**Lower Back Pain Symptoms dataset**

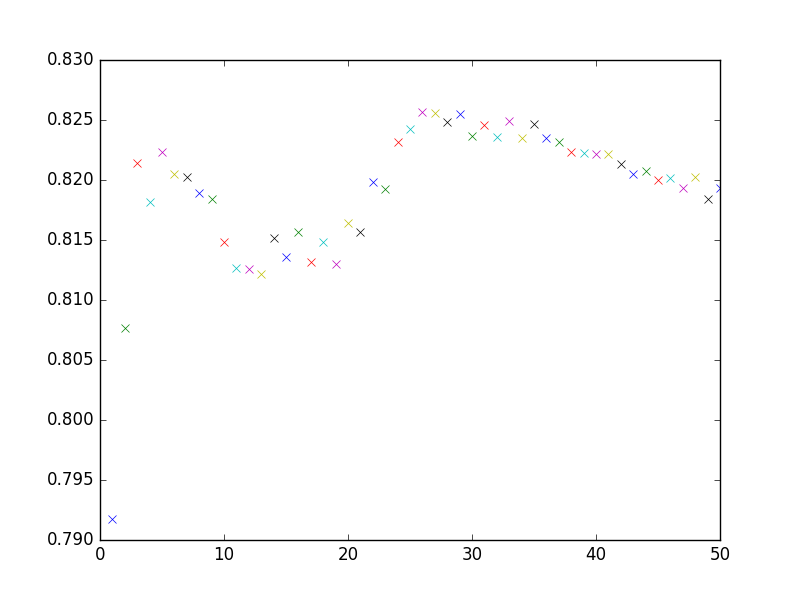
**Problem**

The problem we have is a classification problem. The dataset is about identifying whether a person has an abnormal or normal spine using collected spine details/data. The dataset contains 310 samples. 210 of which are classified abnormal. Every sample contains 12 features with continuous values. Examples of features are scoliosis slope, pelvic radius etc.

**Approach**

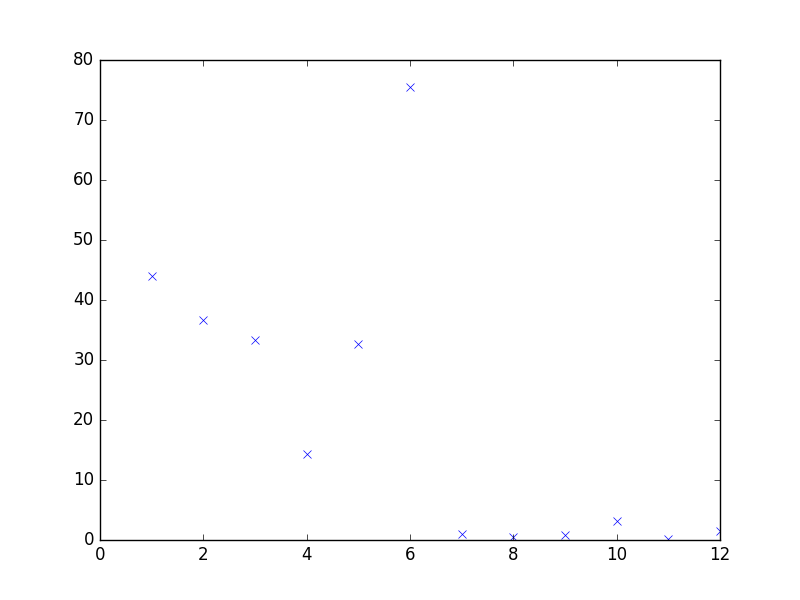
In order to classify normal spines from abnormal spines we will implement a K-Nearest Neighbour (K-NN) algorithm and a Neural Network (NN). The reason why we chose K-NN and a NN algorithm is because we wanted to see if there would be a strong difference is efficiency between a classifier that is capable of making connections between features (NN) and one that can’t (K-NN). We expected there to be correlations between certain features in the case of an abnormal spine. Naturally we chose an NN. Expecting this will result in a better classifier compared to the K-NN

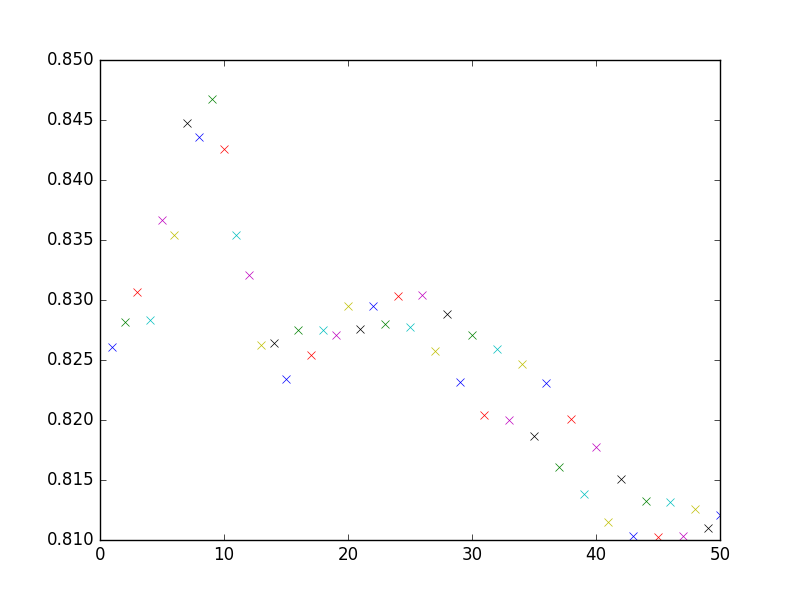
**K-NN**

For the K-NN algorithm we will first choose the optimal value of K. From there we will try to better the score by applying techniques such as feature selection and getting rid of the outliers. In order to find the optimal value for K we implemented the K-NN algorithm multiple times with the K values ranging from 1 to 25. Each time with a random “training” and test set - 80% / 20% - respectively. After this we take the average score of each K (1-25). 

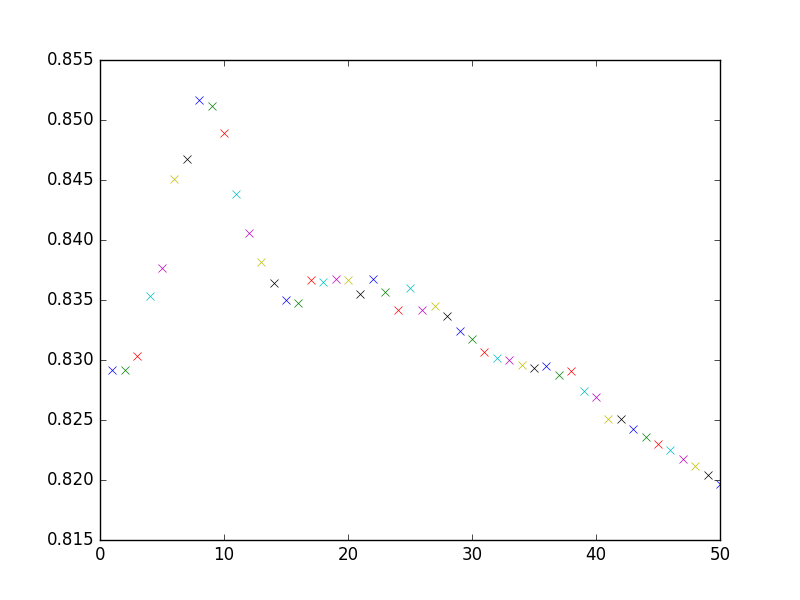
The graph on the right shows two local maxima, one at K=5 and the other at K=27. The global maximum is at K=27, which has an accuracy lying around 82.5%. Based on a reading on Stack Overflow on guessing the optimal value of K our expectations were that the optimal K would be around the square root of the size of our dataset, which is sqrt(310) = 18.[[1]](#footnote-0)

**Improving the K-NN**

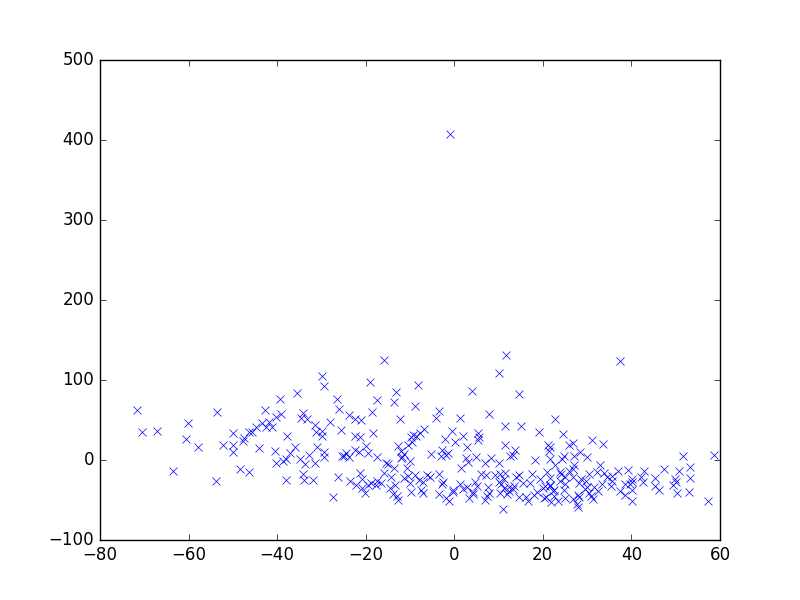
Now that we have a score of around 82,5 % we want to improve it. We will do this by applying univariate feature selection. The best features are based on univariate statistical tests. In the figure below we see the score of each feature. On the y-axis we see the score ranging from 0-100, and on the x-axis are the column numbers which represent the different features (1-12). Here we see that the features 7, 4, 7, 8, 9, 10, 11, and 12 have the lowest scores. This indicates that they are less relevant in making the decision between the Abnormal and Normal classes. 

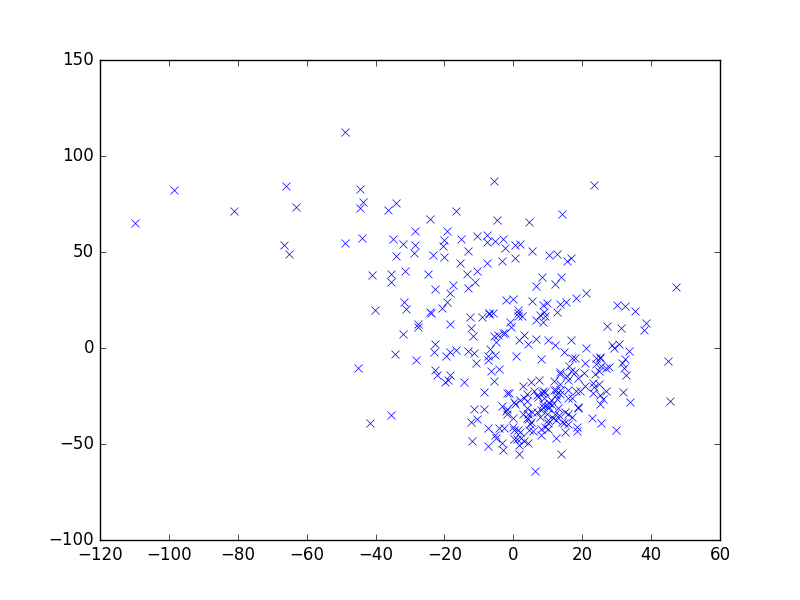
To improve the score of our K-NN classifier we will eliminate the features 4, 7, 8, 9, 10, 11, and 12. By doing so we will also have to look at a new optimal value for K. 

In the figure on the right we see that with feature selection the score is improved with +- 2,25, now returning a score of 84,75 %. The optimal value of K is now K = 9. Noticeable is that there is a alternating score for even and uneven values of K. For K>13 the score is consistently fluctuating. Though decreasing when the K increases. The even values of K are higher than the uneven. We suspect that this has something to do with the fact that the K-NN does not weigh the distances but we are not entirely sure about the reason behind this specific pattern. In the image below we see the results after changing the weights from “uniform” to “distance”. Now the points that are closer weigh more than neighbours that are further away. The weighted distance smoothens the function. Additionally the score is 1 point higher. The optimal value of K is still K= 8 or K=9.



**Removing outliers**

Another way to improve our K-NN classifier was getting rid of the outliers in the data so that there is less noise. In order to do this we implemented the manifold algorithm from sklearn in order to reduce the 12 dimensional data to 2 dimensions and create a image from which we could determine the outliers. In the image below one can see the 2 dimensional visualization. This is one of the many different shapes we got. Even Though the shape was changing every time we ran the code, there was always one big outlier. In this case located in the top middle. After removing this point in row 115 the manifold returned a more clustered set of datapoints.



Unfortunately there was no noticeable improvement. Probably due to the fact that there was only one outlier that was removed.

**Neural Network**

For the NN we will first try to find an architecture that returns the best score. For the architecture of our NN we had to run different structures multiple times in order to see which architecture gave the best performance. The NN is kind of a black box so it is very hard to argue for a certain architecture except that some work better than others in terms of score and f1-score. Our approach for finding the best architecture in the end was by trial-and-error. We tried to find literature which would suggest a possible architecture but none of the literature out there was specific enough to reduce to our case and implement it. Thus we systematically tried tuning the different parameters of the NN. The best combinations of the parameters we tried are as are currently in the program. We did not explore all the possible parameter options because we are unfamiliar with many of the. Changes we made were: changing the activation from the default ‘relu’ to ‘logistic’ and increasing the maximum amount of iterations the NN could run. Finally we arrive at a NN that has 2 hidden layers, each with 12 nodes.

**Improving Neural Network**

The results of the NN were quite difficult to analyse mainly due to the fact that the score we found fluctuated a lot even though we took the average score of 50 iterations (we couldn’t do much more iterations since this already took a long time to run). This being said we tried to improve our NN in the following ways: We changed the default architecture with the hidden layer of (100,) to the hidden layers of (12,12). This change didn’t improve the score much, but it did make the score more stable, which we considered an improvement since it gives more value to the other changes we made, thus we used the architecture of (12,12) hidden layers. Using this architecture to learn on the dataset with all the (training) data, we found that we got a score of around 0.80. Applying the same architecture on the dataset after applying feature selection, we found an equivalent score, if not better. Removing the outlier did not result in any noticeable improvement.

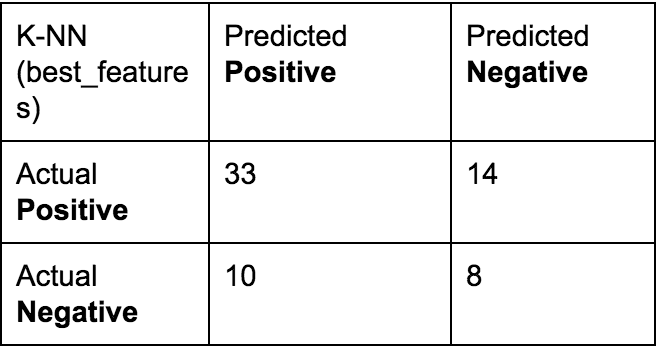
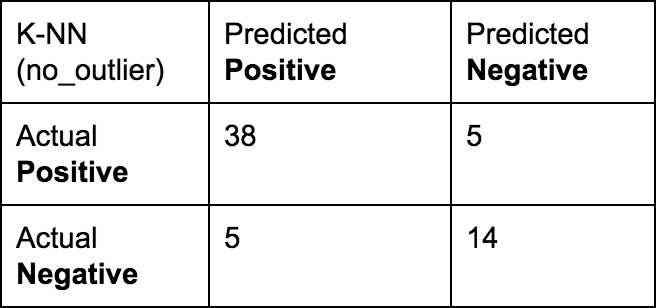
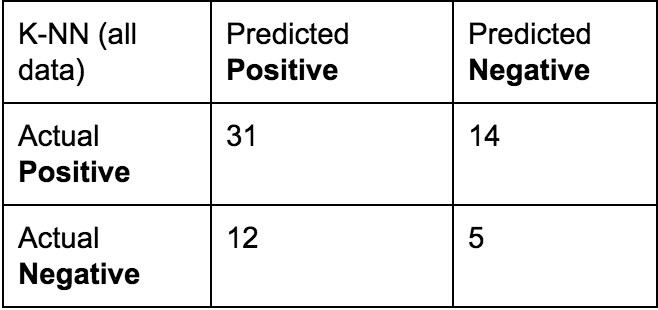
Another method that we used to improve our NN was feature scaling. This had the opposite effect because the score was much worse. With feature scaling the NN ran for a much shorter time but it was not for the better. All in all the average score with the optimal parameters and data is around the 0.8, with this score the related f1-score is on average 0.75.

**Confusion Matrix**

We later decided to include the confusion matrix so the confusion matrices do not hold the values that would give the same scores. They do however closely resemble the graphs and more importantly give us information on the algorithms correct and incorrect classifications.

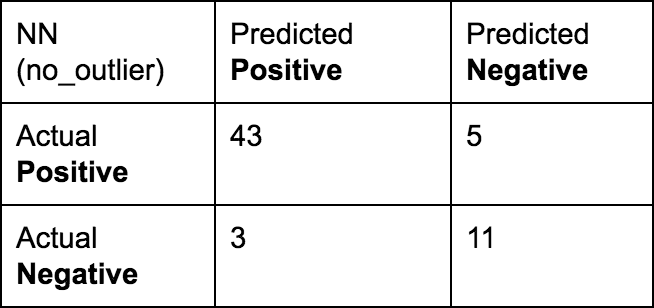
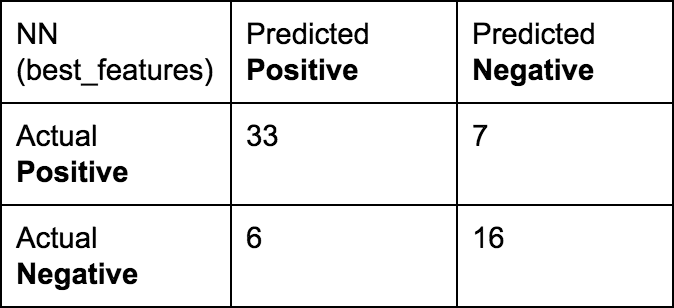
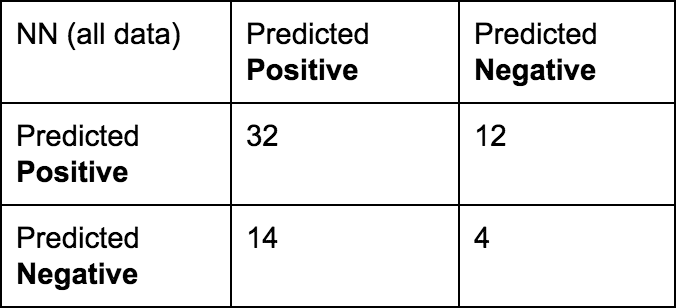
In the case of classifying normal or abnormal spine it is preferable to have false positive (FP) over false negative (FN). In the case of a FP a person has to have its spine checked out for abnormalities, in the case of a FN, however, a person will go home without having been treated for the abnormalities in its spine. This is the opposite from Andrews video’s because in our case the **Positive = Abnormal** **spine** and the **Negative = Normal spine**.

**K-NN confusion matrices**



For now we have to neglect the difference in score because it is not representative of the average score we mentioned earlier on in the report. We can however see that K-NN with the no\_outlier dataset has the lowest FN compared to the FP. Thus even if the score of the K-NN with no\_outlier doesn’t perform any better it is still preferable over K-NN with Best\_features. This conclusion is only applicable for this case (we did not have time to compute averages. This will be mentioned in the discussion).

**NN confusion matrices**



With the case of the NN it turns out that NN with all data has the lowest ratio FN to FP. In terms of misclassifying in the least harmful this would be the preferable option.

**Conclusion**

Using the two classifying algorithms K-nearest-neighbors and Neural Networks, we tried to classify whether a case of back-pain is normal or abnormal based on 12 medical measurements/features. By applying the K-nearest-neighbour we could classify with a score of 0.855 whether the back-pain was normal or abnormal. This result was found by using 8 nearest-neighbors, applying feature-selection on the dataset, and using a weighted distance. In the case of the Neural-Network algorithm our highest average score was around 0.8. This score was found by choosing the architecture with 2 hidden layers with both 12 nodes and the activation we used was ‘logistic’ rather than ‘refu’. Additionally we applied feature selection. This improved the score of the NN slightly. Applying the confusion matrix on the different classifiers with different parameters gives one significant result. Although K-NN with best\_features works as good as K-NN with no\_outlier, the latter is preferable because the errors in classification are less frequently false negatives. For NN the confusion matrices were not very useful because it showed that the optimal FN to FP was in the worst performing algorithm.

\*\*This last conclusion will be mentioned in the discussion\*\*

**Discussion**

For the K-NN the feature selection was an improvement, in fact more than half of the features turned out to be useless. In hindsight this would have this shows that using decision trees is a better choice than NN because the decision boundary does not have to be that complicated.

We also tried to get rid of some noise. This proved to be quite hard when using multidimensional features. Our attempt did get rid of a noisy data point but the effect on the performance wasn't noticeable. This could also be due to the fact that there were more datapoints with strange values. The manifold however did not distinctly showed these points.

The Neural-Network might not have been the best choice for our dataset. It was difficult to work with because we could not look inside the ‘black box’ to see what happens to the data. This makes the whole process a guessing game. I assume that experience -which we lack- with implementing a NN is key to successfully implementing a NN. A NN is very sensitive to overfitting but again we don't know if this is the case because we could not figure out how to look at the weights in the NN. This would have been interesting because in a simple model we could have looked at our how the NN numerically deals with the importance of features instead of applying feature selection ourselves and seeing whether it improves anything. Another thing we could have done is look at the error after learning. If the error is very small we know that the NN is overfitting and does not generalize well enough. This shows us the downside of implementing a NN from sklearn instead of building one from scratch. In the latter situation it would have been much easier to retrieve such numerical information since we would know where to look.

The confusion matrices that we discussed were made using only 1 iteration. This together with the fact that our results (especially the NN) tended to fluctuate the conclusion drawn are not very strong. The reason why the confusion matrices were created using only 1 iteration is simply because it proved to be difficult to implement when using the average of the predicted classes. We were also limited by time.

1. (2012, July 19). value of k in k nearest neighbour algorithm - Stack Overflow. Retrieved December 23, 2016, from <http://stackoverflow.com/questions/11568897/value-of-k-in-k-nearest-neighbour-algorithm> [↑](#footnote-ref-0)