Analysing Mice Protein Expression Levels and Predicting Class-Type

Wyatt Lawrence Jenkins
Student, Practical Data Science (COSC2670)
RMIT University
College of Science, Engineering and Health
Student Number: s3770872
s3770872@student.rmit.edu.au
Melbourne, Australia

Date of report: 29/05/2020

Table of contents:

Abstract
This study focuses on the analysis of a UCI dataset containing the expression levels of 77 different proteins found in the cerebral cortex of mice. The range of mice studied include normal mice and trisomic mice (mice with down syndrome), mice injected with memantine and mice injected with saline, as well as mice who have been stimulated to learn (context-shock) and mice who have not (shock-context). These 3 differences between mice make up 8 (2^3) different classes of mice. This study attempts to differentiate these classes based on the recorded proteins of each sample mouse through classification modelling. Two different modelling techniques; k-nearest neighbour and decision tree classification are both employed and compared in this study to distinguish classes of test mice. Features are selected using the hill climbing technique and the results of the models are validated via a 50/50 train/test split as well as the *k*-folds cross validation strategy. The relationships between the proteins and the experiment controls are also observed.

1. Introduction

The dataset being studied in this paper contains 38 control mice and 34 trisomic mice (72 mice in total). For each mouse, the protein expression levels across 77 distinct proteins were recorded 15 times. This evaluates to a total of 1,080 (72 * 15) different recordings of each protein. The expression levels of each protein were recorded in the nuclear fraction of the cerebral cortex in each of the mice, hence each of the 77 protein names is followed by the substring:

'_N'. The class labels given to each mouse were in a 'x-YY-z' format, where 'x' is the genotype, 'z' is the treatment used and 'YY' represents whether the mouse has been stimulated to learn or not. The table below distinguishes the 8 different classes of mice featured in the dataset.

Table 1 – Different classes of mice in the data set.

Class	Details
c-CS-m	Normal mice, stimulated to learn, injected with memantine
c-SC-m	Normal mice, not stimulated to learn, injected with memantine
c-CS-s	Normal mice, stimulated to learn, injected with saline
c-SC-s	Normal mice, not stimulated to learn, injected with saline
t-CS-m	Mice with down syndrome, stimulated to learn, injected with memantine
t-SC-m	Mice with down syndrome, not stimulated to learn, injected with memantine
t-CS-s	Mice with down syndrome, stimulated to learn, injected with saline
t-SC-s	Mice with down syndrome, not stimulated to learn, injected with saline

Advancements in machine learning have led to large datasets being more and more valuable in business as well as research. Modern algorithms can take a set of rows containing information and classify each row depending on what the data contains. Data which might appear to be without a pattern to the naked eye can be dissected by such algorithms and be placed into categories or classes with impressive accuracy. Much like how spam is detected in your email inbox, the same algorithms can classify the mice in this data set into one of the above based only on the protein expression levels. This study will attempt to achieve that goal using k-nearest neighbour and decision tree classification to determine which of the classes a given mouse belongs. The methodology in this paper details the cleaning of the data set, the statistical findings in this dataset given some hypotheses and how the classifiers mentioned were utilised for this dataset to classify each mouse. The results are then discussed and finally a conclusion is given, including a recommendation on a classification algorithm for determining the class given a set of mice protein expression levels.

2. Methodology

2.1 Data Cleaning

Neither descriptive statistics nor data mining will be successful with a data set ridden with errors. It is vital that any data set is polished for the results of any exploration or classification to have any value. Therefore, the rule is that cleaning the data comes first and this study makes no exception.

It should be noted that all technical tasks performed in this study were completed in *Jupyter Notebook*, using the *Anaconda3* distribution for *Python*. The Python libraries used were among *pandas*, *numpy*, *matplotlib*, *seaborn* and *sklearn*.

First, the data set was downloaded from the UCI machine learning repository and was in the form of an excel (.xls) file. A quick look in *Microsoft's Excel* confirmed that the data set contained its own column headings and they were sufficient for the purpose of studying the data, so they were kept.

Next, the data types were checked. All the columns containing the protein expression values were floating point types and the 4 columns containing the nominal data held objects. As these are the ideal data types, they were left unchanged.

Any missing values were then counted for each column. The columns containing the nominal data had no missing values, but some proteins had missing cells. Since the data will be fed to classification algorithms at a later stage, it is best not to have any missing values in the data set. The decision was made to impute the mean value for each protein to its respective missing cells. Unfortunately, it does add some level of disingenuity to the data, but it does mean that we get to keep the other variables in the observation and it was decided that losing important information was too big of a price to pay.

As protein expression levels can only be positive, it then follows that there cannot exit any negative values in the data set. A sanity check was performed on the data using a small piece of *Python* code which searched through all the protein expressions and returned all the negative values. Fortunately, there was only one negative value throughout the ~83,000 expression levels. This value was changed to the mean value of that column for the same reason as above.

Finally, all four object columns had all strings stripped for good measure. A count of distinct values was subsequently performed for each nominal-data column to locate any typos in the string values that these features contained. No such typos were found and with that, the data is ready for exploration.

2.2 Data Exploration

After the had been cleaned, the proteins relative to the controls were explored in various ways. It was noted that certain controls in this data could produce differences in the protein expression levels within the nuclear cortex of mice. Given this possibility, some hypotheses were developed with the attributes within this dataset in mind. About 10 different hypotheses were explored and, in this study, those that led to the most interesting results are noted. Some of hypotheses explored were:

Hypothesis #1: There are some classes of mice that will exhibit higher levels of a particular protein than others.

This hypothesis was explored based on an arbitrarily chosen protein (the NR2A protein) to see if there was any notable difference in the protein expression levels when sorted by class.

A boxplot was chosen to visualise the relationship as it showed the central tendency of the protein for each class as well as any potential outliers.

Hypothesis #2: The effect of treatment has produced some difference in some protein expression levels.

As treatment via injection would have consequences on the brain of the mice, it seemed natural that protein levels would change as a result of a saline or memantine injection. Here, I explored how different mice were affected by memantine and saline in two proteins: pCREB and BRAF.

These two proteins were compared with each of the two treatment types via a scatter plot where the Memantine and Saline data points were distinguished by colour. A scatter plot was chosen to show this information as it captured the whole story between the continuous data (proteins) and categorial data (treatment type). It was easiest to see the distribution of protein expression points in a scatter plot and compare the saline treated mice with the memantine treated mice.

Hypothesis #3: Whether or not a mouse is stimulated to learn will have a large effect on protein expression levels.

A bar graph which was generated as part of exploration showed that the CaNA and pCAMKII proteins had the largest discrepancies in the mean value of these proteins between those mice that had been stimulated to learn (C/S) and those who had not (S/C).

Using a scatter plot, these proteins were explored further with a comparison between C/S mice and S/C mice. Again, a scatter plot best demonstrated any relationship here as there was no loss of the continuous aspect of the protein data and the data plot points were separated by colour for the categorical part of the data.

2.3 Data Modelling

2.3.1 The *k*-Nearest Neighbour Classification Algorithm

The first model employed was a classification model called *k*-nearest neighbour. The k-nearest-neighbour technique is a supervised learning classification algorithm. It is a simple yet widely used technique that has many applications. The first step in data modelling is to select the appropriate features for our model. There are several ways to accomplish this, however, in this study we will focus on the hill climbing technique. The hill climbing technique involves taking a subset of candidate features (in this case, a subset of proteins) and continuously modifies the subset if the new subset returns a greater result than the previous iteration. Once a new subset returns a result that is lower than the previous iteration, the hill climbing stops as it has effectively reached an optimal subset; a 'peak'. Performing this technique on the mice protein dataset, hill climbing returned a subset of 50 different proteins of the 77 in total.

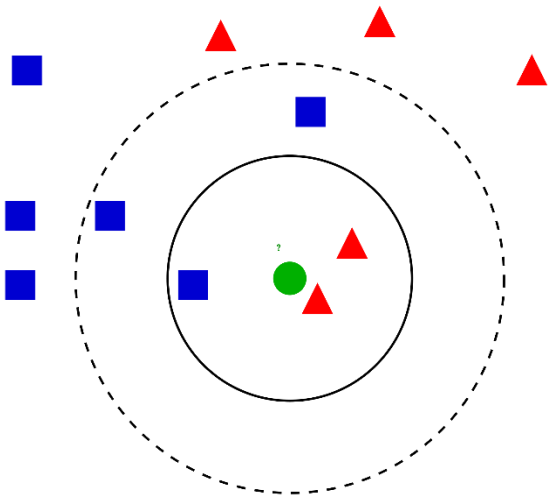


Figure 1- KNN Classification Diagram

These 50 features were then fed into a basic k-nearest neighbour classification with the default parameters. This algorithm will take a new observation (i.e., a new mouse), look at what class each of the *k* closest surrounding neighbours fall into, then give the new observation a class depending on what its *k* nearest neighbours are classified as. As the *k* value increases, the number of existing observations that are compared to the new observation increases. In this case, the default *k* value is 5, so our new mouse is being compared to the 5-nearest observations. Adjusting this parameter, the accuracy increased as we went lower. A

value of 3 seemed most suitable as it was not too low as to overfit the data but also was not too high that it was underfitting the data. A change from 5 nearest neighbours to 4 increased the accuracy of the model on the 39 selected features. A change from 4 to 3 increased the accuracy of the model even further.

The next parameter to consider was the weights. There are two different weights to consider: 'uniform' or 'distance'. The default value is 'uniform' which means that the distance from each of the *k* data points to the observation is not taken into consideration in modelling. Changing the weights parameter to 'distance' improved the accuracy of the model on the mice protein data set as it then factored in the distance between existing observations and the new observation. Given any two existing observations, the one that is closer to the new observation will be given more weight than the one that is further away. Adjusting the weight from the default 'uniform' value to 'distance' improved the accuracy of the model.

Two other parameters to consider are the 'metric' and 'p' parameters. These parameters are responsible for calculating the distance between the new observation and an existing one, as previously discussed. The p value is parameter for the Minkowski distance metric. If a value of 1 for p is chosen, this is the equivalent of the Manhattan distance, if a value of 2 is chosen, the Euclidean distance is used. For any other p value, the Minkowski distance between the observations is given by Equation 1. A *p* value of 1 produced the best result in the KNN model for the mice protein data set, which is equivalent to the Manhattan distance metric.

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

Equation 1- Minkowski Distance Equation

The model was then trained and evaluated using two different methods. The first method was using an even split with the data where a pseudorandom 50% of the data was used for training the model and the other 50% was used for evaluating the performance of the model.

The second method used was the k-folds cross-validation technique which splits the data into *k* parts and uses each part as a test set with the rest used as training. This gives a more comprehensive evaluation of the model as it uses all the data in the entire data set for training and testing. In this case, the data was split into 8 folds. The comparison between each validation method is further detailed in the results section of this study.

2.3.2 The Decision Tree Classification Algorithm

After obtaining a result for the *k*-nearest neighbour algorithm, we then used a different technique for modelling the data as to compare two different models. This time, the decision tree classification algorithm was used. Like *k*-nearest neighbour, the decision tree is also a supervised classification technique which has the same goal – to classify new observations of mice into 1 of 8 classes, given a set of protein expression levels.

In the same way we selected proteins for the k-nearest neighbour algorithm, we will again select proteins based on the results of hill climbing optimisation. In this instance, a hill climb using the default decision tree parameters returned 17 distinct protein features. These features were then passed onto a decision tree classifier.

The decision tree classifier works by basically taking a feature, giving a condition to that feature, then splitting the observations based on whether they meet the condition or not. It does this continuously through all features until it arrives at a leaf node – the leaf node represents the classification given to that observation.

Figure 2 demonstrates a basic decision tree as a visualisation. The conditions including variables X, Y, Z and W are the conditions, the edges that branch from these nodes are the results, and A, B, C and D are the resulting leaf/terminal nodes which represent the classes the observations have been placed.

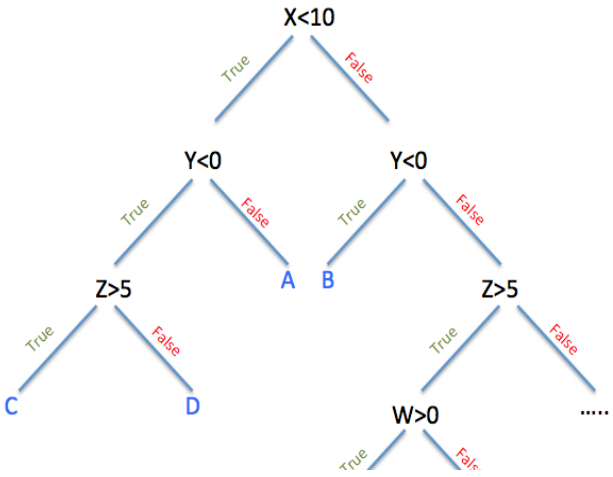


Figure 2 – Basic Structure of a Decision Tree

The decision tree classifier is more tedious to fine-rune than the *k*-nearest neighbour algorithm. The reason for this is the number of parameters in the decision tree is quite large and the range of possible values for those parameters is expansive. The first parameter to consider is the criterion parameter. This parameter accepts one of two string values: 'gini' or 'entropy'. The 'gini' value refers to the gini index which creates binary splits in the tree where the greater the Gini index, the higher the homogeneity. The equations for the Gini index and entropy are detailed in Equation 2.

$$Gini: Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$
$$Entropy: H(E) = -\sum_{j=1}^{c} p_j \log p_j$$

Equation 2 - Gini Index and Entropy Equations

For this dataset, the Gini index was used. The reason is that the theoretical difference between the Gini index and entropy criterion is very little in terms of decision tree performance. Gini index is, however, faster to compute due to the logarithmic function in the entropy equation, hence Gini index was chosen as the parameter for 'criterion'.

The 'max features' parameter determines how many features to look at when deciding the best split. For this data set, we did not limit the maximum features as throughout the modelling process, there was not much risk of overfitting this data. More on overfitting will be covered in the results section of the study.

The minimum samples per leaf parameter dictates the smallest quantity of samples that can be at a leaf node in the tree. Therefore, a split cannot occur if it leaves less than the minimum number of samples in this parameter at a leaf node. The best model accuracy seemed to be achieved at a minimum leaf samples of 3, so this is the value used for this parameter in the model.

Another parameter, minimum samples split, is similar to the minimum samples per leaf except that this prevents a node from splitting if it has less than the number of samples given by this parameter. After tuning this variable, a value of 8 provided the best result for the accuracy of the decision tree.

To give a fair comparison between the classification ability of the decision tree versus the *k*-nearest neighbour algorithm, the decision tree model was evaluated in the same way the KNN model was evaluated – a 50/50 split as well as *k*-folds cross-validation. It should be noted that when evaluating both models, the parameter 'shuffle' was set to *true*. This is required to get a balanced distribution of data between the training set(s) and testing set(s). The reason for this is that the mice protein expression data set is organised by class. Shuffling the data randomises it so that the data has no order. If the data was not shuffled, there would be a large (perhaps even pure) concentration of one class in the training set with the other classes excluded to the testing set, hence giving an extremely low accuracy as it may be able to classify half the observations very well (in a 50/50 split), but completely fail at classifying the rest.

3. Results

The results are showcased in 3 distinct sections:

- 3.1 Descriptive Statistics
- 3.2 KNN Classification
- 3.3 Decision Tree Classification

3.1 Descriptive Statistics

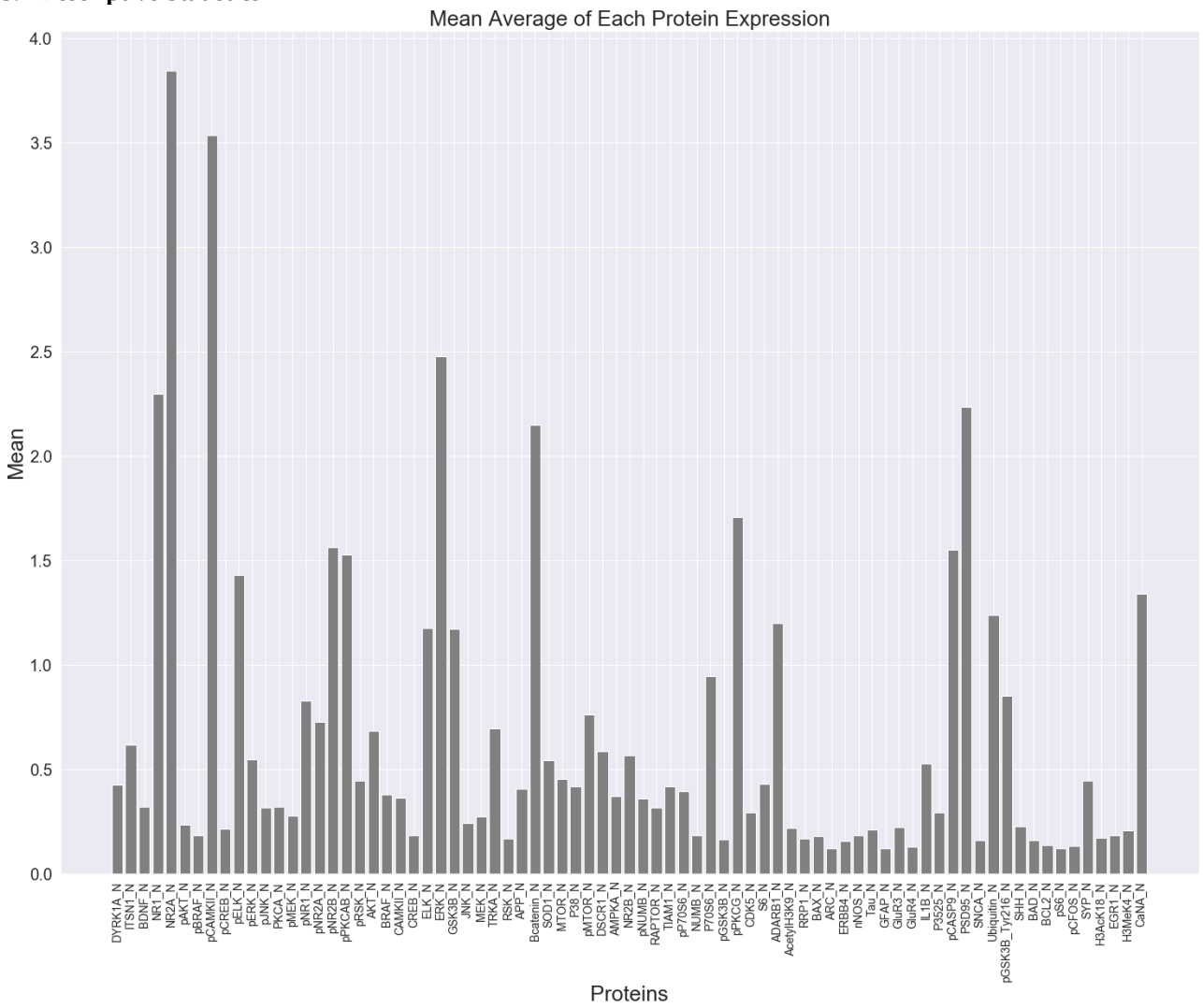


Figure 3 – Mean Average of each Protein Level

Figure 4 describes the five-number summary of all 8 classes of mice with respect to the NR2A protein (*hypothesis #1*). Figure 5 shows the relationship between the memantine and saline treatments with respect to the pCREB and BRAF proteins (*hypothesis #2*).

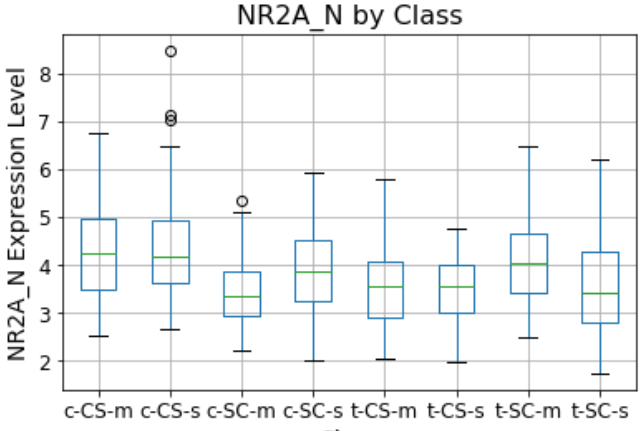


Figure 4 – Box Plots Across All Classes for Protein NR2A

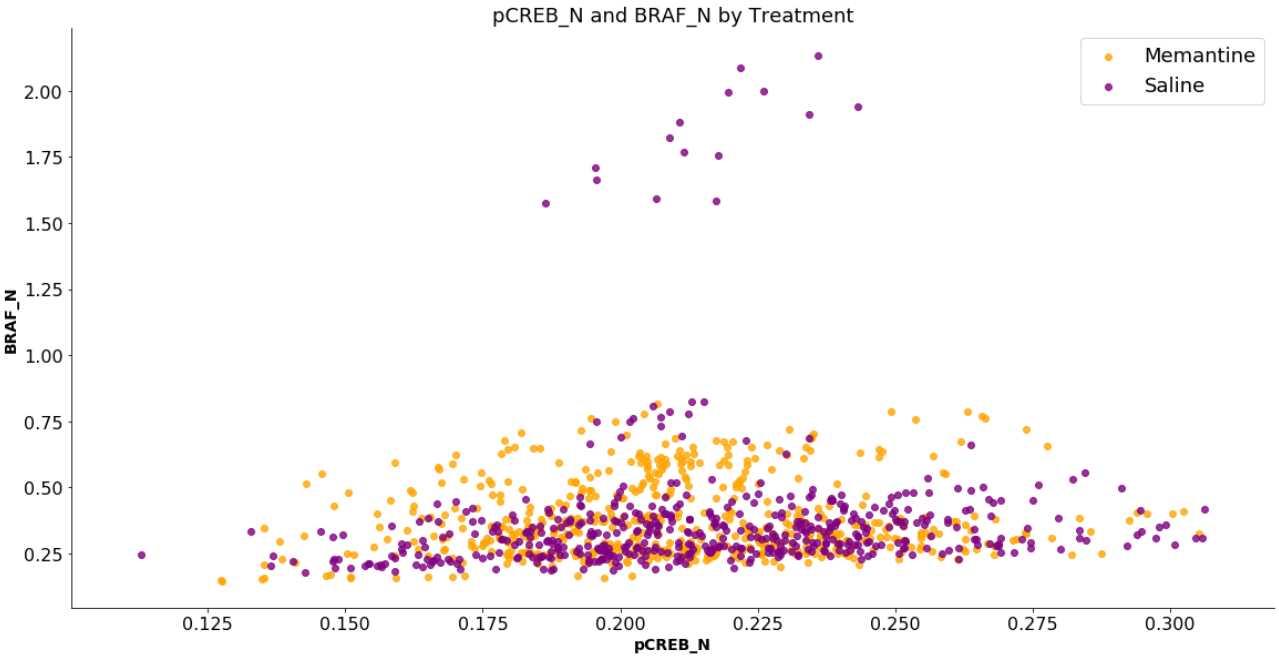


Figure 5 – Scatterplot, Memantine vs Saline, by pCREB and BRAF

The figures below (Figures 5 and 6) represent the effects of stimulation via control-shock on different protein levels in 4 different proteins: pCAMKII, CaNA, ITSN1 and SOD1 (*hypothesis #3*).

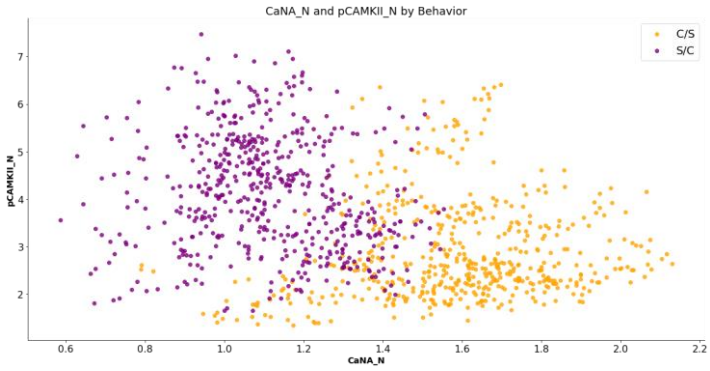


Figure 6 – Scatterplot, Control-Shock vs Shock-Control, by pCAMKII and CaNA

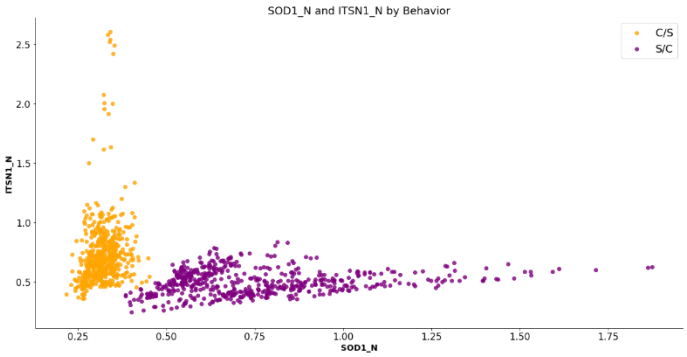


Figure 7 – Scatterplot, Control-Shock vs Shock-Control, by ITSN1 and SOD1

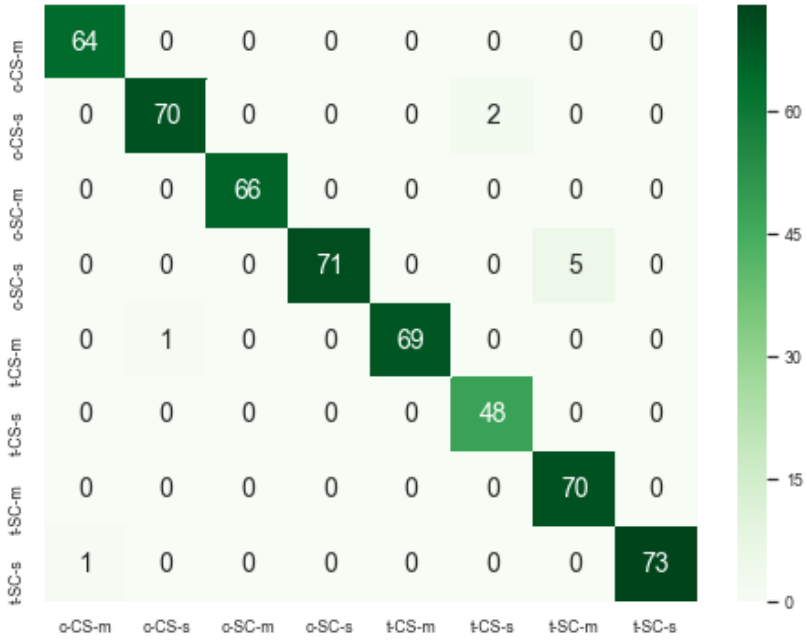
3.2 KNN Classification

The hill climbing using a vanilla KNN Classifier produced 50 proteins as features. Those proteins selected were (in no particular order):

Bcatenin_N, NUMB_N, JNK_N, EGR1_N, GluR3_N, GluR4_N, P3525_N, RRP1_N, pERK_N, pCFOS_N, pPKCG_N, pNR2A_N, pGSK3B_N, AKT_N, CaNA_N, pS6_N, PSD95_N, H3MeK4_N, pCASP9_N, pMTOR_N, NR2B_N, nNOS_N, CAMKII_N, BAD_N, Tau_N, CDK5_N, pCREB_N, DYRK1A_N, BCL2_N, S6_N, H3AcK18_N, BAX_N, pMEK_N, CREB_N, SOD1_N, APP_N, ADARB1_N, TIAM1_N, ERBB4_N, RSK_N, pP70S6_N, IL1B_N, BRAF_N, pBRAF_N, ITSN1_N, SNCA_N, pAKT_N, pELK_N, SYP_N and PKCA_N.

Training the data on the above features in a 50/50 train/test split resulted in an accuracy of **0.98333** for KNN classification.

A confusion matrix was generated to showcase the predictions by the KNN Classifier and the matrix output is shown in Figure 8 in the form of a heatmap.



. Figure 8 - Confusion Matrix for k-Nearest Neighbour Classification Model

The resulting classification report details the precision, recall, f1-score and support values for the model. The report is as follows:

	precision	recall	f1-score	support
c-CS-m	0.98	1.00	0.99	64
c-CS-s	0.99	0.97	0.98	72
c-SC-m	1.00	1.00	1.00	66
c-SC-s	1.00	0.93	0.97	76
t-CS-m	1.00	0.99	0.99	70
t-CS-s	0.96	1.00	0.98	48
t-SC-m	0.93	1.00	0.97	70
t-SC-s	1.00	0.99	0.99	74
accuracy			0.98	540
macro avg	0.98	0.98	0.98	540
weighted avg	0.98	0.98	0.98	540

After *k*-folds cross-validation, the following accuracy scores were reached for each of the 8 folds:

```
[Fold 0] score: 1.0000
[Fold 1] score: 0.9926
[Fold 2] score: 1.0000
[Fold 3] score: 1.0000
[Fold 4] score: 0.9926
[Fold 5] score: 1.0000
[Fold 6] score: 1.0000
[Fold 7] score: 0.9926
```

3.3 Decision Tree Classification

The hill climbing using a vanilla Decision Tree Classifier produced 18 proteins as features. Those proteins selected were (in no particular order):

Bcatenin_N, NUMB_N, JNK_N, EGR1_N, GluR3_N, GluR4_N, P3525_N, RRP1_N, pERK_N, ARC_N, P38_N, pPKCG_N, pNR2A_N, AKT_N, CaNA_N, pCREB_N, pPKCAB_N and CREB_N.

Training the data on the above features in a 50/50 train/test split resulted in an accuracy of **0.73148** for Decision Tree classification.

A visualisation of the Decision Tree on the above features is observed in Figure 9.

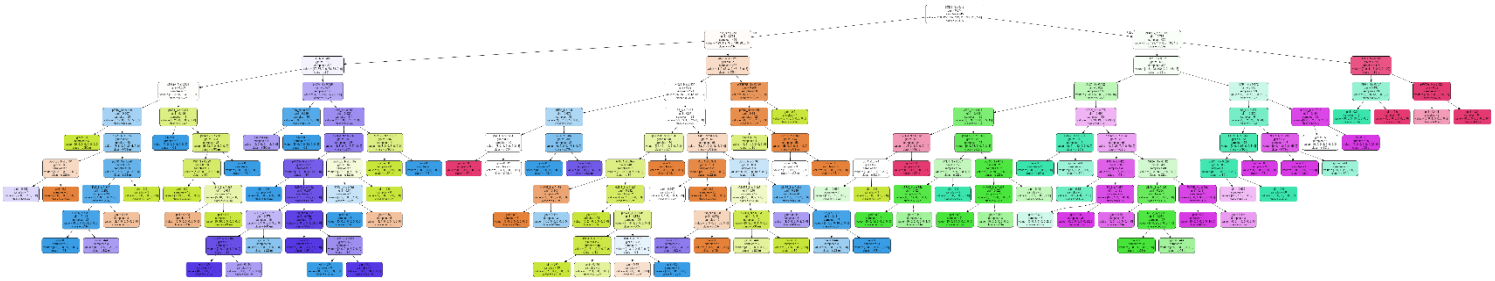


Figure 9 - Decision Tree Visualisation Using 18 Selected Features/Proteins

For improved readability, a larger image can be found here: https://imgur.com/zWT5okM

A confusion matrix was generated to showcase the predictions by the Decision Tree Classifier and the matrix output is shown in Figure 10 in the form of a heatmap.

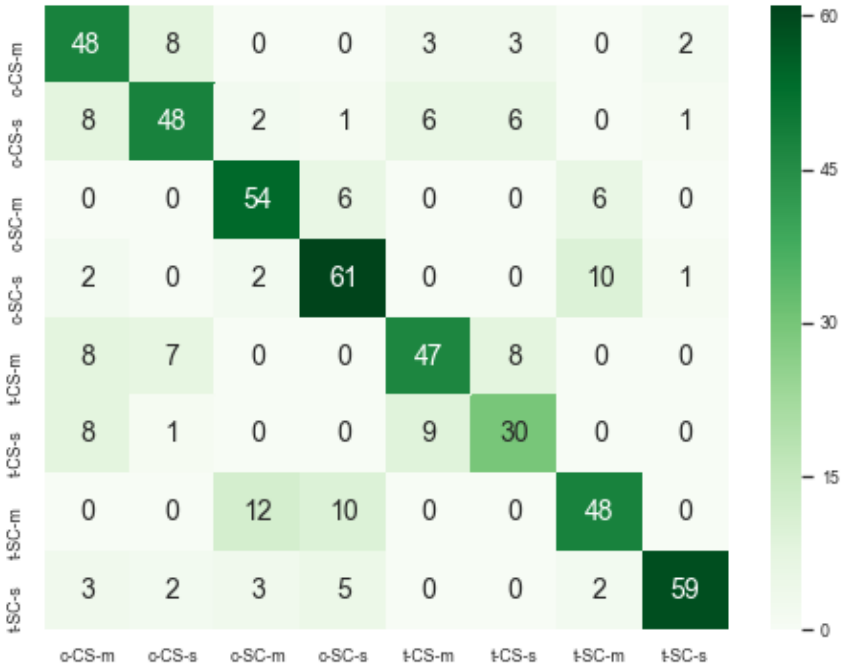


Figure 10 - Confusion Matrix for Decision Tree Classification Model

A classification report details the precision, recall, f1-score and support values for the Decision Tree. The report is as follows:

	precision	recall	f1-score	support
c-CS-m	0.62	0.75	0.68	64
c-CS-s	0.73	0.67	0.70	72
c-SC-m	0.74	0.82	0.78	66
c-SC-s	0.73	0.80	0.77	76
t-CS-m	0.72	0.67	0.70	70
t-CS-s	0.64	0.62	0.63	48
t-SC-m	0.73	0.69	0.71	70
t-SC-s	0.94	0.80	0.86	74
accuracy			0.73	540
macro avg	0.73	0.73	0.73	540
weighted avg	0.74	0.73	0.73	540

A *k*-folds cross-validation on the Decision Tree produced the following scores across 8 different folds:

```
[fold 0] score: 0.7778
[fold 1] score: 0.7630
[fold 2] score: 0.8000
[fold 3] score: 0.7852
[fold 4] score: 0.7852
[fold 5] score: 0.7852
[fold 6] score: 0.7704
[fold 7] score: 0.8296
```

4. Discussion

4.1 Descriptive Statistics

The first hypothesis states that there are some classes of mice that exhibit higher levels of a particular protein than others. One such protein, NR2A, was studied across all eight classes of mice. A box plot was drawn for each class for the five-number summary of the NR2A protein, as shown in Figure 4. It appears that control mice who have been stimulated to learn returned a higher mean expression level for NR2A, with the treatment type not playing a large role here. In this case, the stimulated control mice that have been injected with saline (c-SC-s) had shorter whiskers but 3 outliers whereas the non-stimulated control mice that have been injected with memantine have slightly larger whiskers but no outliers.

It is also noted that the smallest mean values for NR2A were found in stimulated trisomic mice (t-CS-s, t-CS-m) as well as non-stimulated control mice injected with memantine (c-SC-m). The initial hypothesis evaluates to be somewhat true, but there is not a strong case for this particular protein.

The second hypothesis argues that treatment had a role to play in the protein expression levels across all test mice. Figure 5 illustrates the relationship between mice injected with memantine and mice injected with saline in the protein levels of pCREB and BRAF via a scatterplot. This scatterplot demonstrates a high amount of support for the hypothesis as there is a clear distinction between the purple points (saline-injected mice) and yellow points (memantine-injected mice). The difference in the protein levels in the pCREB protein appears to be fairly uniform across both treatment-types. It is the BRAF protein, however, that returns higher values in memantine-injected mice than in saline-injected mice. It is also worth noting the there are two groups of outliers, one at the top of the graph and the other just at the top-end of the yellow points. It's also interesting to note that these points are also saline-injected mice.

The third hypothesis mentioned in this study says that whether or not a mouse has been stimulated to learn has a large effect on the protein expression levels of the mouse. Figure 6 represents the relationship between mice who have been stimulated to learn (C/S) and those that have not (S/C) in regard to proteins CaNA and pCAMKII. This scatterplot has very strong support for the hypothesis in that stimulated mice have significantly higher levels of CaNA and slightly lower levels of pCAMKII than non-stimulated mice. There is a clear separation in the plot between stimulated mice and non-stimulated mice with only a moderate amount of overlap between the two groups.

Figure 7 shows even stronger support for the third hypothesis than Figure 6. Figure 7 shows the relationship between stimulated and non-stimulated mice in a scatterplot like Figure 6, only now it plots a different protein pair; SOD1 and ITSN1. Looking at this graph, it is easy to tell immediately there is a significant contrast between the two behaviour groups. Stimulated mice return a fairly concentrated range of values for SOD1, lying between ~0.2 and ~0.4. The ITSN1 values for stimulated mice also generally fall between ~0.3 to ~1.2, but there are some outliers than stretch to ~2.6.

On the other hand, the non-stimulated mice had a range of values for these proteins that did not have a large range for ITSN1, but stretched from ~0.3 to ~1.8, with most of the points concentrated between ~0.3 and ~1.1.

This pattern is almost mirror-like between the two groups and the shows how dissimilar stimulated and non-stimulated mice are with respect to proteins SOD1 and ITSN1.

4.2 Data Modelling

This part of the study compares the two modelling techniques employed throughout the research of the data set. The results from both the *k*-nearest neighbour classifier and decision tree classifier will be evaluated and compared in terms of their ability to classify mice based on proteins selected by the hill climbing technique. This will be done as a comparison on the general accuracy of each model. Finally, a recommendation will be given for this this type of data on which classifier is most suitable.

Starting with the *k*-nearest neighbour classifier, the hill climbing algorithm utilised a vanilla KNN classifier for selecting features, as not to give any bias to feature selection. Interestingly, 50 proteins of the 77 in total were selected as features for the KNN model which is over twice as many as the mere 18 selected by the hill climbing used with the decision tree classifier.

Between both sets of selected features, the intersecting proteins were: Bcatenin_N, NUMB_N, JNK_N, EGR1_N, GluR 3_N, GluR4_N, P3525_N, RRP1_N, pERK_N, ARC_N, P38_N, pPKCG_N, pNR2A_N, AKT_N, CaNA_N, pCREB_ N, pPKCAB_N and CREB_N. These 15 proteins were common between both sets of features, leaving only 3 proteins in the decision tree hill climbing: ARC_N, P38_N and pPKCAB_N.

It struck me as strange that the decision tree classifier did not select the SOD1 and ITSN1 proteins as we saw in data ex ploration that these two proteins were very determinant of whether or not a mouse was stimulated or non-stimulated. A possible explanation is that the hill climb found an alternative peak to the one that would have given these two proteins as features. As a result, it might have performed better or worse, depending on how discriminant the selected features ar e between classes. It is also necessary to keep in mind that the C/S or S/C component is just one third of the class type.

For both classifiers, a 50/50 split was used for training and testing. This was chosen for two reasons. The first is to keep the split the same for each classifier as to give a fair evaluation of each model for comparison purposes. The second reas on is that the dataset is not very large, so not having enough training data results in a low support score for the models a nd can give misleading results. A large training set can also train the model too closely to the data, which results in an o verfitted model. A 50/50 split seemed to be the best compromise here.

The final overall KNN accuracy across 50 features over a 50/50 train/test split was 0.98333. This was achieved using a k value of 3, a p value of 1 and weighting points by distance. A uniform weight seemed to only decrease the accuracy of the classifier and it performed better when it evaluated points based on the inverse of the distance of each point to the n ew data point. A k-value of 3 was chosen as it seemed to perform best at this value - increasing the value only decreased the classifier's ability to accurately determine mouse-class. This is the same for the 'p' value of the classifier – the Man hattan distance performed best (a 'p' value of 1) and the classifier's ability only decreased as the metric changed to Eucl idean (a 'p' value of 2) and so on.

The resulting accuracy of the decision tree across 18 features in a 50/50 train/test split was 0.73148. As the gini index is faster than entropy but results in marginal accuracy differences, it was used for this classifier. This accuracy was achiev ed using no limit on max features as the 18 features was already quite a small set of the possible 77, so limiting the tree any further resulted in a lower accuracy. A minimum samples per leaf of 3 and a minimum sample size to split of 8 had the best performance for accuracy, where adjusting the values any further decreased the accuracy of the model.

Looking at the confusion matrix of each classifier, it further confirms that the KNN classifier was superior. For the KN N confusion matrix illustrated in Figure 8, we see that across all classes it was mostly accurate. The biggest blunder occ urred at the c-SC-s class where this class was predicted accurately 71 times, but incorrectly 5 times where the model ins tead predicted t-SC-m. This is very interesting as c-SC-s and t-SC-m are different across 2 out of the 3 features. This mi ght imply that whether or not a mouse was stimulated had a lot of power in determining the class which could explain th e misprediction of the KNN classifier.

Figure 10 represents a heatmap of the confusion matrix for the decision tree classifier. This classifier performed poorly across all classes. A very notable aspect of this matrix is that the classifier predicted the t-SC-m class correctly only 48 t imes out of 70. There were only two other classes which were predicted for t-SC-m mice and they were c-SC-m and c-S C-s. Similar to the KNN classifier, it appears that the SC/CS feature is common between mice predicted correctly and th ose predicted incorrectly, suggesting that it possess a higher weight in predictive power.

Unsurprisingly, the k-folds cross validation scores returned higher average accuracy scores for each classifier, with the KNN classifier reaching a sort of ceiling and capping at 100% for some folds and the decision tree classifier reaching as high as 0.8296 for one of the folds but averaging about a ~0.78 accuracy, up from the ~0.73 from the 50/50 train/test sp lit.

4.3 Recommendation

The results speak for themselves - the KNN classifier is much more superior at determining the class of mice based on p rotein expression levels. Why might this be the case, though? Decision trees are prone to being overfitted but due to the low accuracy of the decision tree classifier on this dataset, that is clearly not happening here. With a 0.73% accuracy, th e classifier is not suffering from doing its job too well – it is not doing it well enough.

The reason for this could be that the decision tree does not handle the continuous nature of the data from the protein exp ression levels well enough. Referring to Figure 9, it can be seen that each protein is split into categories to create the dif ferent branches of the tree. Each data point in the protein levels is unique and valuable – but what the tree is doing is bre aking those values down into categories (is *y* protein less than or greater than *x* value) and splitting the observations bas ed on what category it falls into. This causes the tree to lose information that is otherwise very valuable. On the other ha nd, the KNN classifier does not waste such information – every point has some fixed distance on some vector space and is evaluated contextually with each observation it attempts to classify. This is likely why the KNN classifier outperform s the decision tree.

For future data sets with protein expression levels, it is not recommended that the decision tree is used for such data and that either the KNN classifier or an alternative is used.

5. Conclusion

It was found that certain proteins can be determinant of the various categories that make up the class of each mouse. So me proteins provided minimal insight into the characteristics of each mouse while some were found to be extremely dis criminant. Classifying mice in the context of protein expression levels in the nuclear cortex can be done with great accu racy, however, it is important that right tools are used to do this. Because of the continuous nature of the data, one might refrain from using a classifier like the decision tree where the details of the information are lost. The KNN classifier is a much more effective algorithm for determining the class of any given mouse and is the suggested method going forwa rd for this domain of data.

6. References

[1] Archive.ics.uci.edu. 2020. *UCI Machine Learning Repository: Mice Protein Expression Data Set*. [online] Available at: <https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression>

[2] R), I. and Srivastava, T., 2020. *K Nearest Neighbor | KNN Algorithm | KNN In Python & R*. [online] Analyti cs Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>

[3] Scikit-learn.org. 2020. *Sklearn.Neighbors.Kneighborsclassifier — Scikit-Learn 0.23.1 Documentation*. [onli ne] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.ht ml>

[4] Medium. 2020. *Machine Learning: Decision Tree Classifier*. [online] Available at: <https://medium.com/ma chine-learning-bites/machine-learning-decision-tree-classifier-9eb67cad263e>

[5] plot, m., Pakki, N. and Sletfjerding, M., 2020. *Matplotlib/Seaborn: First And Last Row Cut In Half Of Heat map Plot*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/56942670/matplotlib-se aborn-first-and-last-row-cut-in-half-of-heatmap-plot> [Accessed 10 June 2020].

[6] Yongli Ren, 2020. Lecture Notes and Tutorial Material, Practical Data Science (COSC2670), RMIT Univer sity.