CS5014 - Practical2

140012021

April 2018

1 Introduction

The aim of this practical was to gain experience in working with real experimental, imperfect and limited data which has not been analysed before. The data would need to read in, cleaned and processed. A classification model will need to be created, to predict output classes based on a set of inputs given from the files mentioned in the specification. We will also need to evaluate the performance of the model.

There were two sections to the practical. The first data set that would be looked at would be the binary classification set. This set only had do output options, either green or red. The 2nd data set was a multiclass, which output options were blue, green, pink, red and yellow.

2 Experimental Preparation

Before conducting any experiments to investigate what the best classification model could be for predicting output classes based on a set of inputs, the data that would be used, needed to be inspected. During the first practical data leakage occurred. Data leakage can happen when the data is visualised and analysed before splitting into training and testing sets. To stop this from occurring, the order in which things execute needs to be changed. In this practical first the data is read in from the files that are listed in the specification. This is done by using read_csv which is a function in the pandas library specifically made for reading in csv files. After the data is read in, the shapes of the data is checked. The y values that were read in for both the data types (binary and multi class) had to flattened in order for it to be of an appropriate shape to later be used. Next the data would need to be split into test and training data. This was done through the train_test_split function in sklearn. A random_state was used here so that the data would be split in the same way every time. This was important so that when different classifications was compared with each other, then the same data was used with both.

After splitting the data, the training data from the splits from both the classifications were plotted against the wavelength file. Below in Figures 1 and

2 you can see the output when graphed against wavelength for Binary and multi class respectively.

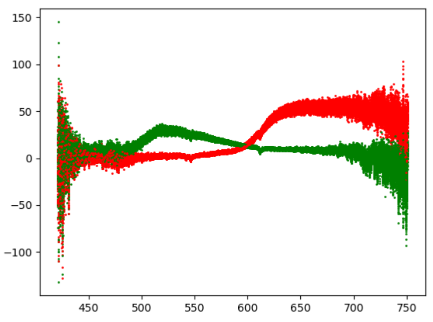

Figure 1: Graph showing the Binary data set results with the wavelengths

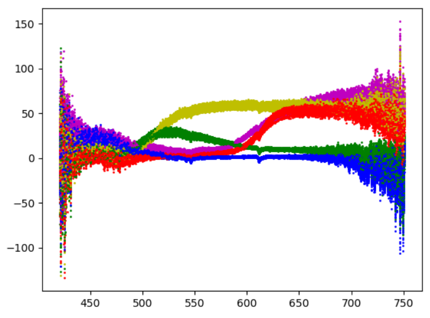

Figure 2: Graph showing the Multi Class data set results with the wavelengths

From inspection of the graphs it is immediately obvious that there is quite a bit of difference in each classification output from different inputs. In Figure 1 the Binary class graph there is only a slight cross over, meaning that for basically any given feature a classification model would return a score of 1.0 (a perfect score). The same can be seen in the multiclass, due to there being more classification options there are more crossovers but there are still many points where there are no crossover.

The above is true except from the very beginning and end of the two graphs. It can be seen from these two graphs that there is so many points from all the different classifications that they could get confused. If a model was struggling to obtain a good classification score, filtering out that data would improve the performance. But as you can see from later results this wasn't a necessarily step. And by not doing so other variations could be looked into for improving the accuracy scores. Also in a real world situation there might be data that is less clean being used, so by working with the unfiltered data a model could be created that would be able to preform on this unclean data set.

3 Logistic Regression

Logistic regression is the more simple of the two model techniques that will be discussed. It works towards categorising the data. Unlike Linear regression which outputs is continues, logistic regression is categorical. Logistic regression uses a function called the Sigmoid function.

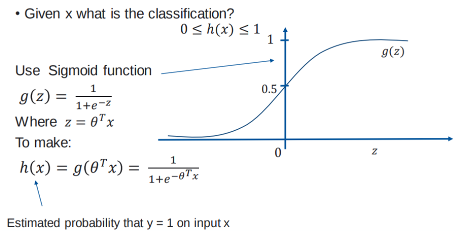
• Given x what is the classification?
$0 \le h(x) \le 1$
Use Sigmoid function
$g(z) = \frac{1}{1+e^{-z}}$
Where $z = \theta^T x$
To make:
$h(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$
Estimated probability that y = 1 on input x
1
0.5
0
z
$g(z)$

Figure 3: Diagram showing logistic regression and the Sigmoid function[1]

The equation for the Sigmoid function g(z) in the figure above is the equation for the Sigmoid function. The θ in the equation represents the weights applied to the input x. When this equation is applied into the Sigmoid function the equation in the above figure $h(x)$. This estimates the probability of the output y on the input x.

4 Neural Network

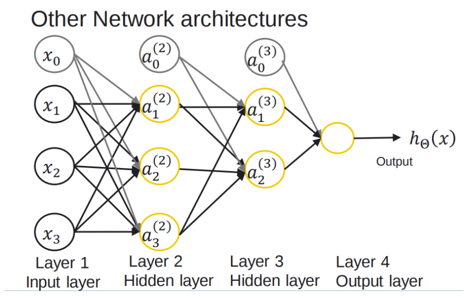

Figure 4: Diagram showing topology of a neural network [2]

A basic summary of a Neural Network is that it uses forward and backward propagation to classify a model. These are carried on through iterations of these functions, until there is no longer improvement on the loss function or the model has converged. Forward propagation works by applying a weight to the input of a node, multiplying them together and feeding it into a node in the next layer. The concept of backward propagation is reduce the overall error of the network. The main difference between logistic regression and a neural network is "between the input and output layer, there can be one or more non-linear layers, called hidden layers" [3] The topology of these hidden layers can be changed and different topologies suit different situations. As well as being able to alter the shape of the hidden layers there are other variables you can alter to change the behaviour of a neural network. These different options will be investigated in the experimental section of the practical.

There were several different activation functions that the neural network could use. An activation function is a node at the end of a neural network to determine the output of the classification. The identity activation solver is useful to implement linear bottlenecks. The function itself is a linear function, therefore the output of the function can have any range attached to it. The next activation function that can be used it the logistic activation function. This works the same as the logistic regression discussed previously in this section. The tanh activation function works similar to the sigmoid function that has been described., except that it has a range from -1 to 1 instead of the range of 0 to 1 that appears in the Sigmoid function. The advantage of having this range is that negative input inputs will be mapped negative and zero inputs will be mapped near the zero in the tanh graph. The final activation function that will be looked at is the ReLu (Rectified Linear Unit) activation function.

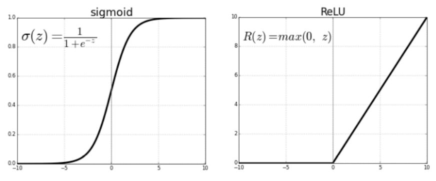

Figure 5: Graphs showing the comparison between Sigmoid and ReLu functions [4]

The ReLu function has a range of 0 to infinity. As all negative values come out as zero. due to the graph of the function being half rectified at the bottom.

There were also different solvers that were used. The first was was the lbfgs solver which is an optimiser from the Quasi-Newton methods. Which means they are used to find either zeroes or local maxima and minima of functions. It's a popular algorithm to when "optimizing the parameters of large-scale log-linear models" [5] The other two solvers that were used were sgd (Stochastic gradient descent) and adam, which is an extension from the sgd solver. Stochastic gradient descent maintains a single learning rate for all weight updates, and the value of the learning rate will not be altered during training. For adam there is a learning rate for each weight in the neural network and they are separately adapted as the learning unfolds. This improves performance of the network especially on problem with sparse gradients.

All of these variations of a neural network can help it preform in different environments. A neural network needs to be tailored to an situation.

5 Experimental Results

In the last practical, randomness was used when selecting the features to use. To improve on this, a function called SelectKBest was used. Again this is a function found in sklearn. The way that SelectKBest works in that it scores all the features using a function. The function that was chosen for this was f_classif. SelectKBest also take a value that changes the number of features that it returns. It returns this number of features by removing all but the k highest scoring features.

Every experiment was run with the same concept. Each experiment would be run on the best feature, then the 2 best features and so on until all 921 features were being used in the experiment. The 3 different experiment forms that would be used, would be checking the time required to fit the model, the time to predict the model and the average accuracy score obtained. All of these would be run with different models and set ups.

The first run that was made was using basic logistic regression on the binary data set. This was chosen as it would be the most computationally simple model and data set. The graph in figure 3 was obtained.

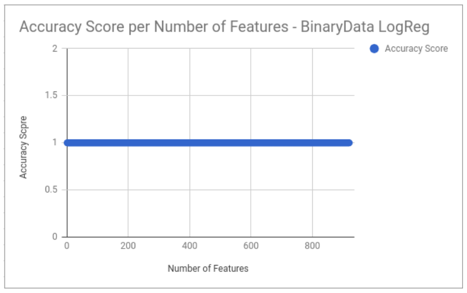

Figure 6: Accuracy Score for Logistic Regression on the Binary data set.

The graph clearly shows that an accuracy score of 1.0 was obtained. This score, as with the other scores that would be calculated in the remaining experiments would be done using the accuracy_score function from sklearn. With basic logistic regression returning a perfect score, it was then tried on the more complicated data set, the multi class data set.

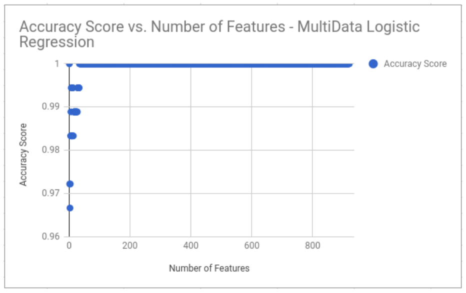

Figure 7: Accuracy Score for Logistic Regression on the Multi Class data set.

The above graph in figure 4 also shows that basic logistic regression could be used to obtain a accuracy score of 1.0 across basically all features and set of features. After realising this, the above experimental set ups were thought of and implemented. Instead of investigating how to obtain a perfect score, which is trivial, as seen with the graphs in Figures 1,2,6 and 7.

The other model that was used to classify the input data was a neural network. The default settings from sklearn were used for the neural network. The neural network was then run on the same data as the logistic regression model. The graphs in figures 8 and 9 were produced when running this model on the data.

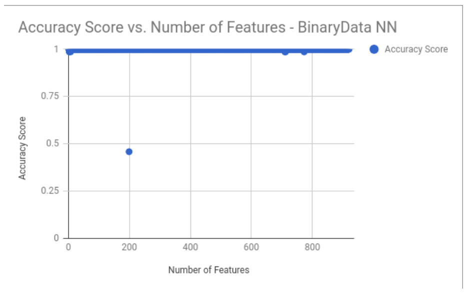

Figure 8: Accuracy Score for neural network on the Binary data set.

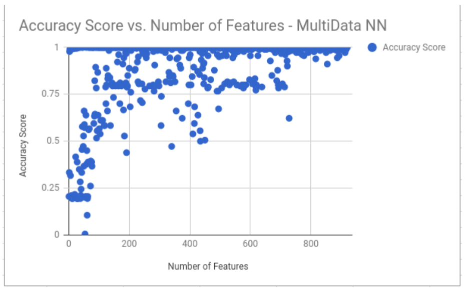

Figure 9: Accuracy Score for neural network on the Multi class data set.

Similar to the the logistic regression the neural network preformed incredibly well with the binary data, nearly having a perfect score, other than a few outliers. The difference is when it comes to the multi class data set. There is a definite curve visible in the data as the number of features increases. The table shows the average accuracy score over the different number of features for each model on each data set.

Model	Binary	Multiclass
Logistic Regression	0.998914223669924	0.99848594522861622
Neural Network	0.99825069369043296	0.92869465556761854

The table shows that the model and data set that does not preform nearly as well as the other combinations is that of the multi class data set and the neural network. It was then decided that this would be further investigated to see if this value could be brought up to the other values. This would be done through looking at the activation function, solver and different layers that could be used.

Firstly different activation functions were looked at, figures 9-11 you can see the diff

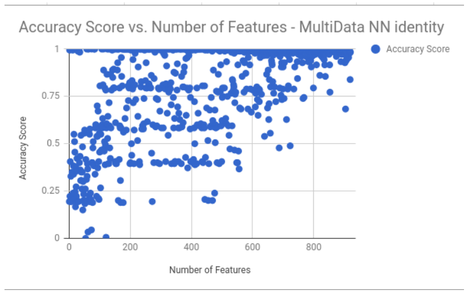

Figure 10: Accuracy Score for neural network (with identity) on the Multi class data set.

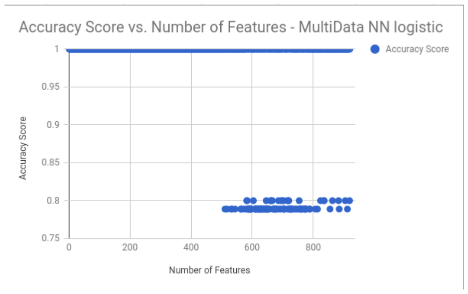

Figure 11: Accuracy Score for neural network (with logistic) on the Multi class data set.

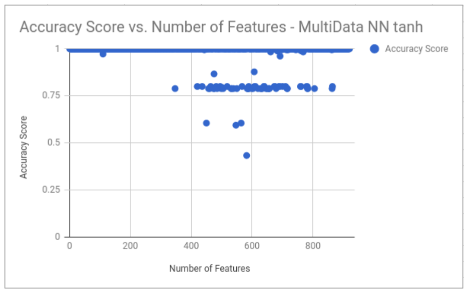

Figure 12: Accuracy Score for neural network (with tanh) on the Multi class data set.

The identity function does not work as well as the other functions. I believe this is due to the linear nature of the solution it offers. The data appears to have linear scale to it in figure 10. The logistic activation function has worked much better than the the original settings. This wasn't surprising as it was going to function the same as the logistic model from earlier. Finally the tanh activation function was used in figure 12. This again preformed the previous activation function. By the tanh graph handling negative numbers better it preformed slightly better than the Sigmoid function, in the logistic activation function. The average scores for these activation functions can be seen below. From the table you can see that tanh performed the best.

Activation Function	Accuracy Score
identity	0.818397877670406583
logistic	0.976716129810591533
tanh	0.983532392327179711

Next the different solvers that could be used with the model were experimented with. The results for the accuracy scores they achieved for the different numbers of features can be seen in figures 13 - 14.

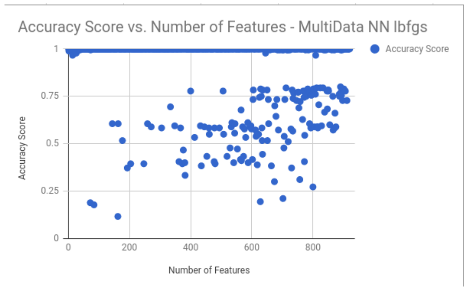

Figure 13: Accuracy Score for neural network (with lbfgs) on the Multi class data set.

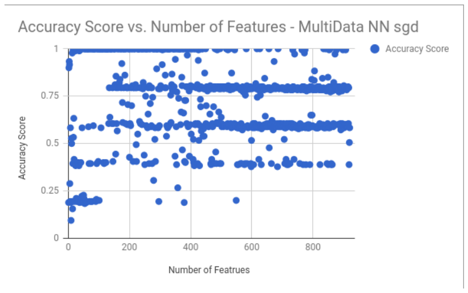

Figure 14: Accuracy Score for neural network (with sgd) on the Multi class data set.

Figure 14 is incredibly interesting. There are nearly 5 distinct rows in the data output, which is equal to the number of classifications the model was using. The plot was made in Google sheets where the graph is interactive. There you can see what number each of the points is. When looking into this you can see that when going through the data in the order it was plotted, you can see that it plots alternatively throughout the different layers. This would suggest that the learning rate is too high on the model. It would keep flipping backwards and forth between what it believes is correct. So for sgd solver in the neural network a lower learning rate would need to be used in this example. The average score for each of the solvers is in the table below. The table shows that these solvers did not improve the performance of the network massively.

Solver	Accuracy Score
lbfgs	0.93426830739534239
sgd	0.73748341175051357

The final variation that was looked into with the neural network was the different layers that could be used in the neural network. In comparison with the original set up of the neural network at the beginning of the section, the neural network this time contained more layers, each of which had a greater depth. By including more layers and nodes in the neural network, the bias of each value is reduced as it is spread further through the network. The output layer for the model will always be equal to the number of classifications the model was doing.

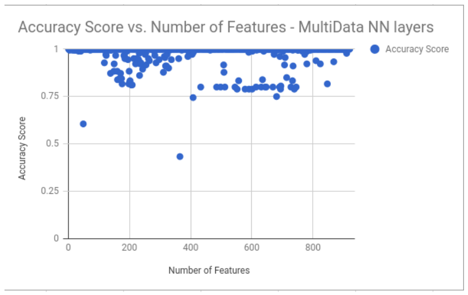

Figure 15: Accuracy Score for neural network (with different layers) on the Multi class data set.

Solver	Accuracy Score
Increased Layers in Model	0.98587887561828891

By increasing the number of layers in the model the score has significantly increased. This due to the bias of each value being decreased. However it is important to remember that different combinations of layers can be better for different scenarios and further experimentation would be needed to conclude an optimum layer configuration or the data.

Overall after looking at how the different variations could improve how a neural network preforms a final model was created taking the variations that produced the best results. These were tanh for the activation, adam for the solver and a increased layer size. When these options were set in the neural network the following results were obtained.

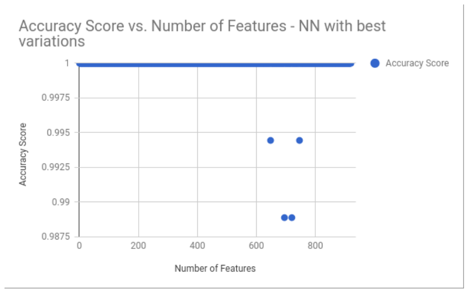

Figure 16: Accuracy Score for neural network (with best combined settings) on the Multi class data set.

Model	Accuracy Score
Neural Network with improved options	0.99887803112558815

Figure 16 and the table above show that the newly improved neural network has preformed very well. It's accuracy score now higher than the logistic regression from earlier, by a very small margin. However this does not make this version of a neural network the best model as there are other considerations to be made such as training and prediction times.

The time taken for the fitting and predicting for all the variations of models mentioned in the report thus far was recorded and graphed. All of these will be included in the Appendix with a few being highlighted in this section. In a real world situation the data could be significantly larger, so a model that takes a long time to train on a small data set the problem would be magnified on the larger data set. Also there can be scenarios where prediction needs to be very quick, such a system that is using machine learning for live predictions. Below in Figures 17 - 20, the fit and prediction times for the logistic regression model and the final neural network are compared.

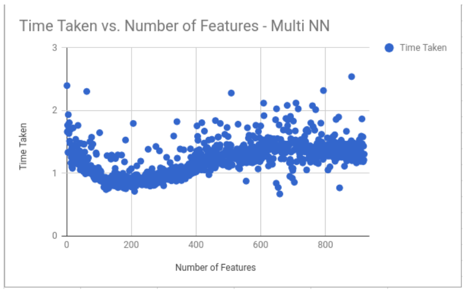

Figure 17: Time to fit the final neural network for each number of features

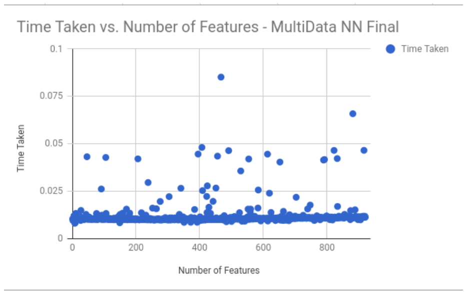

Figure 18: Time to predict the final neural network for each number of features

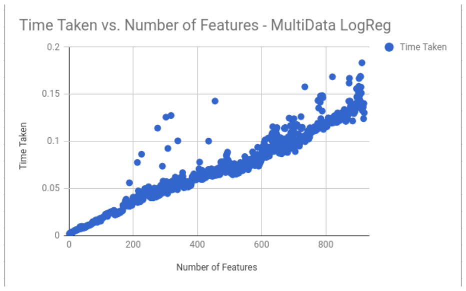

Figure 19: Time to fit the logistic regression for each number of features on multi data

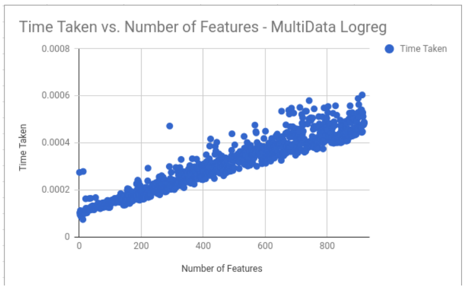

Figure 20: Time to predict the logistic regression for each number of features on multi data

In figure 17 you can see a curve of a sine like nature. It is hard to tell if this would be flattened out over a longer experiment. This would be something that could also be looked into further in depth experiments. The curve has been produced between 1 and 2 seconds. This is a large training time for the system when it has very few features. The training time however seems to not increase as the number of features increase. So if the data and number of features were increased in size, the training time would not increase a huge amount, making it a scalable model. If the model was needed for live prediction, such as a project like RadarCat[6] then the neural network would be a good system to use as the prediction time is low and constant for all the different numbers of features that could of been used.

When analysing the logistic regression both the fit time for the model and the prediction time are of a linear shape. For both fit and predict for the model they increase as the number of features increase. This is purely due to way the data is fed through a logistic regression model as we discussed earlier. For the data set we were provided the time taken to fit the model is lower for logistic. If the number of features increased though, the fit time would eventually get worse, when compared to the neural network. So for larger data sets the neural network would be a better model. Again this is shown in the prediction time, the logistic regression model is quicker for predicting, but if the number of features increased the prediction time for the logistic regression model would eventually get worse than the neural network.

The reason the logistic regression model was chosen to produce the final solution for both data sets was that, firstly we were tasked with getting the most accurate prediction, which narrows it down to the two models above. But due to the design of the data set it would be better and more realistic to use the logistic regression model to produce the predicted output. This output can be seen the appendix.

6 Evaluation

Overall I feel I have completed the practical well and in depth. I have created a model to predict the output of the files that needed to be classified. I also created many models, altering how they work to attempt to find out what model using what settings would be best for this scenario. I also discussed in depth how each of the models worked, what changing each variable would do. I also feel I explained why certain things happened, such as the graph made due to the over fitting discussed earlier. Finally I analysed what model would be best for this scenario and what each model would be best suited too in a real world environment.

I took on the feedback from the last practical, looking into data leakage and trying to prevent this from happening. I looked into and implemented a better selection method than random, using select k best instead. Finally I tried to describe everything in more detail.

7 Conclusion

Overall I feel I have gained experience working with real experimental data, learning experimental skills in the process. Finally I feel I have learned about how neural networks operate, including their different settings.

8 References

[1] https://studres.cs.st-andrews.ac.uk/CS5014/Lectures/CS5014_L07_logistic_regression_part2.pdf - CS5014 lecture 7 Logistic Regression part 2 by Dr David Harris-Birtill.

[2] https://studres.cs.st-andrews.ac.uk/CS5014/Lectures/CS5014_L15_neural_networks_part1.pdf - CS5014 lecture 15: Neural Networks part 1 by Dr David Harris-Birtill.

[3] http://scikit-learn.org/stable/modules/neural_networks_supervised.html

[4] https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[5] https://www.microsoft.com/en-us/research/publication/scalable-training-of-l1-regularized-log-linear-models/

[6] Hui-Shyong.Y Flamich.G Shrempf.P Harris-Birtill.D Quigley.A, Radar Categorization for Input Interaction (2016)

9 Appendix

Output for Binary Data Set
1
0
0
1
0
0
0
1
0
0
1
1
1
0
1
1
0
1
0

Output for Multi Class Data Set
0
2
0
0
0
2
0
4
1
4
3
3
2
0
4
2
4
3
3
4
1
2
1
4
2
3
2
0
1
1
1
3
1
2
4
2
3
1
4
0
0
1
1
3

Output for Multi Class Data Set Continued
0
3
4
3
4

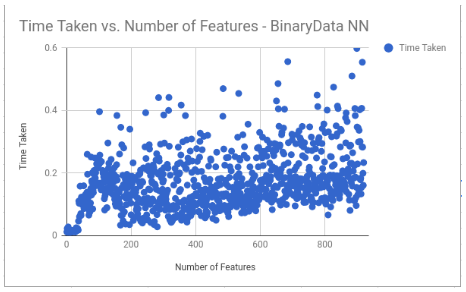

Figure 21: Time to fit the neural network for each number of features on binary data

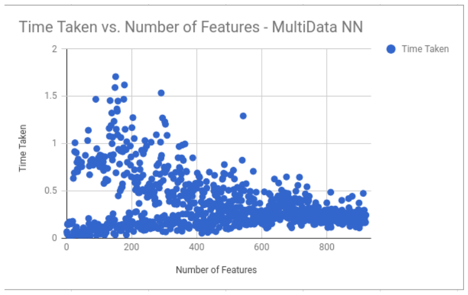

Figure 22: Time to fit the neural network for each number of features on multi class data

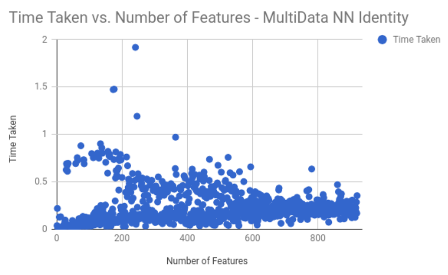

Figure 23: Time to fit the neural network (with identity function) for each number of features on multi class data

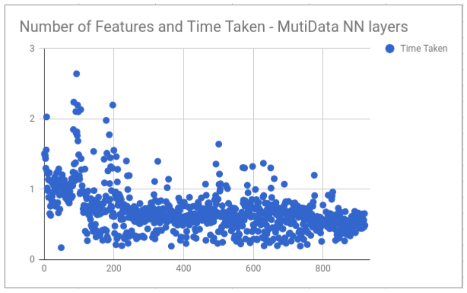

Figure 24: Time to fit the neural network (with different shape of layers) for each number of features on multi class data

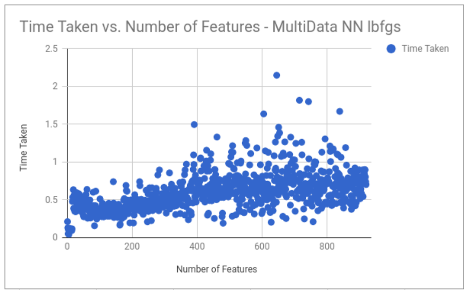

Figure 25: Time to fit the neural network (with lbfgs solver) for each number of features on multi class data

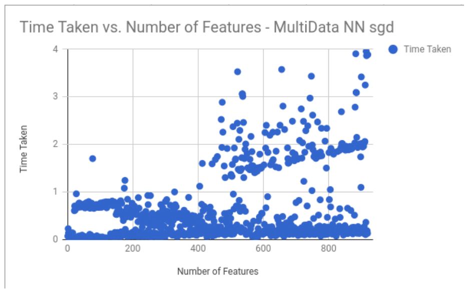

Figure 26: Time to fit the neural network (with sgd solver) for each number of features on multi class data

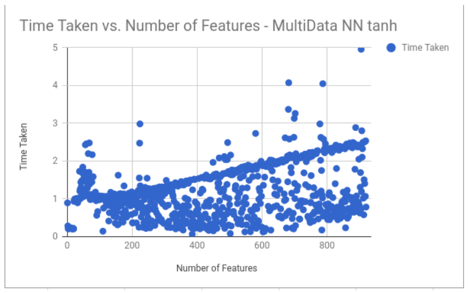

Figure 27: Time to fit the neural network (with tanh function) for each number of features on multi class data

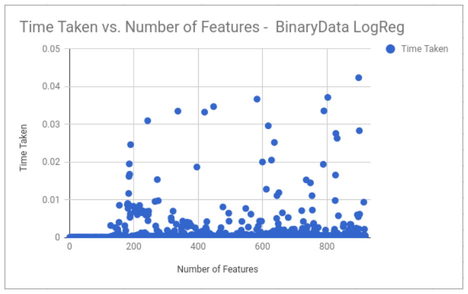

Figure 28: Time to predict with logistic regression for each number of features on the binary data

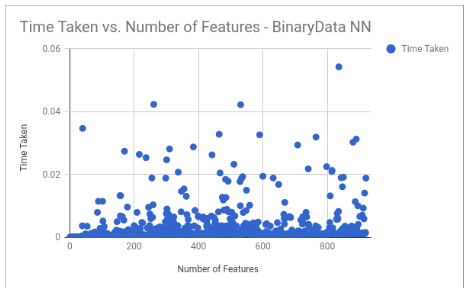

Figure 29: Time to predict with neural network for each number of features on the binary data

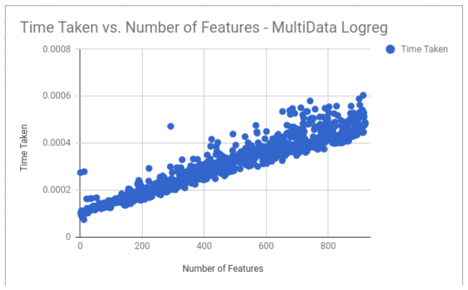

Figure 30: Time to predict with logistic regression for each number of features on the multi class data

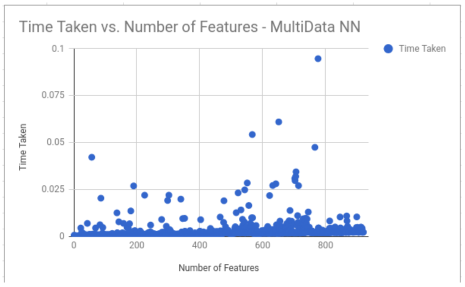

Figure 31: Time to predict with neural network for each number of features on the multi class data

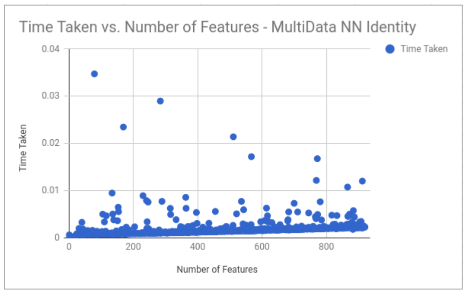

Figure 32: Time to predict with neural network (with indetity function) for each number of features on the multi class data

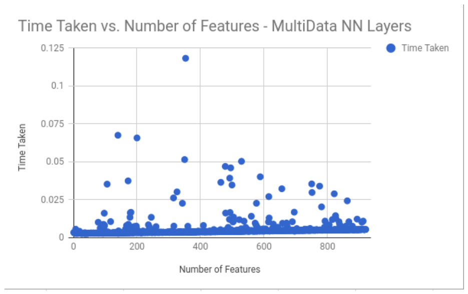

Figure 33: Time to predict with neural network (with different shape of layers) for each number of features on the multi class data

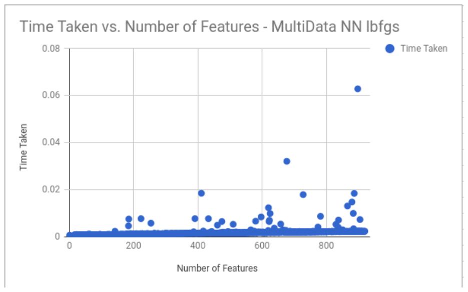

Figure 34: Time to predict with neural network (with lbfgs solver) for each number of features on the multi class data

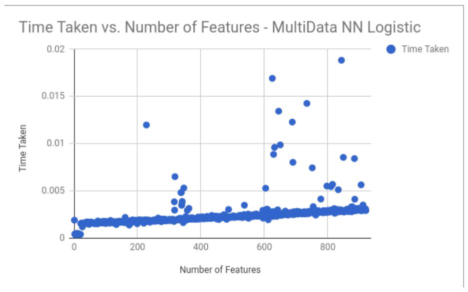

Figure 35: Time to predict with neural network (with logistic function) for each number of features on the multi class data

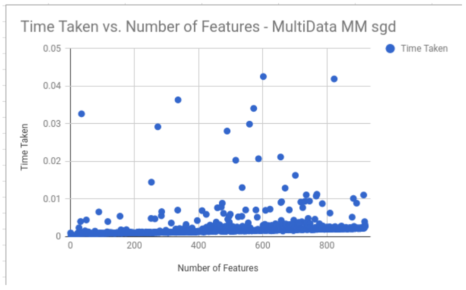

Figure 36: Time to predict with neural network (with sgd solver) for each number of features on the multi class data

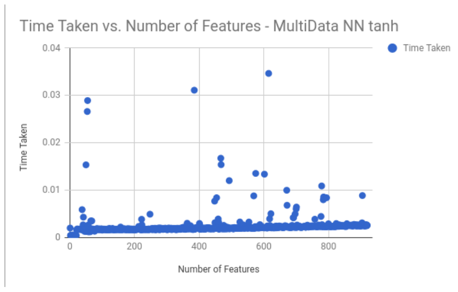

Figure 37: Time to predict with neural network (with tanh function) for each number of features on the multi class data