# Fetal Health Classification

Machine Learning and Artificial Intelligence

Academic year 2020/2021

Filippo Grotto VR460638

September 13, 2021

# Contents

# 1 Motivation and rationale

This project is heavily inspired by a kaggle task [2] about fetal health classification.

> Reduction of child mortality is reflected in several of the United Nations' Sustainable Development Goals and is a key indicator of human progress. The UN expects that by 2030, countries end preventable deaths of newborns and children under 5 years of age, with all countries aiming to reduce under-5 mortality to at least as low as 25 per 1,000 live births.
>
> Parallel to notion of child mortality is of course maternal mortality, which accounts for 295 000 deaths during and following pregnancy and childbirth (as of 2017). The vast majority of these deaths (94%) occurred in low-resource settings, and most could have been prevented.
>
> In light of what was mentioned above, Cardiotocograms (CTGs) are a simple and cost accessible option to assess fetal health, allowing healthcare professionals to take action in order to prevent child and maternal mortality. The equipment itself works by sending ultrasound pulses and reading its response, thus shedding light on fetal heart rate (FHR), fetal movements, uterine contractions and more.

In this context this project is not only an application of machine learning techniques but it also have some meaningful application into real world scenarios thanks to the real data provided.

# 2 State of Art

# 3 Objective (Problem definition and Dataset)

The dataset comes from UCI Machine Learning Repository [3] and it is composed by

> 2126 fetal cardiotocograms (CTGs) automatically processed with the respective diagnostic features measured. The CTGs are were classified by three expert obstetricians and a consensus classification label assigned to each of them. Classification was both with respect to a morphologic pattern (A, B, C. ...) and to a fetal state (N=normal; S=suspect; P=pathologic). Therefore the dataset can be used either for 10-class or 3-class experiments.

We will address the 3-class classification problem so we will try to classify the data according to normal, suspect or pathologic.

# 4 Methodology

The classification problem is addressed using the following techniques:

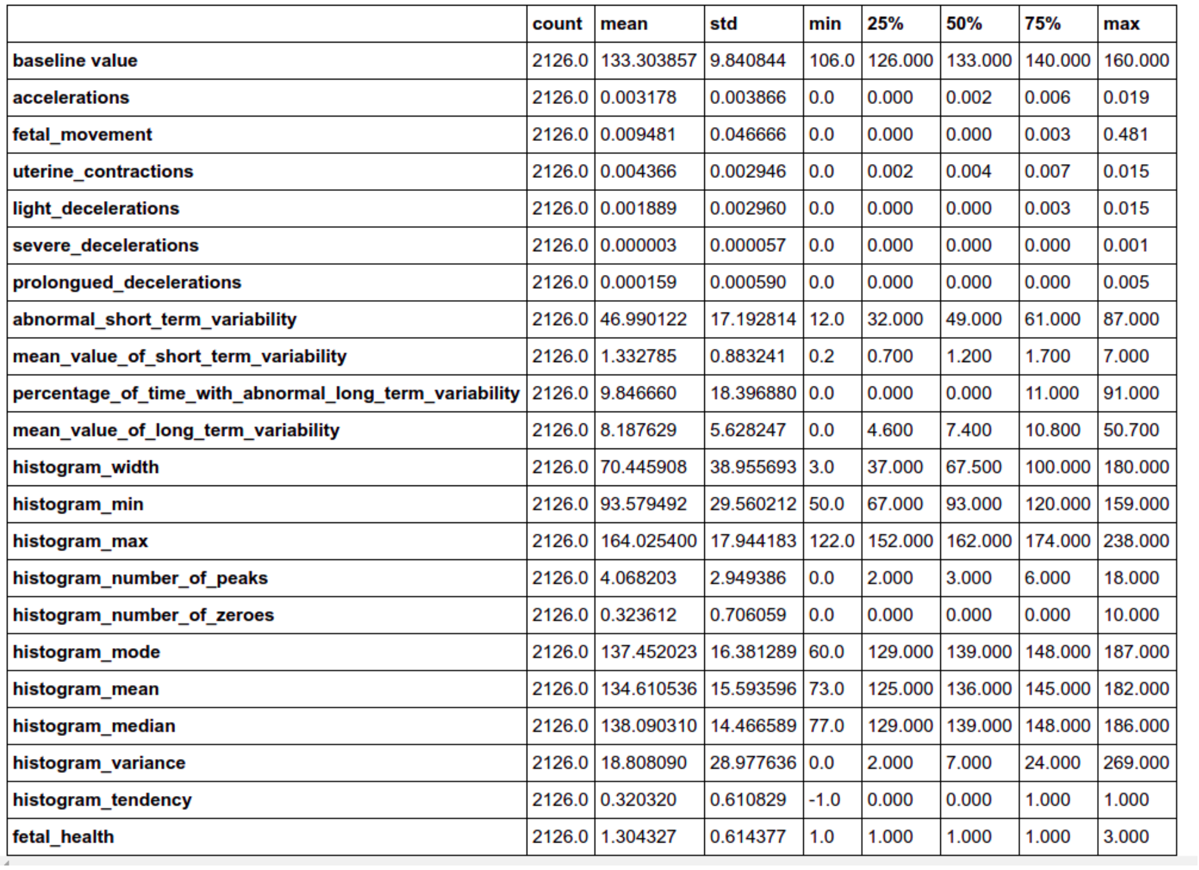
- Naive Bayes [4]
- SVM (with different kernels linear and RBF ) [4]
- KNN (with different k based on preliminary analysis) [4]
- ANN [5]

Moreover, PCA and Fisher (LDA) dimensionality reduction techniques are considered as well as the effects of data scaling and/or normalization. Finally some exploration tasks are reported such as the use of a subset of features according to statistical correlation and a brief analysis of the semi-supervised problem.

# 5 Experiments and Results

## 5.1 Dataset exploration

The dataset is composed by 2126 entries with 22 features (21 features + fetal health which is our target for the classification). From a first analysis there are no null or empty values but there are 13 duplications which we dropped. A brief description of the dataset is reported in Fig 1.



| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| baseline value | 2126.0 | 133.303857 | 9.840844 | 106.0 | 126.000 | 133.000 | 140.000 | 160.000 |
| accelerations | 2126.0 | 0.003178 | 0.003866 | 0.0 | 0.000 | 0.002 | 0.006 | 0.019 |
| fetal_movement | 2126.0 | 0.009481 | 0.046666 | 0.0 | 0.000 | 0.000 | 0.003 | 0.481 |
| uterine_contractions | 2126.0 | 0.004366 | 0.002946 | 0.0 | 0.002 | 0.004 | 0.007 | 0.015 |
| light_decelerations | 2126.0 | 0.001889 | 0.002960 | 0.0 | 0.000 | 0.000 | 0.003 | 0.015 |
| severe_decelerations | 2126.0 | 0.000003 | 0.000057 | 0.0 | 0.000 | 0.000 | 0.000 | 0.001 |
| prolongued_decelerations | 2126.0 | 0.000159 | 0.000590 | 0.0 | 0.000 | 0.000 | 0.000 | 0.005 |
| abnormal_short_term_variability | 2126.0 | 46.990122 | 17.192814 | 12.0 | 32.000 | 49.000 | 61.000 | 87.000 |
| mean_value_of_short_term_variability | 2126.0 | 1.332785 | 0.883241 | 0.2 | 0.700 | 1.200 | 1.700 | 7.000 |
| percentage_of_time_with_abnormal_long_term_variability | 2126.0 | 9.846660 | 18.396880 | 0.0 | 0.000 | 0.000 | 11.000 | 91.000 |
| mean_value_of_long_term_variability | 2126.0 | 8.187629 | 5.628247 | 0.0 | 4.600 | 7.400 | 10.800 | 50.700 |
| histogram_width | 2126.0 | 70.445908 | 38.955693 | 3.0 | 37.000 | 67.500 | 100.000 | 180.000 |
| histogram_min | 2126.0 | 93.579492 | 29.560212 | 50.0 | 67.000 | 93.000 | 120.000 | 159.000 |
| histogram_max | 2126.0 | 164.025400 | 17.944183 | 122.0 | 152.000 | 162.000 | 174.000 | 238.000 |
| histogram_number_of_peaks | 2126.0 | 4.068203 | 2.949386 | 0.0 | 2.000 | 3.000 | 6.000 | 18.000 |
| histogram_number_of_zeroes | 2126.0 | 0.323612 | 0.706059 | 0.0 | 0.000 | 0.000 | 0.000 | 10.000 |
| histogram_mode | 2126.0 | 137.452023 | 16.381289 | 60.0 | 129.000 | 139.000 | 148.000 | 187.000 |
| histogram_mean | 2126.0 | 134.610536 | 15.593596 | 73.0 | 125.000 | 136.000 | 145.000 | 182.000 |
| histogram_median | 2126.0 | 138.090310 | 14.466589 | 77.0 | 129.000 | 139.000 | 148.000 | 186.000 |
| histogram_variance | 2126.0 | 18.808090 | 28.977636 | 0.0 | 2.000 | 7.000 | 24.000 | 269.000 |
| histogram_tendency | 2126.0 | 0.320320 | 0.610829 | -1.0 | 0.000 | 0.000 | 1.000 | 1.000 |
| fetal_health | 2126.0 | 1.304327 | 0.614377 | 1.0 | 1.000 | 1.000 | 1.000 | 3.000 |

**Figure 1:** Description of all feature dataset

Another observation to make is the presence of high class imbalance that comes from the real dataset. In Fig 2 it is visible that **Normal** represents around the 78% of the entire dataset.

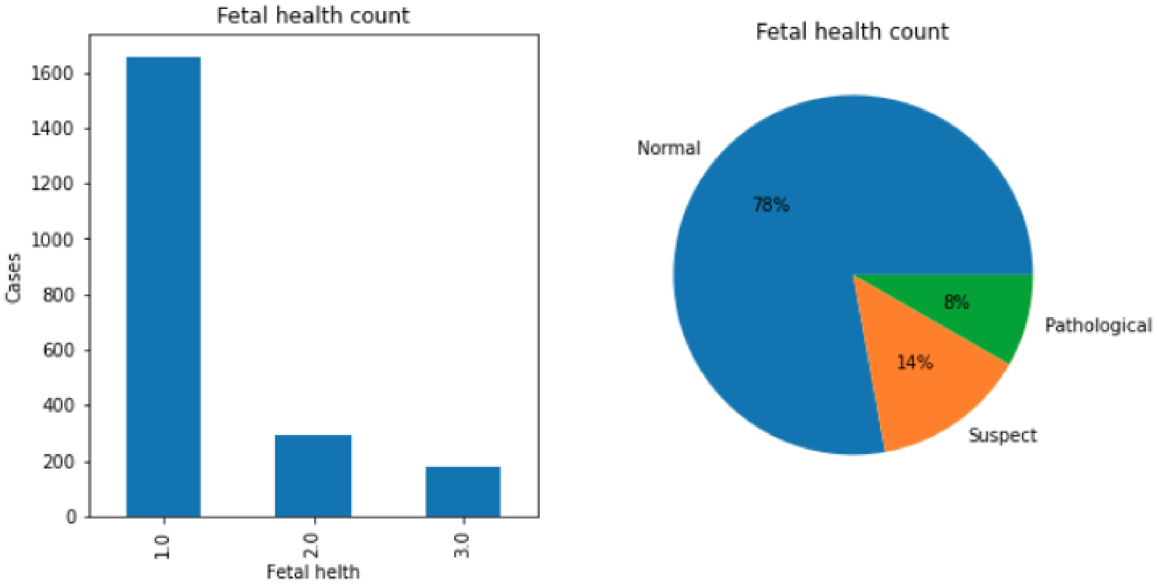


**Figure 2:** Class imbalance in the dataset, around 78% is classified as normal fetal health

Having verified the presence of class imbalance we will use different performance measures for our models in order to avoid misleading considerations:

- **Precision**
- **Accuracy**
- **Recall**
- **F1 Score**
- **Confusion Matrix**

Moreover in order to deal with the class imbalance several options are possible:

- Oversampling the dataset
- Undersampling the dataset
- Use different weights in the classification

In this context **oversampling** was evaluated as the best approach for this project since the dataset for the minority class is pretty limited and classifiers with different weights are not extremely performant (the **Suspicious** class had no precision at all) and it's difficult to estimate correct values of the weights.

In Fig 3 an histogram for each feature is provided. We could clearly see that at least 8 features are extremely skewed and contain a significant amount of outliers. In order to assure a flawless performance of the classifiers models we will scale every feature by standardization.

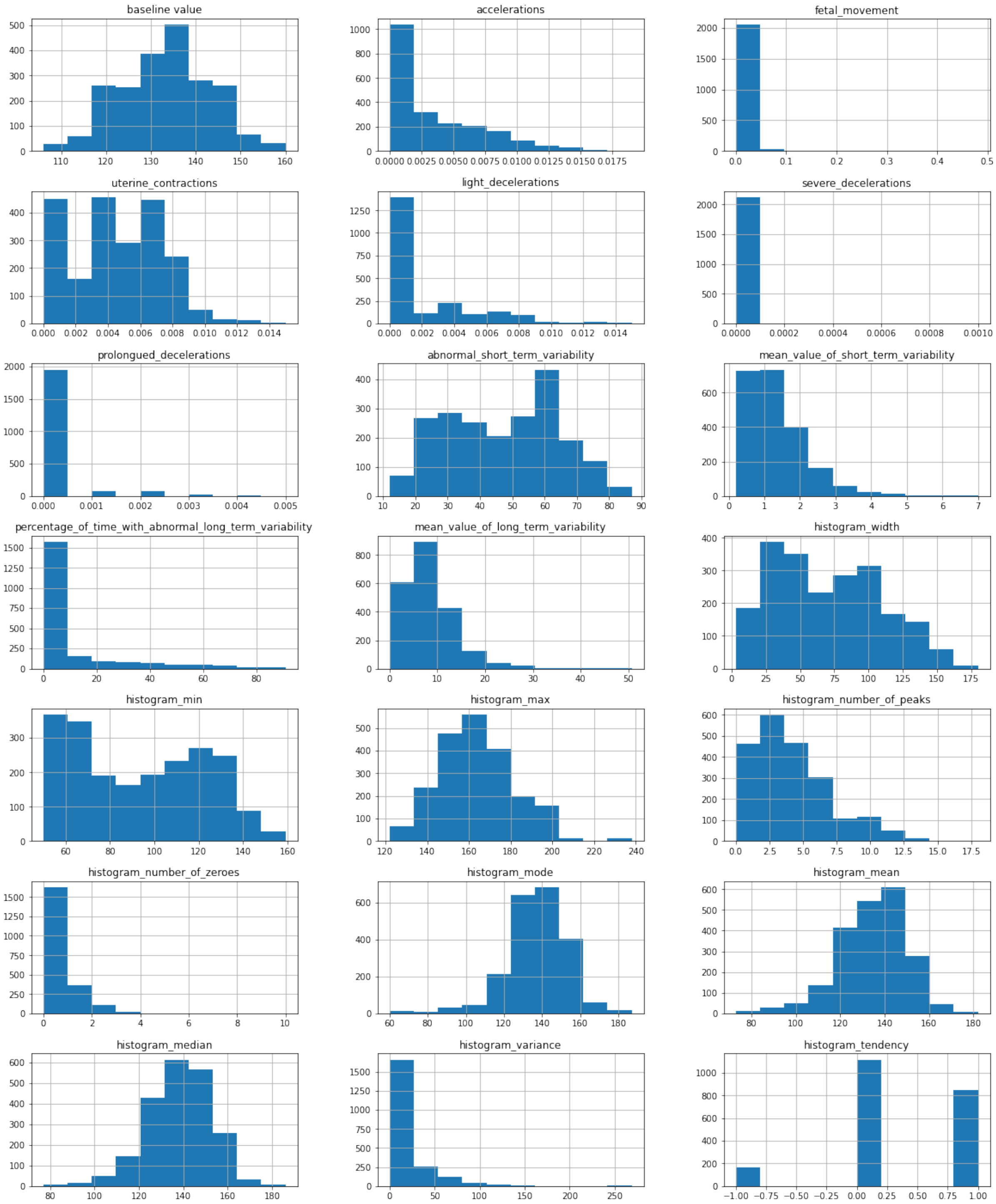


**Figure 3:** Histogram of each feature

Finally we can analyse the correlation between features of the dataset in order to evaluate a possible subset to improve our performances. In table 1 the features with more than 30% correlation with **fetal health** are reported. In Fig 4 a heatmap is reported with the correlation between our features.

| Feature Name | Correlation |
|---|---|
| accelerations | -0.364066 |
| prolongued decelerations | 0.484859 |
| abnormal short term variability | 0.471191 |
| percentage of time with abnormal long term variability | 0.426146 |

**Table 1:** Feature with correlation higher than 30%

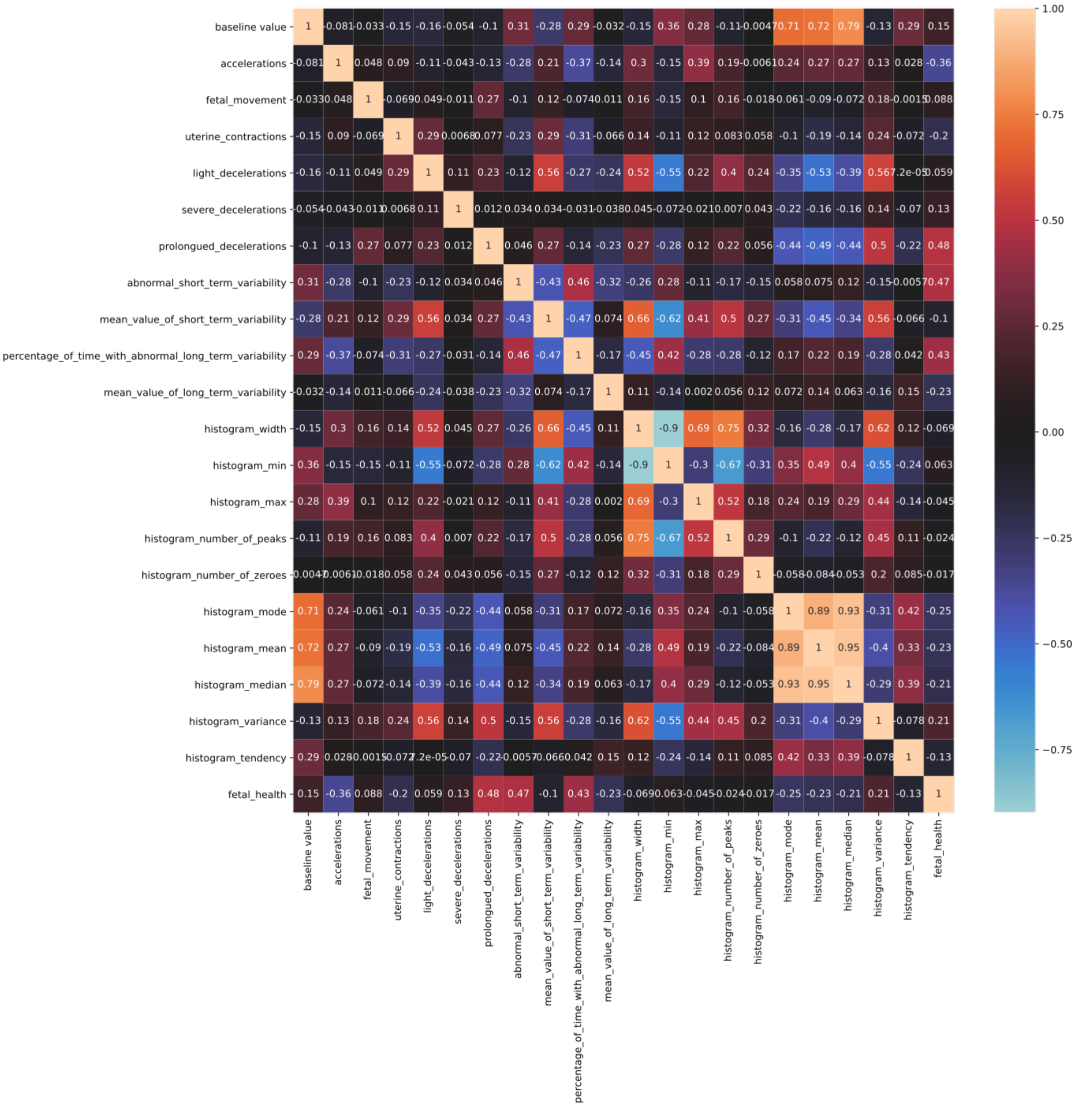


**Figure 4:** Correlation between features

As a result the features with high correlation are **accelerations**, **prolongued decelerations**,**abnormal short term variability** and **percentage of time with abnormal long term variability**. They will be considered in a further analysis as a reasonable subset of the dataset to exploit our models.

## 5.2 Data scaling

In Fig 5 the original data is reported. It's visible that the data requires a proper scaling to avoid the presence of dominant features that might affect our analysis. In Fig 6 the scaled data is reported where **StandardScaler** was used. Data normalization is not actually reported since from experiments it doesn't provide much more benefits and it's irrelevant for the sake of this report.

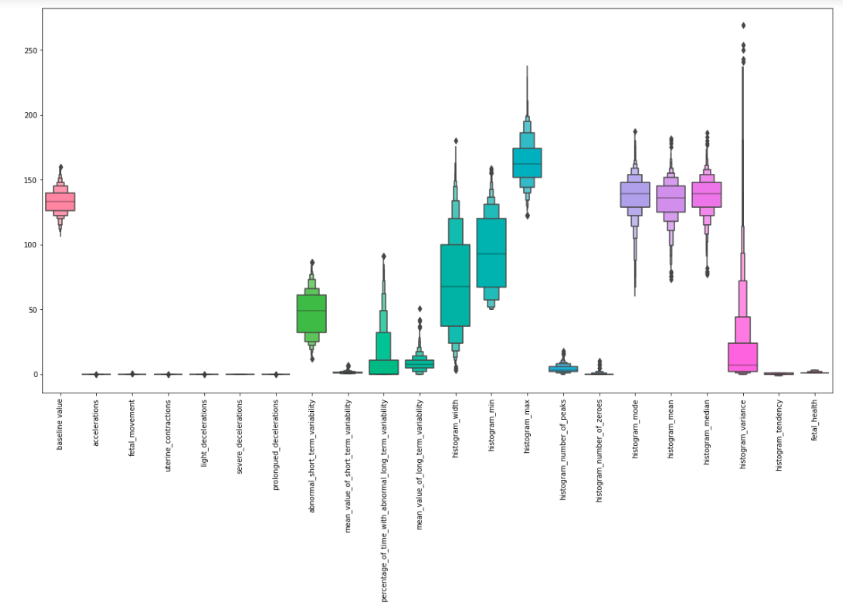


**Figure 5:** Plot of the original data without scaling

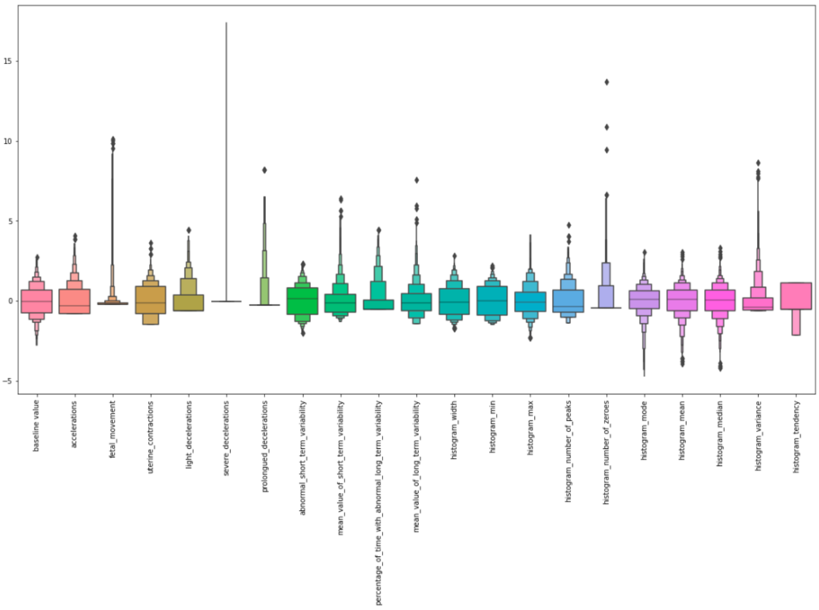


**Figure 6:** Plot of the scaled data

## 5.3 PCA dimensionality reduction

In Fig 7 the analysis of PCA dimensionality reduction is considered. In particular the graph rappresents the cumulative explained variation ratio with respect of all the possible components we might want to consider.

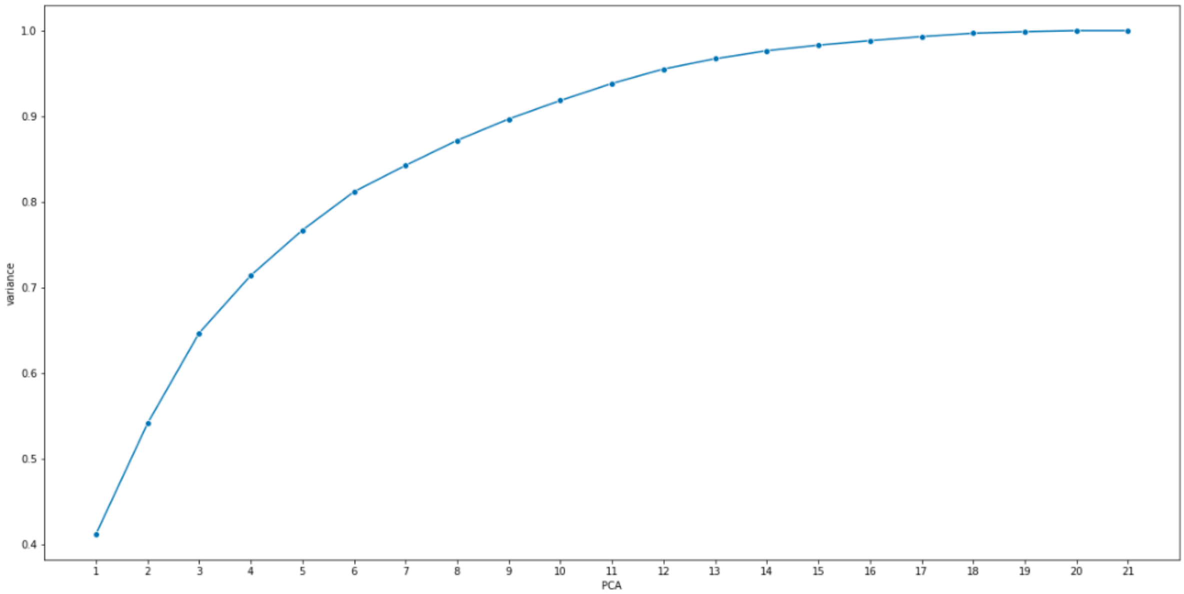


**Figure 7:** PCA cumulative explained variance ratio wrt of components

In particular the last five components don't provide much more information (close to zero) and with 6 components we have a representation of around 80% of our dataset. In table 2 the results obtained by our models over the 6 components is briefly reported to get an idea of the performances.

| Model | accuracy | precision | recall | f1 score |
|---|---|---|---|---|
| SVM linear | 0.818792 | 0.823341 | 0.818792 | 0.820200 |
| SVM poly | 0.847651 | 0.856942 | 0.847651 | 0.849789 |
| SVM rbf | 0.851678 | 0.863157 | 0.851678 | 0.853885 |
| SVM sigmoid | 0.722819 | 0.725679 | 0.722819 | 0.723402 |
| Gaussian NB | 0.775839 | 0.797311 | 0.775839 | 0.778650 |

**Table 2:** Evaluation of our models over 6 PCA components

## 5.4 LDA dimensionality reduction

In this section a brief analysis of Linear Discriminant Analysis is reported in Fig 8 where we can easily see that the cumulative variance ratio accumulated with 1 component rappresents more than 80% of our dataset. Moreover, both the case with 1 and 2 components are briefly reported in table 3 and 4 to get an idea of the performances.

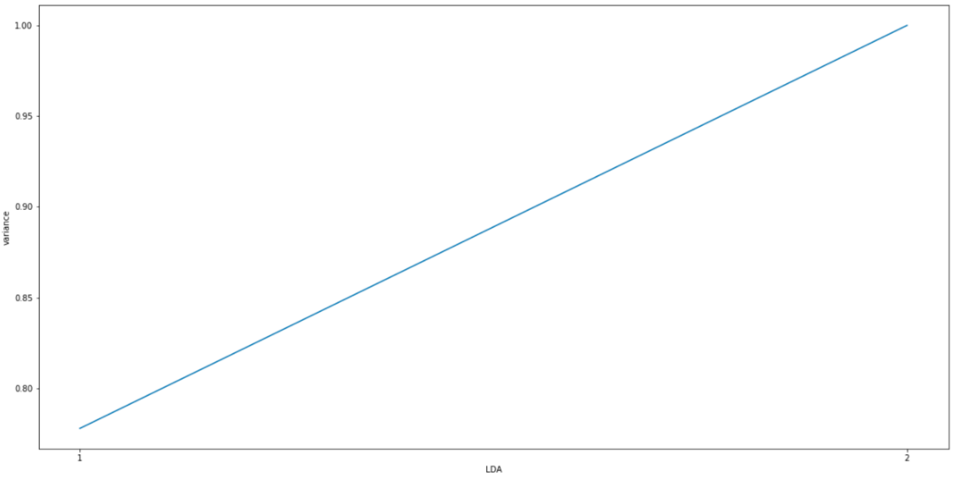


**Figure 8:** LDA cumulative explained variance ratio wrt of components

| Model | accuracy | precision | recall | f1 score |
|---|---|---|---|---|
| SVM linear | 0.773154 | 0.775784 | 0.773154 | 0.774252 |
| SVM poly | 0.755034 | 0.791533 | 0.755034 | 0.761348 |
| SVM rbf | 0.779195 | 0.787309 | 0.779195 | 0.781934 |
| SVM sigmoid | 0.698658 | 0.714204 | 0.698658 | 0.703990 |
| Gaussian NB | 0.768456 | 0.768600 | 0.768456 | 0.768378 |

**Table 3:** Evaluation of our models with 1 LDA component

| Model | accuracy | precision | recall | f1 score |
|---|---|---|---|---|
| SVM linear | 0.846309 | 0.855568 | 0.846309 | 0.848366 |
| SVM poly | 0.840940 | 0.876105 | 0.840940 | 0.844297 |
| SVM rbf | 0.860403 | 0.876728 | 0.860403 | 0.862739 |
| SVM sigmoid | 0.755034 | 0.754202 | 0.755034 | 0.754471 |
| Gaussian NB | 0.835570 | 0.850102 | 0.835570 | 0.838258 |

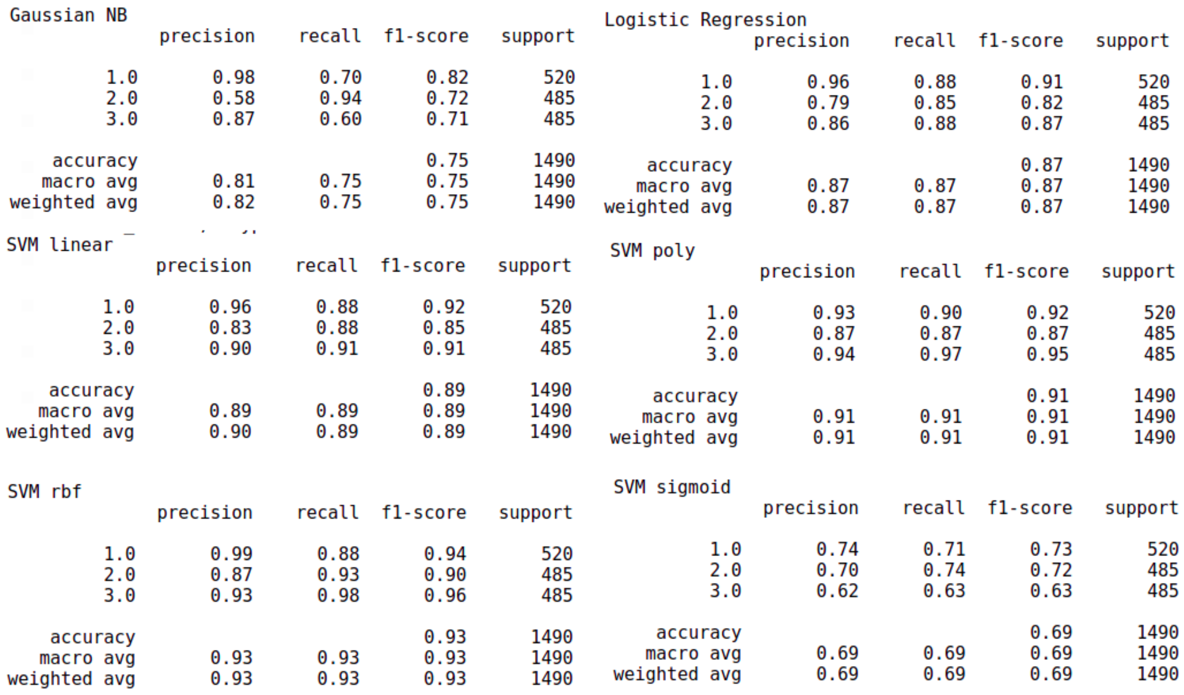
**Table 4:** Evaluation of our models with 2 LDA components

In general in case of uniformly distributed data, LDA almost always performs better than PCA. However if the data is highly skewed (irregularly distributed), like in our scenario, then it is advised to use PCA since LDA can be biased towards the majority class. [6]

## 5.5 Model evaluations with the full dataset

In this section the evaluations of several models are presented. The entire dataset is oversampled using *SMOTE* and then it is partitioned using *train_test_split* function from sklearn using around 30% of the entire dataset for the testing set. The results of our models are reported in Fig 9. It's visible that SVM with rbf kernel produces the best results over our dataset. This is coherent with the analysis in [4] and what the literature provides us about Support Vector Machines Kernels. All models use as default OneVsRest approach. Let's consider:

- 1.0 is Normal class (520 samples in the testing set)
- 2.0 is Suspicious class (485 samples in the testing set)
- 3.0 is Pathologic class (485 samples in the testing set)



```
Gaussian NB
              precision    recall  f1-score   support

         1.0       0.98      0.70      0.82       520
         2.0       0.58      0.94      0.72       485
         3.0       0.87      0.60      0.71       485

    accuracy                           0.75      1490
   macro avg       0.81      0.75      0.75      1490
weighted avg       0.82      0.75      0.75      1490
```

```
Logistic Regression
              precision    recall  f1-score   support

         1.0       0.96      0.88      0.91       520
         2.0       0.79      0.85      0.82       485
         3.0       0.86      0.88      0.87       485

    accuracy                           0.87      1490
   macro avg       0.87      0.87      0.87      1490
weighted avg       0.87      0.87      0.87      1490
```

```
SVM linear
              precision    recall  f1-score   support

         1.0       0.96      0.88      0.92       520
         2.0       0.83      0.88      0.85       485
         3.0       0.90      0.91      0.91       485

    accuracy                           0.89      1490
   macro avg       0.89      0.89      0.89      1490
weighted avg       0.90      0.89      0.89      1490
```

```
SVM poly
              precision    recall  f1-score   support

         1.0       0.93      0.90      0.92       520
         2.0       0.87      0.87      0.87       485
         3.0       0.94      0.97      0.95       485

    accuracy                           0.91      1490
   macro avg       0.91      0.91      0.91      1490
weighted avg       0.91      0.91      0.91      1490
```

```
SVM rbf
              precision    recall  f1-score   support

         1.0       0.99      0.88      0.94       520
         2.0       0.87      0.93      0.90       485
         3.0       0.93      0.98      0.96       485

    accuracy                           0.93      1490
   macro avg       0.93      0.93      0.93      1490
weighted avg       0.93      0.93      0.93      1490
```

```
SVM sigmoid
              precision    recall  f1-score   support

         1.0       0.74      0.71      0.73       520
         2.0       0.70      0.74      0.72       485
         3.0       0.62      0.63      0.63       485

    accuracy                           0.69      1490
   macro avg       0.69      0.69      0.69      1490
weighted avg       0.69      0.69      0.69      1490
```

**Figure 9:** Results of the selected models over our dataset

From a brief analysis we can clearly see that SVM in general works pretty well over our dataset, Naive Bayes and SVM with sigmoid kernel are pretty bad in terms of precision for **suspicious** and **pathologic**. In general the class **Normal** has better performances. Moreover in Fig 10 and Fig 11 the confusion matrixes for the svm with rbf kernel and Naive Bayes are respectively reported. What is clearly visible is the fact that for SVM with rbf there are few false positive and false negative values which is a pretty good result. For Naive Bayes we have a false negative value which is even higher than the true value which is pretty bad for our classification problem. In fact, in the real context, we can't accept to misclassify a suspicious element with a normal one but we can easily accept the opposite (false positive).

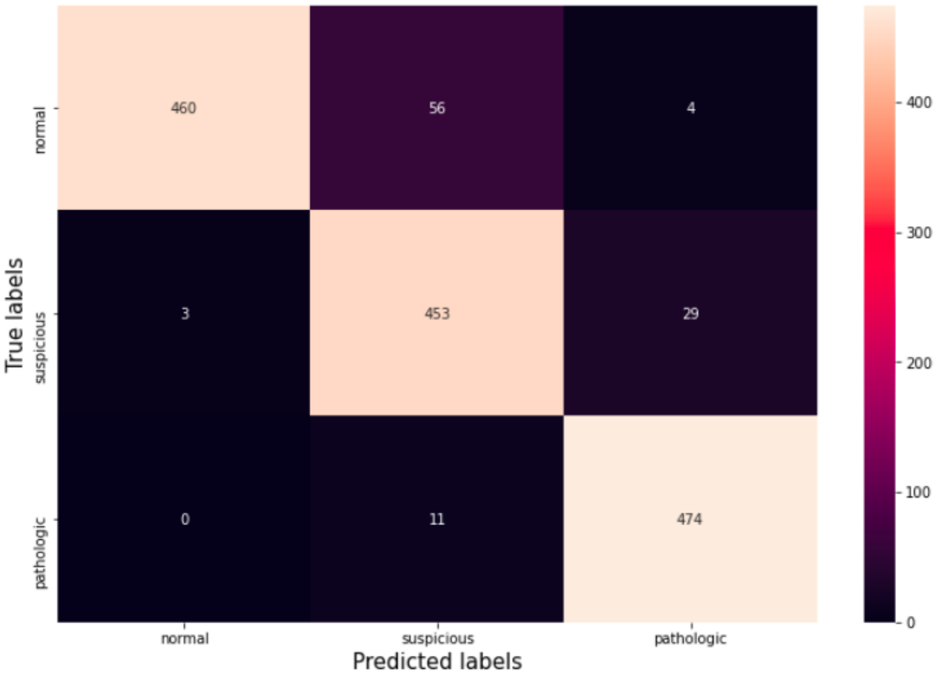


**Figure 10:** Confusion matrix of SVM with rbf kernel

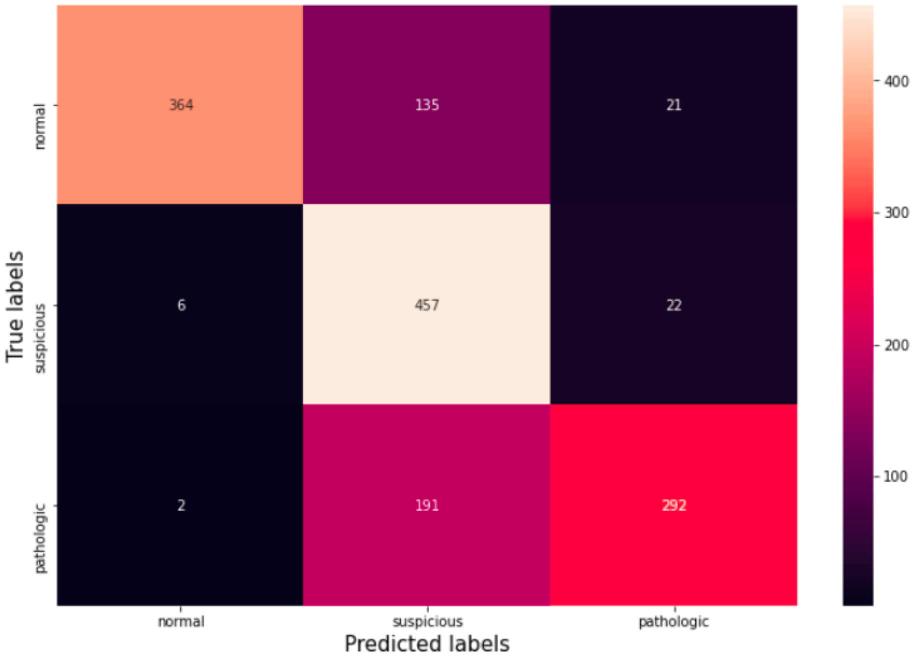


**Figure 11:** Confusion matrix of Naive Bayes approach

### 5.5.1 K-Nearest Neighbors

In this section we will consider the use of K-NN (with an euclidean distance applied) to our dataset. In Fig 12 a brief analysis of several k is provided with the related metrics to get an idea of which are the most promising values of K to use.

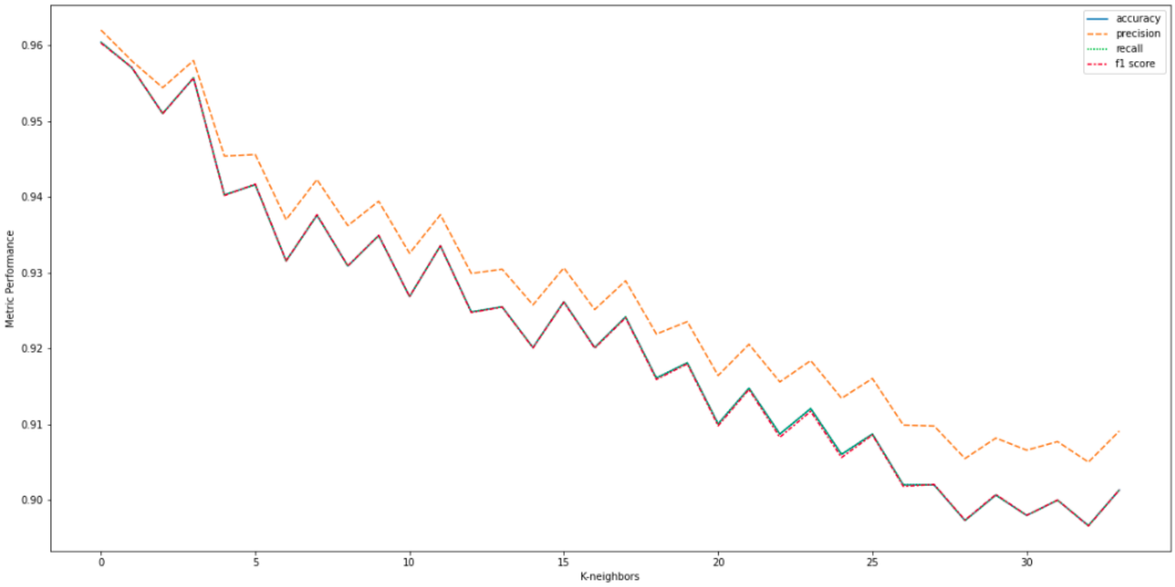
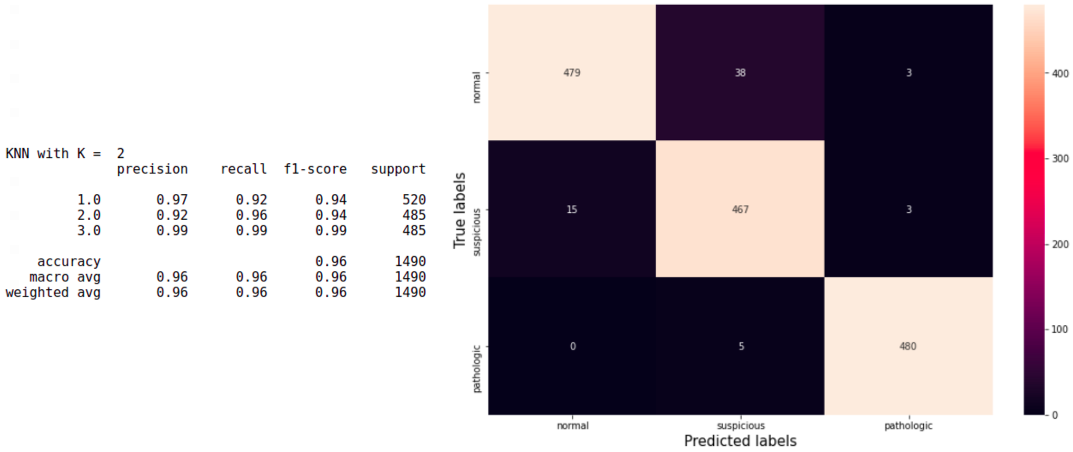


**Figure 12:** K-NN metrics with respect of the value K-neighbors

As a result the values K = 2 and K = 4 are the best choices. In Fig 13 the best case is considered and analysed.



```
KNN with K =  2
              precision    recall  f1-score   support

         1.0       0.97      0.92      0.94       520
         2.0       0.92      0.96      0.94       485
         3.0       0.99      0.99      0.99       485

    accuracy                           0.96      1490
   macro avg       0.96      0.96      0.96      1490
weighted avg       0.96      0.96      0.96      1490
```

**Figure 13:** K-NN with k=2 metrics and confusion matrix

As a conclusion K-NN with k=2 is a good model with few false positive. However the dataset is pretty limited so K-NN is suitable in this case but it might be extremely data intensive for large datasets.

### 5.5.2 Artificial Neural Networks MLP

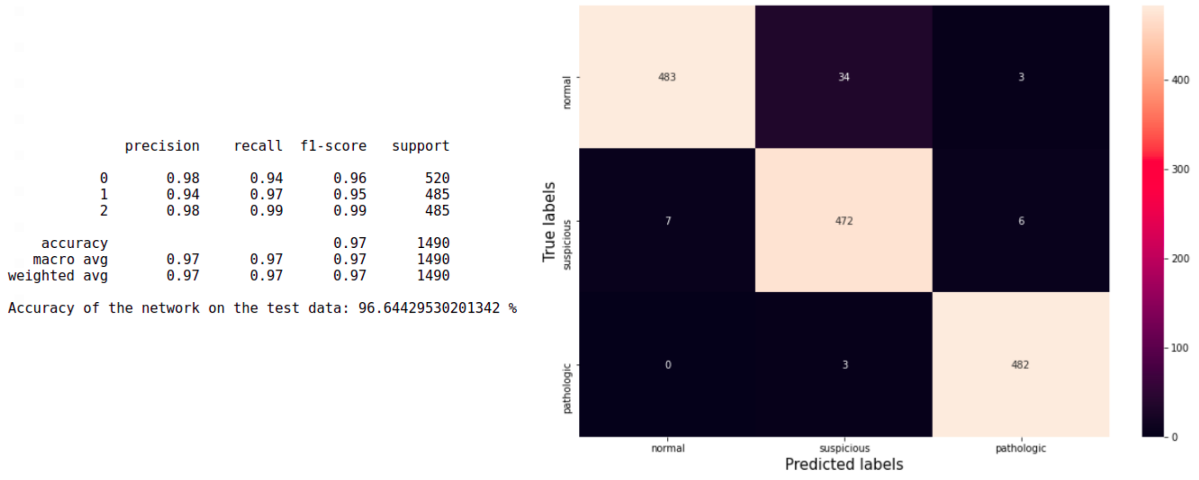
In this section we will brief provide some results obtained by using Artificial Neural Networks to solve the classification problem. In particular we created a multilayer perceptron (MLP) with the following parameters

```
input_size = 21
hidden_size = 100
num_classes = 3
num_epochs = 100
batch_size = 50
learning_rate = 0.001

# Fully connected neural network with one hidden layer
class NeuralNet(nn.Module):
  def __init__(self, input_size, hidden_size, num_classes):
      super(NeuralNet, self).__init__()
      self.fc1 = nn.Linear(input_size, hidden_size)
      self.relu = nn.ReLU()
      self.fc2 = nn.Linear(hidden_size, num_classes)

  def forward(self, x):
      out = self.fc1(x)
      out = self.relu(out)
      out = self.fc2(out)
      return out
```

The MLP uses one fully connected layer, a ReLU as activation function and another fully connected layer (Linear). The criterion used is CrossEntropyLoss. This model have an accuracy of around 96% which makes it a good model for the classification task as reported in Fig 14. In Fig 15 the train accuracies and losses are reported.



```
              precision    recall  f1-score   support

           0       0.98      0.94      0.96       520
           1       0.94      0.97      0.95       485
           2       0.98      0.99      0.99       485

    accuracy                           0.97      1490
   macro avg       0.97      0.97      0.97      1490
weighted avg       0.97      0.97      0.97      1490

Accuracy of the network on the test data: 96.64429530201342 %
```

**Figure 14:** Results of the MLP network

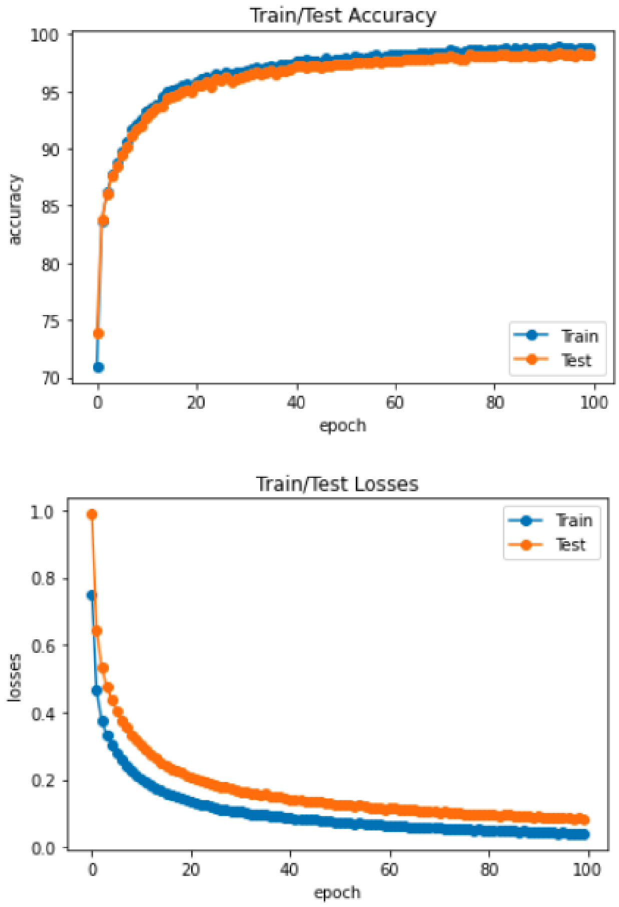


**Figure 15:** Accuracy and losses of the MLP

## 5.6 Other Exploratory Tasks

### 5.6.1 Models evaluation using a subset of features

In this section an evaluation of our models over a subset of features defined in the previous section is presented. Only features with more than 30% correlation with the target **fetal health** are used. They are reported in Table 1. In Table 5 the evaluations of our models are reported according to the defined metrics. Moreover, in Fig 16 the classification report for the best model SVM rbf is reported.

| Model | accuracy | precision | recall | f1 score |
|---|---|---|---|---|
| SVM linear | 0.76 | 0.76 | 0.76 | 0.76 |
| SVM poly | 0.82 | 0.84 | 0.82 | 0.82 |
| SVM rbf | 0.85 | 0.85 | 0.85 | 0.85 |
| SVM sigmoid | 0.56 | 0.57 | 0.55 | 0.56 |
| Gaussian Naive | 0.73 | 0.77 | 0.73 | 0.73 |

**Table 5:** Evaluation of our models over a subset of features

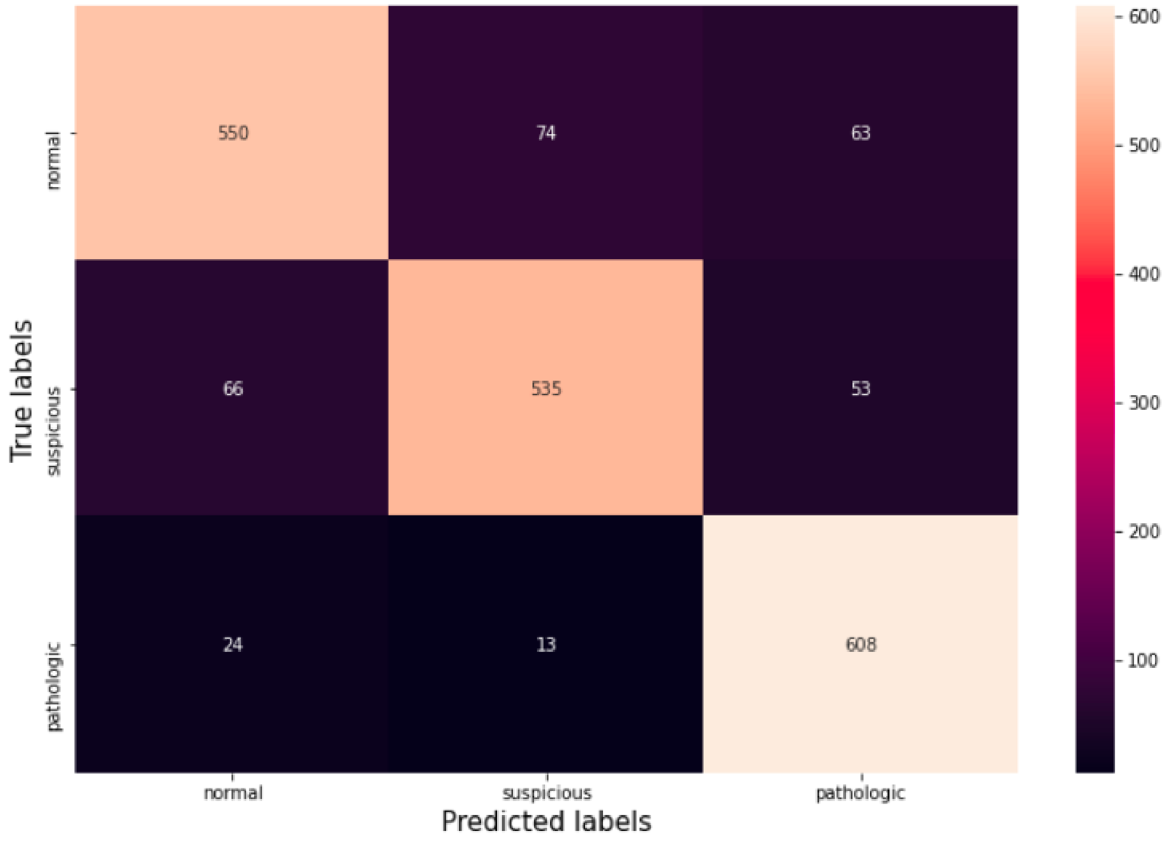


**Figure 16:** Detailed classification report for SVM with rbf kernel

Not all the confusion matrixes are reported but both Gaussian Naive and SVM with sigmoid have severe problems in distinguish the two classes **Suspicious** and **Normal** since they report a lot of false positives and negatives. Anyway we showed how, using a lower number of features, we are still able to get pretty good results with a SVM with rbf and have a decent confusion matrix (low number of false positive an negative) even though it's not as good as using all the features. This is a reasonable result given the fact that the correlation was pretty low around 30%

### 5.6.2 Semisupervised Problem

In this section the semisupervised problem is briefly approached using a simple pseudo labelling method using the SVM rbf which proved to be one of the best classifier for this scenario. We use 100 samples per each class to build the labeled training dataset. We train the classifier with that labeled dataset (300,21) and then we predict the pseudo labels from the unlabeled part of the dataset (3175, 21). We combine the two results and we train a classifier using the full training dataset (with the true labels and the pseudo labels). Finally we predict the test dataset. The results for the semisupervised problem are reported in Fig 17, the results for the fully supervised problem are reported in Fig 18

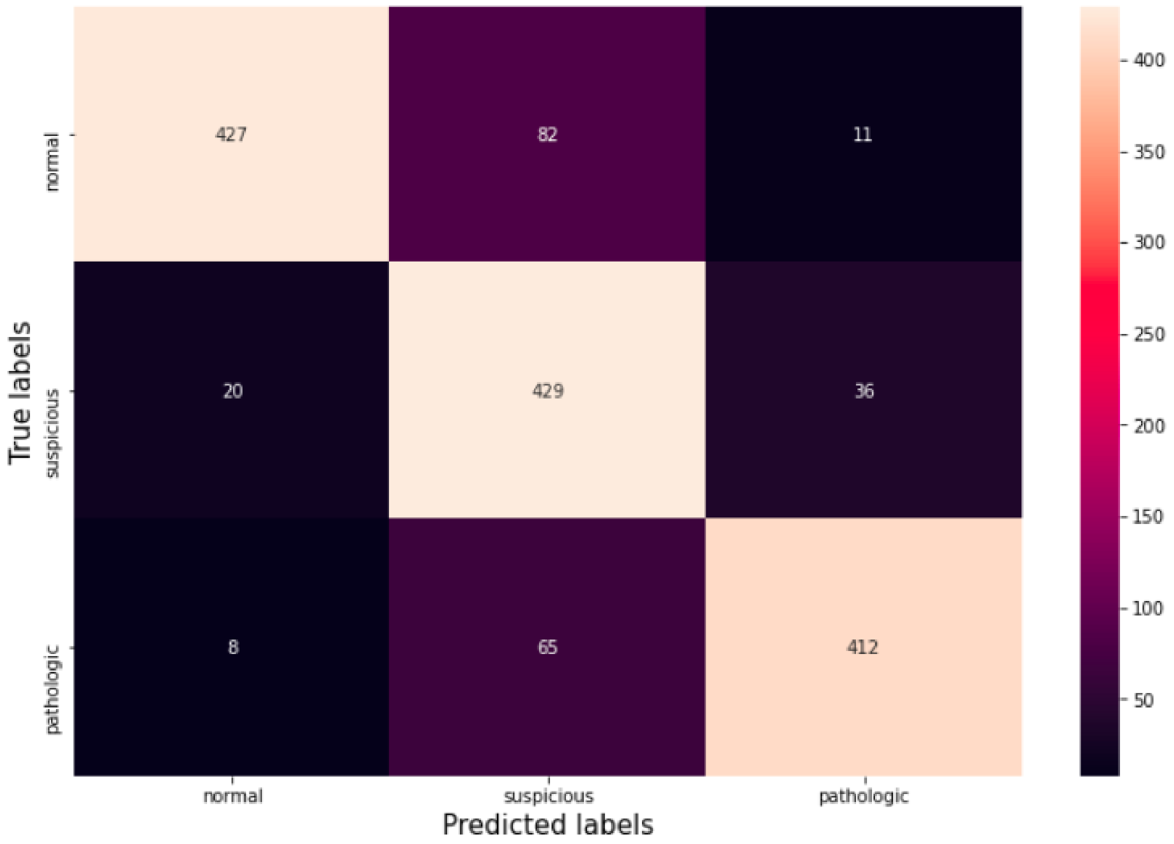


**Figure 17:** Classification report for the semisupervised problem using OneVsOne SVM with rbf kernel

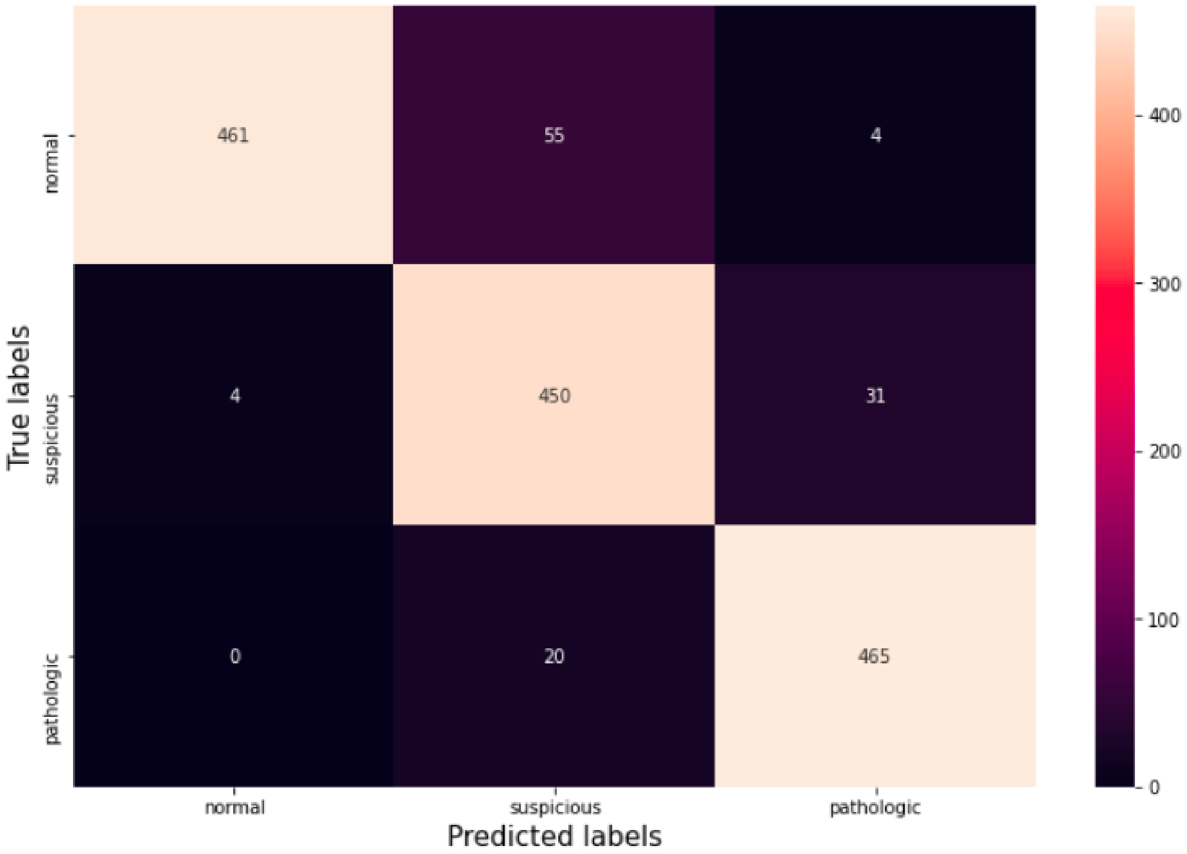


**Figure 18:** Classification report for the fully supervised problem using OneVsOne SVM with rbf kernel

What is clearly visible is the fact that using a semisupervised approach we have less precision with the second class **Suspicious** and we are introducing some false negatives/positives that are visible in the confusion matrix. For real application this is not recommended since false positive should be avoided as much as possible. Several configurations have been tried using more samples per each class or even lower number of samples (100 samples per class got reasonable results)

# 6 Conclusion

In this report the study of fetal health classification has been addressed using a real dataset. Several different models have been analysed using different metrics and different dimensionality reduction methodologies. We showed how SVM with rbf kernel, K-NN with k=2 and MLP produce really reasonable results but they still have some problems in terms of false positives/negatives with the suspicious/normal classification. Finally, it's important to mention that we didn't reach improvements of the state of the art where other more advanced techniques (XGBoost, Random forests, Decision tree models, CNN) are used. [5] [7]

# References

[1] Ayres de Campos et al. (2000) SisPorto 2.0 A Program for Automated Analysis of Cardiotocograms

[2] Fetal Health Classification https://www.kaggle.com/andrewmvd/fetal-health-classification

[3] Cardiocotography dataset https://archive.ics.uci.edu/ml/datasets/cardiotocography

[4] Hoodbhoy, Zahra et al. "Use of Machine Learning Algorithms for Prediction of Fetal Risk using Cardiotocographic Data."

[5] Yılmaz, E. Fetal State Assessment from Cardiotocogram Data Using Artificial Neural Networks.

[6] Aleix M. Martinez and Avinash C. Kak PCA versus LDA

[7] Jianqiang Li , Zhuang-Zhuang Chen, Luxiang Huang, Min Fang, Bing Li, Xianghua Fu, Huihui Wang Automatic Classification of Fetal Heart Rate Based on Convolutional Neural Network