Machine learning prediction modelling*

*Classifying risk factors in medical treatment

Søren Harbo Hermansen
Institute for Electro- and Computer engineering
Aarhus University
Aarhus, Denmark
201509802@post.au.dk

***Abstract*—This paper introduces the classification problem of risk factors in medical treatment of patients. This paper tries to solve this problem by using different machine learning algorithms, that can classify and predict the risk factors. This paper will explore the algorithms and describe their performance, measured in accuracy of prediction on the specific problem. Herein it will compared the algorithms and discuss which of the explored algorithms is the best.**

***Index Terms*—Machine learning, Classification, Nearest centroid, Perceptron, Backpropagation, Neural Network, Principal Component Analysis, Linear Discriminant Analysis**

I. Introduction

To help humans minimize errors, machines have been relied upon more through the time. As machines have become more advanced, so have the use for them. To help people make decision in their fields of specialty, machine learning algorithms can be used. These algorithms can sift through high amounts of data to make a qualitative decision based on the data.

The machine learning algorithms also know as classifiers, are used by training them, meaning that the classifiers are fed some data with a true outcome. The classifier can then tie the data to this outcome. When the classifier is given an input, it can then predict an outcome, based upon which it has been trained.

In the machine learning field, there are a plethora of different classifiers that operate differently, and the different classifiers are highly optimal for different classification problems. This is based on the idea that there is not a single classifier that is optimal for all problems. It is therefore very important to notice that, not all classifiers will perform the same on the data that it has been trained upon. This will then lead to a search for the best classifier, which is the one with the best performance measured in accuracy of predictions.

In this search there are also different ways to optimize the classifiers using the hyper parameters of the classifiers. The data used can also be analyzed to see if there are any irregularities or if anything can be optimized, for the classifiers to become more accurate. The analysis of data is often called pre-processing, and is one of the most important steps when dealing with machine learning, as larger amount of data, will inflict a longer training period.

This paper focuses on a classification problem with a data set that is made up of 450 patients that have been treated with a Total Knee Arthroplasty. For the 450 patients there are 23 columns which consists of information about the patients. The data is split into a training, validation, and test set with a split of 66% training, 11% validation and 22% test.

For the training and validation set a true outcome is also available, denoted as either a zero for success or one for not successful, describing a binary classification problem. Table I shows the distribution of ones and zeros representing the true outcome in the training and validation set. This shows that there are a lot more zeros than ones, which could lead to a harder time of classifying, because of the sparse data.

TABLE I
Class distribution

	Training	*Validation*
0	236	64
1	35	15

II. Methods

This section describes the different methods of machine learning used in identifying and classifying the specific problem.

A. Class imbalance

When there is an imbalance in the data set, balancing can help with creating a better classifier, in that it is easier to classify the differences between the two classes. When there is smaller amounts of data from one class, it can have more outliers which, blurs the true cluster of the class. One way to rectify the imbalance, is to over sample, meaning that, existing data points gets replicated randomly for the given imbalanced class. Another way could be to weight the imbalanced class,

so that it has greater importance in the classifier. This creates a larger importance of data for the imbalanced class, which can lead to a better classifier and better accuracy.

B. Feature selection

When dealing with a data set, data correlation is important to calculate to see if there are data that increases the same through the data. This creates a better overview of what data is important and it helps with narrowing down which types of data that can be removed without having an impact. In larger data sets removing features that, have less importance can make it easier to train in terms of training time. For the data set used in this paper, the correlation between the data can be seen in figure 1.

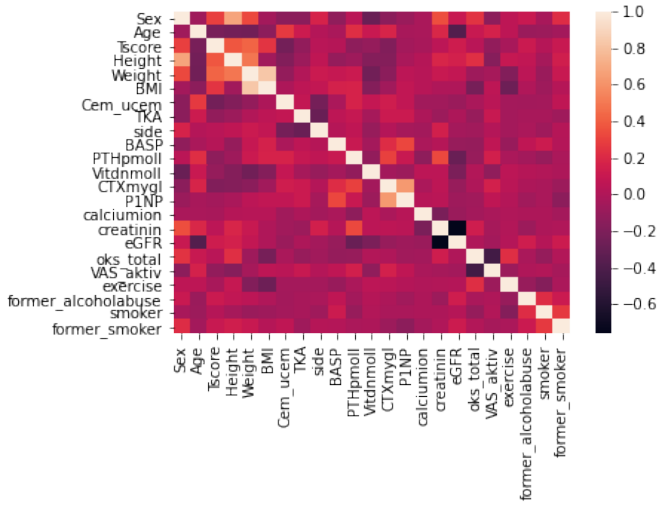

Fig. 1. Correlation between features.

In the figure it can be seen that *height*, *weight* and *BMI* obviously correlates, but also the *CTXmygl* correlates with *P1NP*. Something else is that *eGFR* and *creatinin* does not correlate at all.

C. Missing data

The data set has some few missing points of data, which can be solved in a variety of ways. In this case, it was solved by taking the average of the all the points for the missing feature. As the training set has most data points, this average was used for all the missing data points in the set.

D. Principal Component Analysis (PCA)

As described earlier the data set has 23 columns, which is also called features. Visualizing 23 features is impossible and therefore to visualize the data, Principal Component Analysis can be used. PCA creates Principal components which can be seen as axes of a coordinate system. The Principal components are created according to the variance of the data. The first principal component covers the highest variance in the system, the second component the second most variance.

There are as many components as features in the data. Mostly the first three components cover the most variance in the data. This makes it easier for visualizing the data, as the data can be displayed in a graph made up of the principal components. The dimension reduction created by PCA can also be used for the machine learning classifiers.

E. Linear Discriminant Analysis (LDA)

The classification problem described in this paper is binary which makes Linear Discriminant Analysis a great method to use. LDA creates a line where the data is most linearly separable. The line is created by calculating the mean vectors for the two classes and the variance in the two classes. It will then fit the line where the distance between the two mean vectors is the largest and the variance of the two classes are the smallest. The data can then be projected onto the fitted line and because of this it will be easier to separate the two classes. LDA can be used in conjunction with the different classifiers, creating a better result.

F. Nearest Centroid Classifier (NCC)

The Nearest Centroid Classifier is a distance based algorithm. It is based on the idea that data with specific classes is clustered, which means that data from specific classes are bound to be close to each other. NCC works by calculating a mean vector for the classes present in the data, the mean vector is also called a centroid. For the data set used in this paper it presents two mean vectors. NCC then predicts based on the distance to the centroids. Which ever centroid is nearest the data, is classified to that centroids class. The mean vectors are calculated according to equation 1. Whereas k is the class of the datapoint, and l is the label.

$$\mu_k = \frac{1}{N_k} \sum_{i,l_i=k} x_i, k = 1, \ldots, K \tag{1}$$

Two different distance metrics are used for the NCC, being the euclidean and manhatten distance. Euclidean distance is base on pythagorean theroem $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, and manhatten is based on the sum of vector $d = |(x_2 - x_1)| + |(y_2 - y_1)|$, where d is the distance, (x_1, y_1) and (x_2, y_2) is two points. Centroid is faster when there are low amounts of classes. To extend NCC, the classes can also be subdivided into smaller centroid, using k-means for clustering. This could create a more precise classification.

G. K-Nearest neighbor Classifier (KNN)

The K-Nearest neighbor classifier is also a distance base algorithm, and classifies according to the nearest neighbor of the data that it classifies. It works by measuring the distance from the prediction data to all the points in the test set. It then predicts the class for which the nearest neighbor belongs to. It can then also change the number of neighbors it measures for the classification. The model for KNN is the entire training set opposed to other classifiers.

KNN is a slow algorithm since it measures distance to every data point in the data set, so if the data set is large, the classification is very slow. Changing the number of neighbors can increase the precision of the classifier, but there is also a trade off, of how many to use since it could, accidentally have an overflow of neighbors belonging to the wrong class.

H. Support Vector Machine (SVM)

Support Vector Machine based on the idea of data being linearly separable since it creates a hyperplane which separates the classes in the data set. To increase the separability between the hyperplane and the data margin is used. This margin can be maximized outwards towards the data points that are closest to the hyperplane. The margin will then be maximized when it hits the points in the classes. These points are called support vectors.

I. Perceptron with Backpropagation

The perceptron very much like the support vector machine in that it creates a linear decision function or a hyperplane that separates the classes in the data set. This creates a binary classification.

A linear decision function applied upon vector x is written as:

$$g(x) = \sum_{d=1}^{D} w_d x_d + w_0 \tag{2}$$

Where D is the dimensions in R^D, w is the weight vector and expresses the orientation of the hyperplane. w_0 is the displacement of the hyperplane from the origin. $g(x)$ is the decision function which leads to if $g(x) < 0$ it belongs to one class and if $g(x) > 0$ it belongs to the other, drawing the hyperplane. If the sample is on the line, it cannot be classified, in which a margin can be introduced. The margin makes sure that the decision function is not placed on the samples.

When training with backpropagation it means that when the training data is given, a know label for the output is also known. This means that when measuring the output for the given training data, weights can be applied to rectify the output to be the correct one. For every training data the weights of the system are adjusted to create the optimal weights. The cost function describes the performance of the system and changing the weights should reduce the overall cost function. Although reducing the cost function the most, could also lead to overtraining in that the data that is to be predicted is most likely different from the training data, although thought to be similar.

Backpropagation then works by going backwards through the system to optimize the weights. How much the weights are changed in the system is based on the learning rate described as η. It is important to use different learning rates to optimize the system and the weights. If the learning rate is too high it could jump over the convergence point and if it is too small it will take a long time to reach the convergence point. The convergence point is seen as an optimal solution.

J. Multiple Layer Perceptron (MLP)

The Multiple Layer Perceptron, also called neural network, implements a number of hidden layers of nodes. The hidden layers interconnect the input to the output. Through the hidden layers multiple activation functions and weights are applied to the input. The number of layers and nodes in each layer is hard to choose, since it can lead to different results. The increase it hidden layers means there are more calculations between the nodes in each layer, which can lead to some data having a larger impact on the activation functions. This also applies for the weights being trained in the neural network, which can be more fine tuned by having additional layers. Increasing the amount of nodes in each layer, will increase the amount of weights and activation functions in the neural network. This may lead to a higher training time but may also yield a better result.

III. Experiments and results

This section features all the experiments and result created from using the different methods, described earlier. The results will feature, plots and prediction accuracy's for the classifiers. Every experiment described, has been implemented in python, using the sklearn library to import the classifiers and analysis tools. There is four data sets the classifiers will be tested on, which includes the normal data set, the data set with removed correlated features, the data set with PCA applied and the data set with LDA applied.

A. Principal Component Analysis (PCA)

Figure 2 and 3 shows the scatter plot using the reduced dimension from PCA, whereas there are two PCA components used. The red dots are the class of zeros, and the green are the ones. Figure 2 shows the training set and figure 3 shows the validation set. The PCA is implemented through sklearns PCA.

B. Nearest Centroid Classifier (NCC)

Table II and III shows the results for the nearest centroid classifier with two different distance function, euclidean and manhattan. This is implemented through sklearns NearestCentroid.

TABLE II
NEAREST CENTROID CLASSIFIER WITH EUCLIDEAN DISTANCE

	Normal	***Less Features***	***PCA***	***LDA***
NCC (%)	60	60	56	84

C. K-Nearest neighbor Classifier (KNN)

Table IV shows the result for the K-nearest neighbor with number of neighbors being 1, 5, 10 and 15. This is implemented through sklearns KNeighborClassifier.

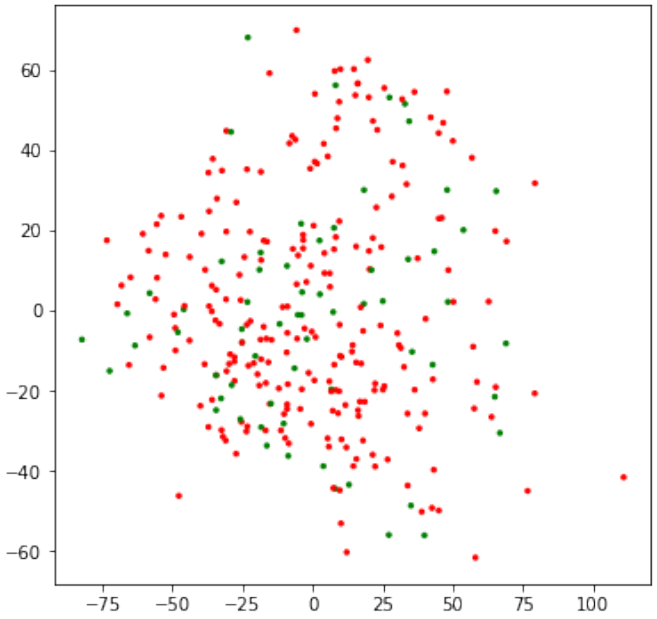

Fig. 2. Scatter plot for 2 component PCA for training set.

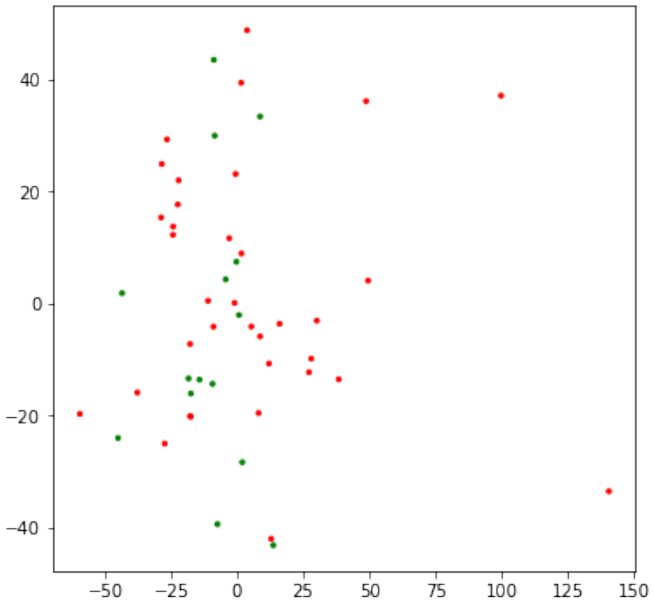

Fig. 3. Scatter plot for 2 component PCA for validation set.

TABLE III
NEAREST CENTROID CLASSIFIER WITH MANHATTAN DISTANCE

	Normal	***Less Features***	***PCA***	***LDA***
NCC (%)	48	64	56	82

TABLE IV
K-NEAREST NEIGHBOR WITH NEIGHBORS (1,5,10,15)

	Normal	***Less Features***	***PCA***	***LDA***
KNN(1) (%)	66	72	66	68
KNN(5) (%)	66	72	70	78
KNN(10) (%)	70	70	72	76
KNN(15) (%)	70	70	70	78

D. Support Vector Machine (SVM)

Table V shows the results for the support vector classification. For the normal and removed data set it does not converge. It also has the feature of balanced weight class, meaning that the imbalanced classes samples are more important, when training. This is implemented through sklearns SVC.

TABLE V
SUPPORT VECTOR CLASSIFICATION

	Normal	***Less Features***	***PCA***	***LDA***
SVC (%)	52	54	52	84

E. Perceptron with Backpropagation

Table VI shows the results for the perceptron trained with backpropagation. There are different learning rates η used, which is 0.1, 0.01, 0.001 and 0.0001. Using sklearns SGDClassifier with hinge loss to implement the perceptron with backpropagation.

TABLE VI
PERCEPTRON WITH BACKPROPAGATION

	Normal	***Less Features***	***PCA***	***LDA***
η: 0.1 (%)	70	70	72	70
η: 0.01 (%)	70	68	54	70
η: 0.001 (%)	30	70	70	70
η: 0.0001 (%)	66	42	64	70

F. Multiple Layer Perceptron (MLP)

Table VII shows the result for the mulitple layer perceptron, with different amounts of perceptrons for three layers. The perceptrons for each layers is (256,128,64), (128,64,32), (64,32,16), (32,16,8) and (16,8,4). This is implemented through sklearns, MLPClassifier.

TABLE VII
MULTIPLE LAYER PERCEPTRON WITH THREE HIDDEN LAYERS

Layers = 3	***Normal***	***Less Features***	***PCA***	***LDA***
(256,128,64) (%)	62	64	60	78
(128,64,32) (%)	66	68	60	76
(64,32,16) (%)	58	60	62	76
(32,16,8) (%)	58	70	66	78
(16,8,4) (%)	66	70	70	80

G. Summary of results

Figure 4 shows the various performances from the classifiers on the validation data set, in graph to see the comparison.

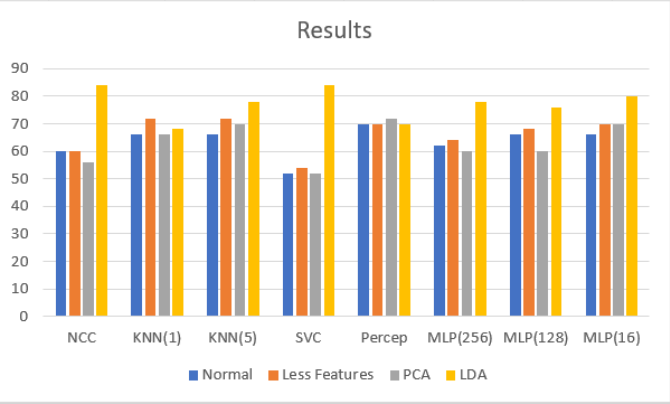

Fig. 4. Results for some of the classifiers

IV. Discussion

Looking at the scatter plots the data is both imbalanced and most of the data is placed at the same spot. This makes it very hard for most of the classifiers predict reliably. Especially looking at the Nearest centroid classifier that creates mean vectors, which would be placed very close to each other. Although when applying the Linear discriminant analysis, the prediction improves, and this makes sense because of maximizing the distance between the mean vectors of the two classes. Using the euclidean distance yields a better result than the Manhattan, also called city block distance.

For the K-nearest neighbor mostly the same for all different neighbor classifications. This classifier also makes better prediction from LDA, it minimizes the variance from data classes, and still maximizes the distance between the classes. It is the same for the Support vector classification in that it performs best on LDA, since it tries to create linear separable cases, and SVC draws a line between the two classes, with margin. Also the SVC weighs the classes differently, making it easier to draw the line. From looking at the scatter plot it would be hard to draw a separation line, which is why the accuracy of the SVC is low on the PCA and the other data sets.

Looking at the perceptrons, the first perceptron classifier has varying result dependent on the learning rate. It has almost the same accuracy for all the experiments with the classifier, but it is best for the PCA at a high learning rate. This ties into the last classifier which is the Multiple layer perceptron which performs best at low amount of nodes, in the hidden layers, and on the LDA data set. This is because the classifier will not become more accurate with additional nodes, because at some points many of the nodes will have the same output, creating zero differences in the final output of the network.

Other ways of increasing the accuracy, is to look at the pre-processing, with the missing data in the data set. There might be other ways to recreate the missing data other than taking the average. This could lead to better performance. It could also be interesting to look at over- and under-sampling or centering the data, to make the classes easier to separate.

V. Conclusion

This paper explored an classification problem for risks in medical treatment, where patient bio-markers and information is used for identifying risks. Different machine learning classifiers was explored and used for solving this classification problem. Herein it revealed that there is multiple classifiers that perform differently. The data set used was very small and the class distribution creates a very difficult task for many of the classifiers explored. Two different dimension reduction methods was used to create different data sets of which the classifiers could predict from, to try create a better classification. For the Linear discriminant analysis the performance for most of the classifiers was increased. In the end the classifiers with the best performances was the Nearest centroid classifier and the Support vector classification, with 84% accuracy.

References

[1] A Iosifidis, Introduction to machine learning, Aarhus university, Department of Engineering, Electrical & Computer engineering, 2018
[2] Python, Sci-kit learn, https://scikit-learn.org/stable/