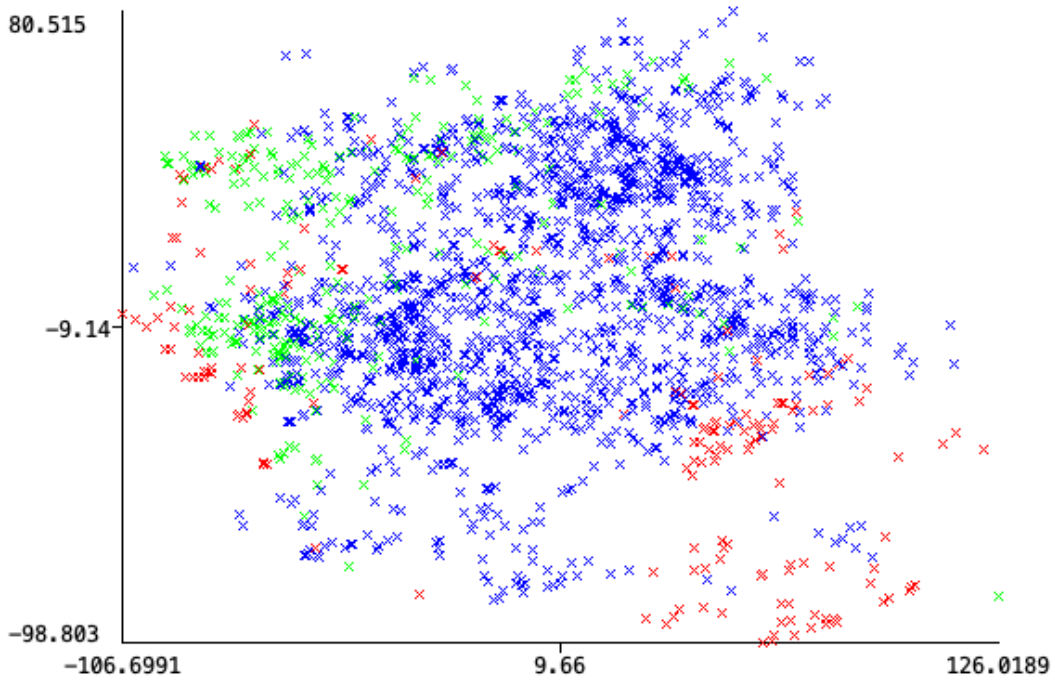

Figure 16: Principal components visualisation

In the visualisation above, **Normal** is represented by blue, **Pathologic** is represented by red and **Suspect** is represented by green.

7.4 Expectation

In my opinion, it is expected that just by doing PCA alone the performance of the classifiers would drop compared to feature selection such as *Information Gain*. This is because even PCA preserves most of the variance in the dataset, it does not take into account which features are more important than others. Therefore, the classifiers would not be able to perform as well as before. Perhaps if we do PCA with more components and followed by feature selection, then the performance of the classifiers would be on par or better.