Student ID: s3796258

Student Name: Syed Haider Saleem

Title:

Mice Protein Expression Data Set

Statement:

Classification of mice protein expression dataset based on the expression level of different proteins. The aim is to identify subsets of proteins that are discriminant between the classes.

Author Information:

Syed Haider Saleem (s3796258)

Contact Details:

0450518465

Date of Report:

6/10/2020

Table of Contents

Abstract

Expression levels of 77 proteins measured in the cerebral cortex of 8 classes of control and Down syndrome mice exposed to context fear conditioning, a task used to assess associative learning. The aim of this project is to identify subsets of proteins that are discriminant between the classes using classification techniques. For this purpose, we have applied two machine learning techniques that includes K-nearest neighbours and decision tree. To resolve the issue of feature redundancy we are going to apply hill climbing tree algorithm and extract the best features. Various exploration techniques are also implied to test and prove some interesting hypothesis using violin plots etc. For each machine learning technique ("KNN", "Decision Tree") the methodology taken, pre-processing done in each case along with their results are described in detail. Both methods are compared using some common evaluation metrics which we decided in the start based on our dataset and problem; accuracy, F1 score and confusion metrics are used for this purpose. The results demonstrate that in general both models performed performed well and gave promising results but kNN outperformed the decision tree with a high accuracy of 99% and an F1 score of 99.

Introduction

Different mice are kept in different environments to find out how different types of proteins expresses in each case. This is important in understanding our different conditions impact individual mice. The experiments were carried out on 38 control mice and 34 trisomic mice (Down syndrome), for a total of 72 mice. In the experiments, 15 measurements were registered of each protein per sample/mouse. Therefore, for control mice, there are 38x15, or 570 measurements, and for trisomic mice, there are 34x15, or 510 measurements. The dataset contains a total of 1080 measurements per protein. Each measurement can be considered as an independent sample/mouse.

What we aim to achieve in this project is to use the above-mentioned dataset and try to apply machine learning models to classify different classes based on the expression of different proteins. For this purpose, we have applied two machine learning techniques including KNN and Decision trees and results for each method is compared.

Feature selection was an integral part in solving our problem as we don't want to have unnecessary features and complex our model, this was done using hill climbing algorithm; this was done for both cases separately. Another important part was to use different visualisation techniques like violin plots, correlation metrics to understand the relationship better and test our hypothesis.

Evaluation was done using accuracy, F1 score and confusion metrics. KNN out formed decision tree with a 99% accuracy corresponding to 82% accuracy.

Methodology

Data Preparation

Data Retrieving

The dataset was taken from Machine Learning Repository https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression which is named as Mice Protein expression dataset. From here we downloaded an excel file which was then converted into a csv file to be later loaded onto the notebook for our convenience.

Load Data

The csv converted file Data_Cortex_Nuclear.csv was then loaded into the notebook using pandas built in function of pd.readcsv( ).

Inspect Data

The dataset contains a total of 1080 measurements. It has 82 features where 77 among them are different types of proteins which are our prime focus along with MouseID, Genotype, Treatment type, behaviour and Class. The classes are somewhat imbalance, but the difference is not that huge that we should be worried about. Also, our data also contains outliers and missing values.

Handling Missing Values

The biggest challenge in this dataset is for sure handling missing values, as almost all the input features ("proteins") has missing values hence dropping rows or columns with missing value is not an option for us as it will greatly reduce our dataset. For dealing with this we replaced the missing value in each particular column with the mean of that column corresponding to its class. The last part (corresponding to its class) is important in my opinion as it helps in maintaining the integrity of data by replacing the mean with its closest neighbour value.

Outliers

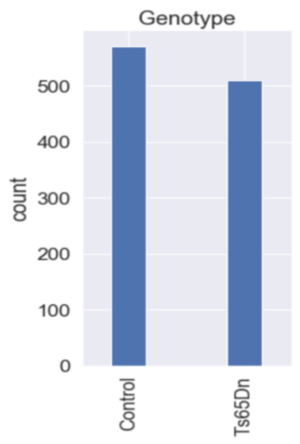
We have also detected a lot of outliers in our dataset but we choose not to remove or replace them as we believe that the particular value represents some real life representation and if we remove that we will create a bias. Plus the dataset is obviously carefully collected and the measurements are assumed to be accurate.

Data Exploration

Exploring Each Column

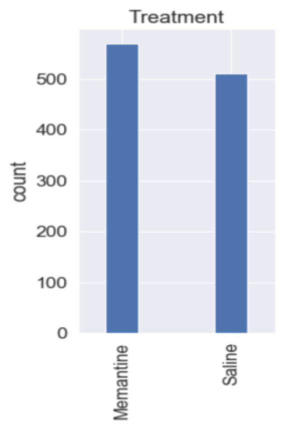
Since we have 84 columns we have chosen some of the most important features based on hill climbing algorithm. We will start by exploring our environment attributes that includes Genotypes, Treatments, Behaviour and Class. Since most of these attributes have binary categorical values hence its best to plot the count of each type of variable in that particular attribute.

```
Control    570
Ts65Dn     510
Name: Genotype, dtype: int64
```



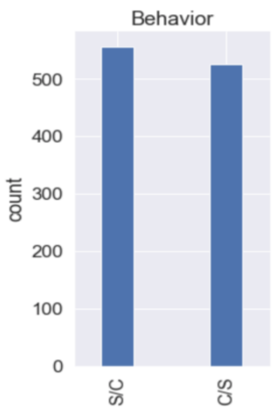
Genotype

```
Memantine    570
Saline       510
Name: Treatment, dtype: int64
Text(0, 0.5, 'count')
```



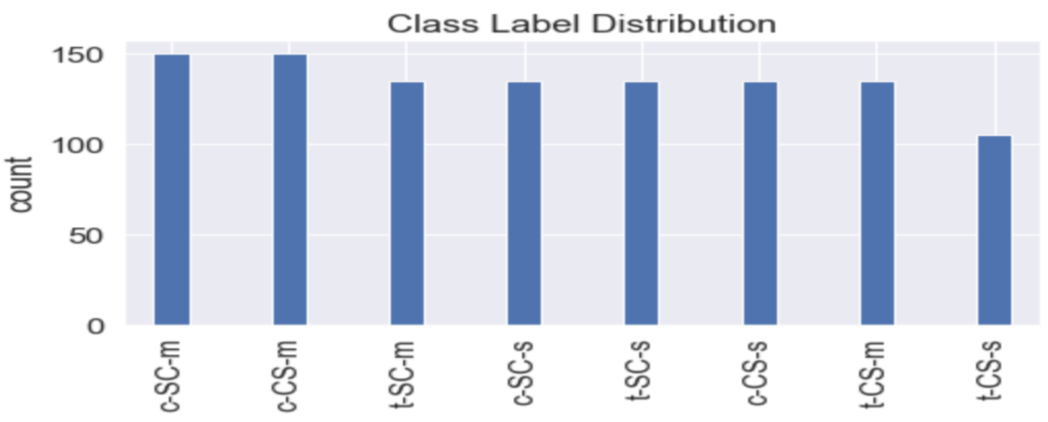
Treatment

```
S/C    555
C/S    525
Name: Behavior, dtype: int64
Text(0, 0.5, 'count')
```



Behaviour

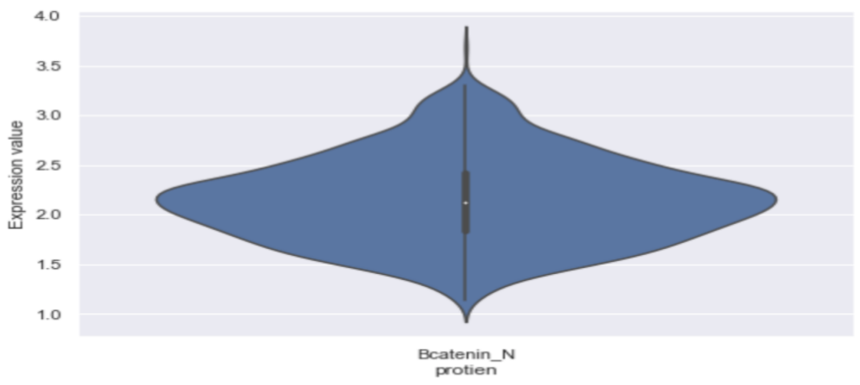
Class Label Distribution representing the minor imbalance in our label class

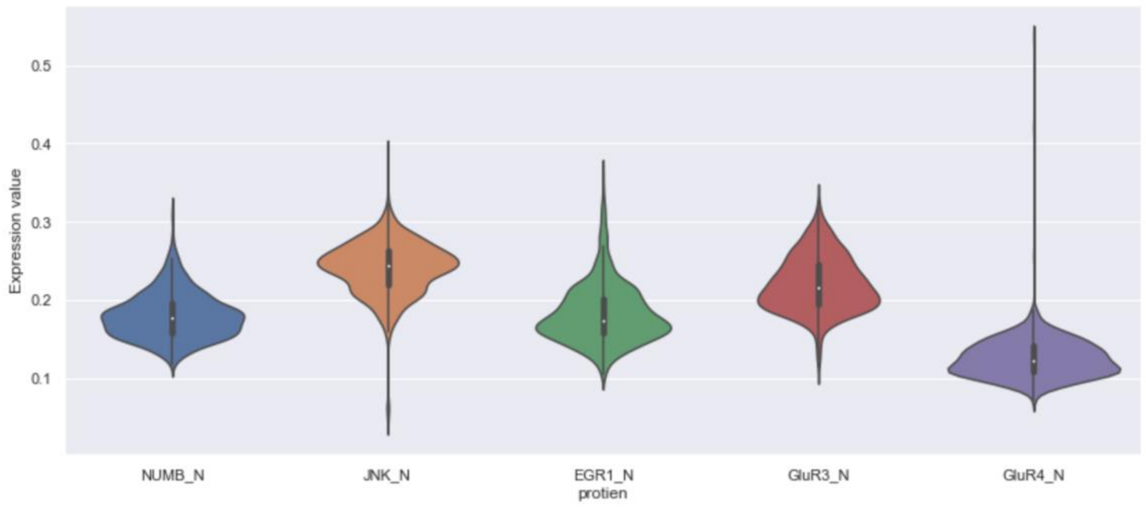


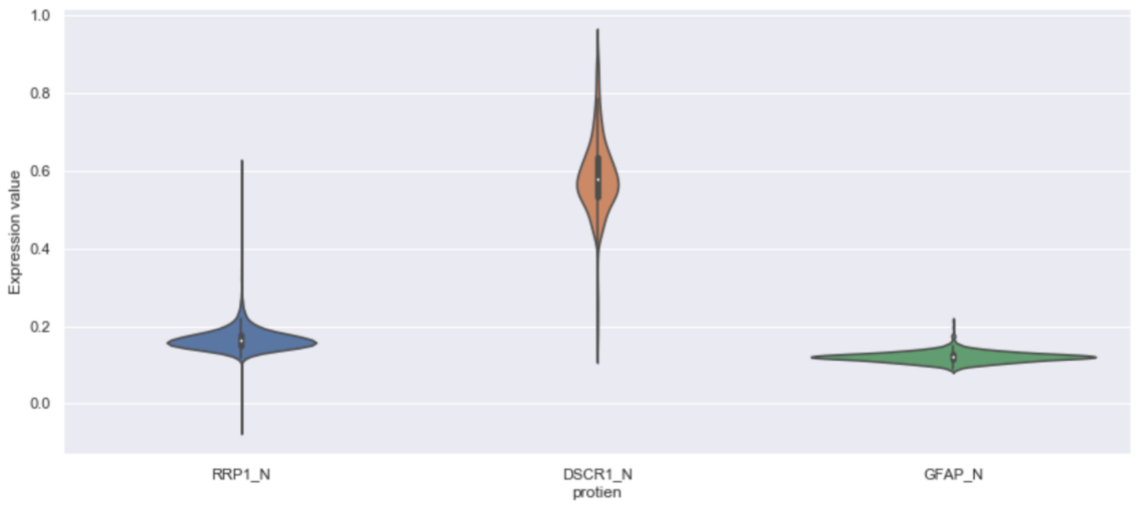
The features (attributes) explored above makes up our output classes (as combination of these attributes are the label class which is “class”). Now we will move on to exploring the input features which are 77 proteins and exploring these are our major concern as the main goal of this project is to identify subsets of proteins that are discriminant among the classes.

Exploring Proteins

We will try to explore some of the most important proteins that contribute in determining the class, for this purpose we will be using violin plots as they are an amazing exploratory tool to see the distribution of our numeric data. We could have used boxplot as well, but violin plot exhibits more information in this case. Violin plots are similar to box plots, except that they also show the probability density of the data at different values usually smoothed by a kernel density estimator. Top 9 protein violin plots are displayed below.





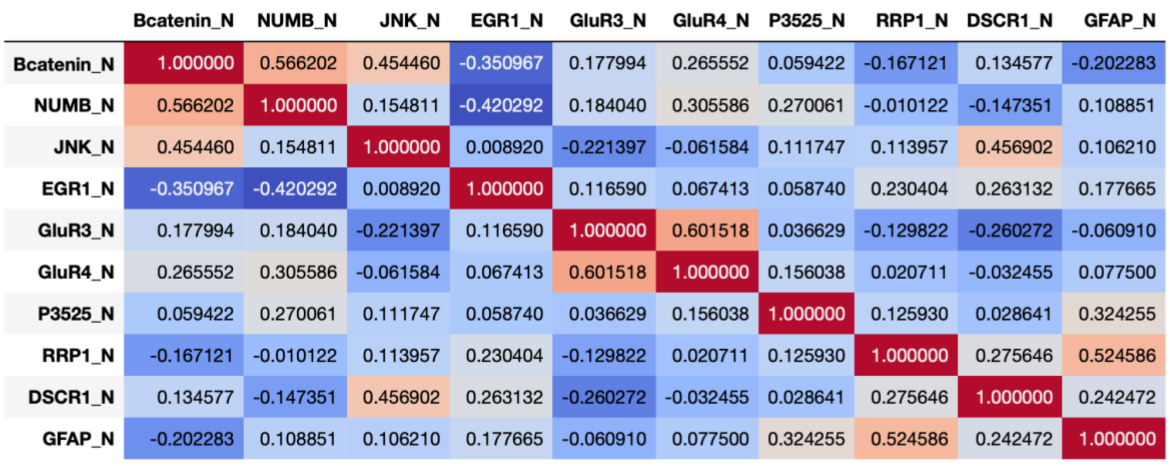


Exploring Pairwise Relationships

We will start by finding the correlation among all the proteins to have a sense of idea how closely related the behaviour of each protein is and whether changes in the expression level of one protein could cause a change in other proteins as well.

Hypothesis: Change in expression level of one protein is independent of other proteins

To prove the hypothesis, we will plot a correlation matrix for our top 10 identified proteins.



	Bcatenin_N	NUMB_N	JNK_N	EGR1_N	GluR3_N	GluR4_N	P3525_N	RRP1_N	DSCR1_N	GFAP_N
Bcatenin_N	1.000000	0.566202	0.454460	-0.350967	0.177994	0.265552	0.059422	-0.167121	0.134577	-0.202283
NUMB_N	0.566202	1.000000	0.154811	-0.420292	0.184040	0.305586	0.270061	-0.010122	-0.147351	0.108851
JNK_N	0.454460	0.154811	1.000000	0.008920	-0.221397	-0.061584	0.111747	0.113957	0.456902	0.106210
EGR1_N	-0.350967	-0.420292	0.008920	1.000000	0.116590	0.067413	0.058740	0.230404	0.263132	0.177665
GluR3_N	0.177994	0.184040	-0.221397	0.116590	1.000000	0.601518	0.036629	-0.129822	-0.260272	-0.060910
GluR4_N	0.265552	0.305586	-0.061584	0.067413	0.601518	1.000000	0.156038	0.020711	-0.032455	0.077500
P3525_N	0.059422	0.270061	0.111747	0.058740	0.036629	0.156038	1.000000	0.125930	0.028641	0.324255
RRP1_N	-0.167121	-0.010122	0.113957	0.230404	-0.129822	0.020711	0.125930	1.000000	0.275646	0.524586
DSCR1_N	0.134577	-0.147351	0.456902	0.263132	-0.260272	-0.032455	0.028641	0.275646	1.000000	0.242472
GFAP_N	-0.202283	0.108851	0.106210	0.177665	-0.060910	0.077500	0.324255	0.524586	0.242472	1.000000

There seems to be some positive relationship between (Bcatenin_N, NUMB_B), (Bcatenin_N, JNK_N), (Bcatenin_N, JNK_N), (JNK_N, DSCR1_N), (GluR3_N, GluR4_N), (RRP1_N, GFAP_N) which basically means that with the increase in expression level of one protein the other increases as well. The strength of red band shows the amount of positive relation.

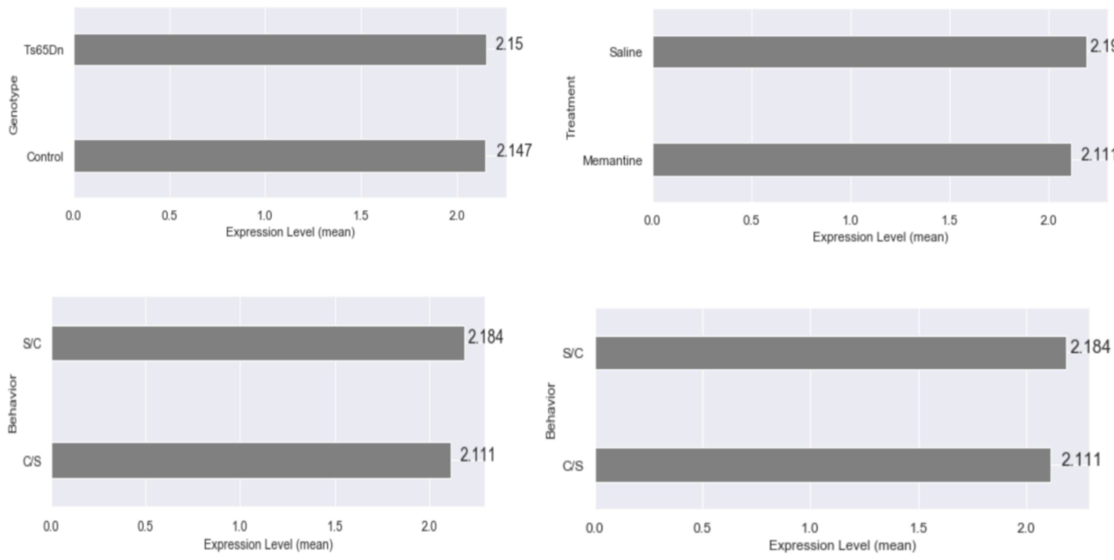
Negative relation can be found in case of (EGR1_N, Bcatenin_N) and (EGR1_N, NUMB_N) where with the increase in expression level of one protein the other one decreases. The strength of blue band depicts amount of negative relation.

While in most cases there seems to be no relation which does make sense.

To get more in depth analysis of pairwise relationships between different columns we will be talking individual proteins. We will start with the most important protein in term of expression level which is **Bcatenin_N**.

Hypothesis: Does expression level of protein "Bcatenin_N" varies significantly with any specific attribute individually ("genotype", "behaviour", "treatment")

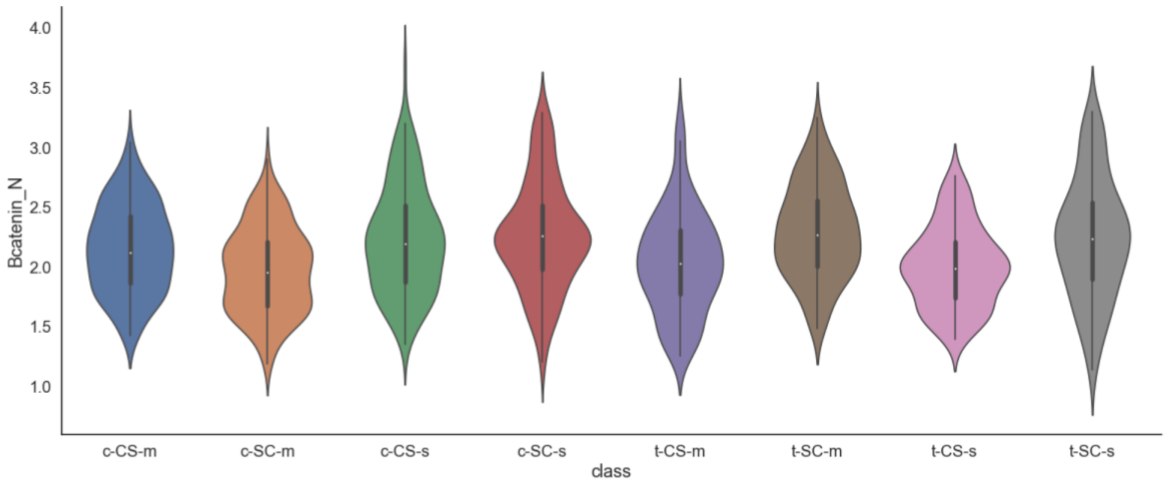
This hypothesis will help us in understanding the individual effects of each environment variables like genotype, behaviour and treatment and can provide good insight that whether there is any specific environment that causes a drastic change in expression levels of protein.



From the above two bar charts we can clearly see that the individual environment variable is not having any significant effect on the mean expression level of this particular protein.

Hypothesis: will the combination of these attributes ("genotype", "Treatment", "behaviour") which is the class have an impact on expression of protein "Bcatenin_N"

Since we have proved that individually these attributes do not have any significant impact, less take them together using the class column and determine the effect on expression level. For this purpose, we have created Violin plots of expression level values across each class.

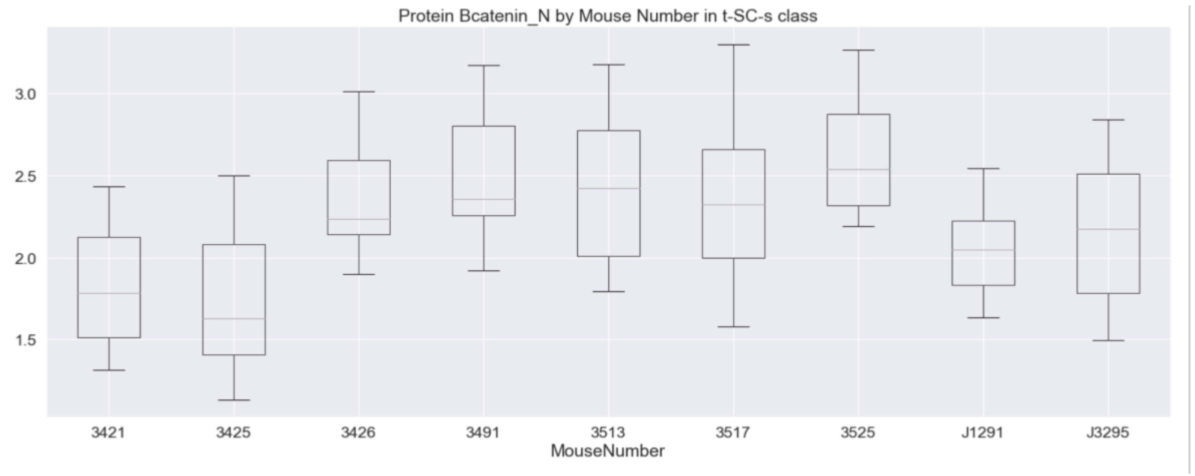


Overall the expression level across each class seems to be consistent for the most part and does not vary drastically across any particular class but there is a notable change in mean expression values in case of few classes and we can clearly see the change. The most notable change is between class t-CS-s and t-SC-s, while few others stands out as well. In the light of above exploration, we can therefore say that expression values of proteins do changes with the class type (which in our case is a combination of variables).

To get more insight let’s take a particular class and dig down a bit more in depth. For this purpose, we will be taking the class "t-SC-s" as it seems to contain diverse expression values and the minimum expression level as well.

Hypothesis: Does each mouse responds differently to protein "Bcatenin_N"

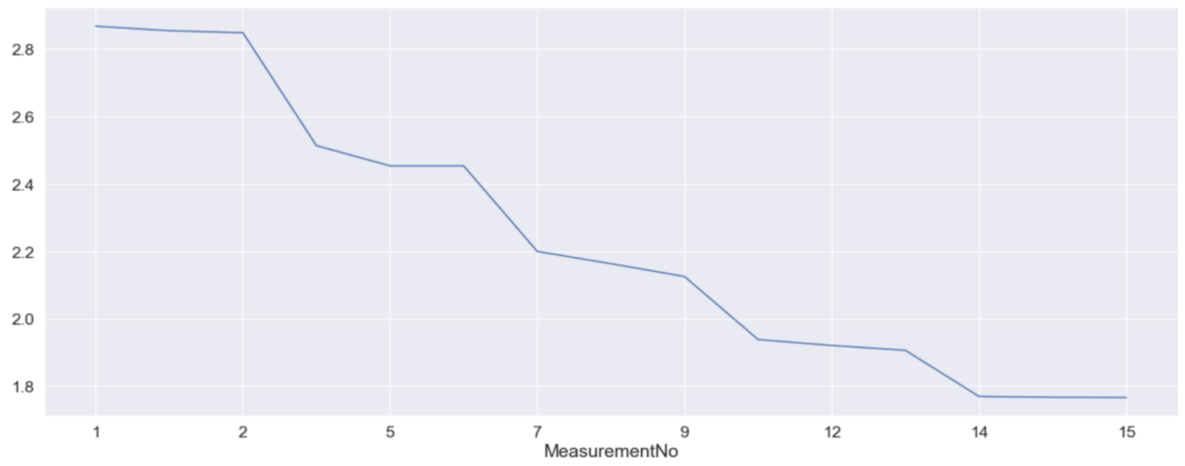
For this particular class type (t-SC-s) and protein (“Bcatenin_N) we will be focusing on individual mice and see if the expression level changes with respect to a particular mouse.



This shows that yes, the expression level does vary with respect to each mouse even in the same class (Environment).

Hypothesis: Does expression value also varies with each measurement in the same mice

Now that we have established that each mouse can respond differently let’s look into the fact that whether measurement number (experiment number) also have an impact



Surprisingly there is an inverse relation in expression level and measurement no, as with every next measurement the expression level of the mice decreases, at least for our particular class and protein.

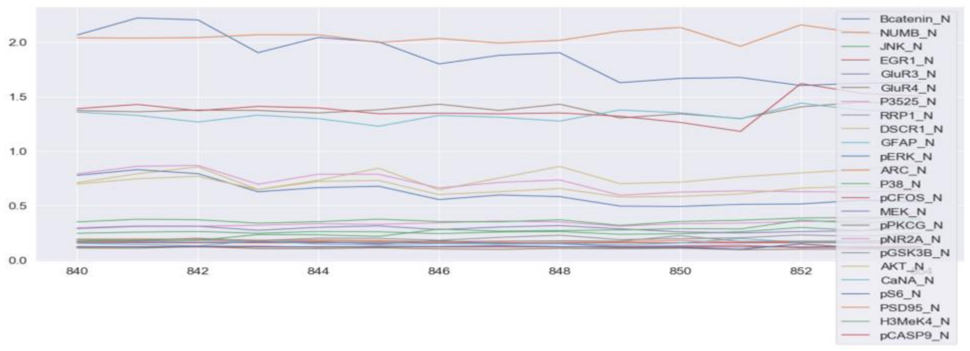
Results and Takeaways from the analysis of protein Bcatenin_N

- Attributes like genotype, behaviour and treatment does not have a significant impact on expression level if taken individually
- Combination of these attributes which is a class (specific environment) does have some impact on expression level of protein
- Individual mouse does respond differently to this protein (at least for our specific case)
- Each next Measurement(mean) shows a decrease in expression value (at least for our specific case)

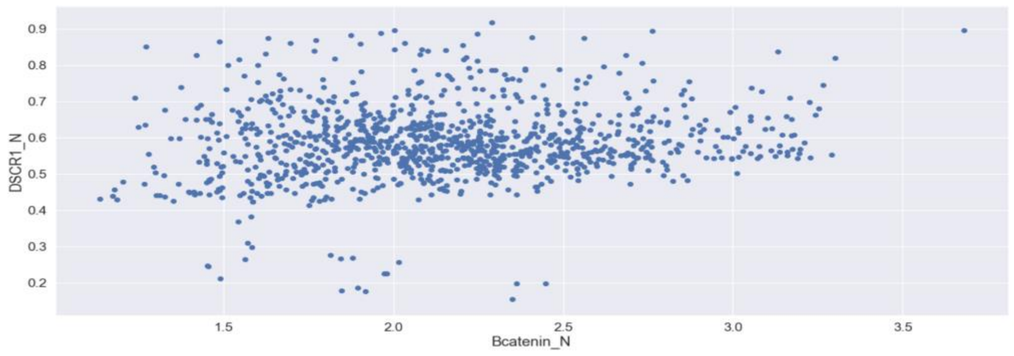
To verify the results we have established from this protein we also did the same exploration on "DSCR1_N" and we found out that almost all the results remained consistent for this protein as well hence we can assume that these results will remain valid for the remaining proteins as well.

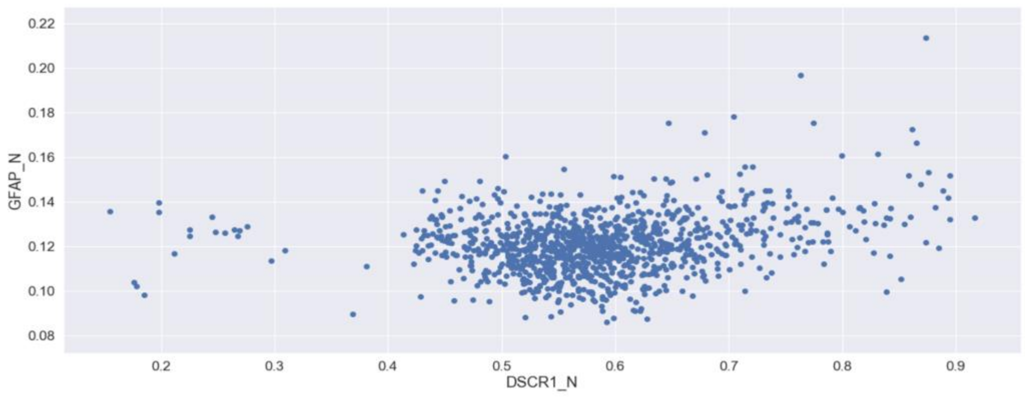
Hypothesis: Does each protein have same expression level in a single specific mouse

From the analysis of protein GFAP_N we have concluded that different protein have different expression level even in the same mouse.



Scatter plots are also used to visualise the relationship among proteins





Data Modelling

For the purpose of classification, we will be using two machine learning models namely K-nearest neighbours and Decision Trees. We will define a common evaluation metrics and try to fit and optimize the model in each case and present our findings and judgement.

Evaluation Metrics

Defining a right evaluation metrics is important as it gives us feedback on our built model and helps us in improving the performance of the model by tuning it or picking the right model. We will be talking accuracy as our main evaluation metrics along with F1 score to ensure that our results are consistent also taking slight imbalance in class ratio into prospective.

K-Nearest Neighbours

K-Nearest Neighbors is a machine learning technique that determines the label of unknown (target) class by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). It determines the class label based on proximity (distance) with its neighbors and assume that the new data point lying near the class belongs to that particular class.

Pre-processing

- **Separating Input and output features**

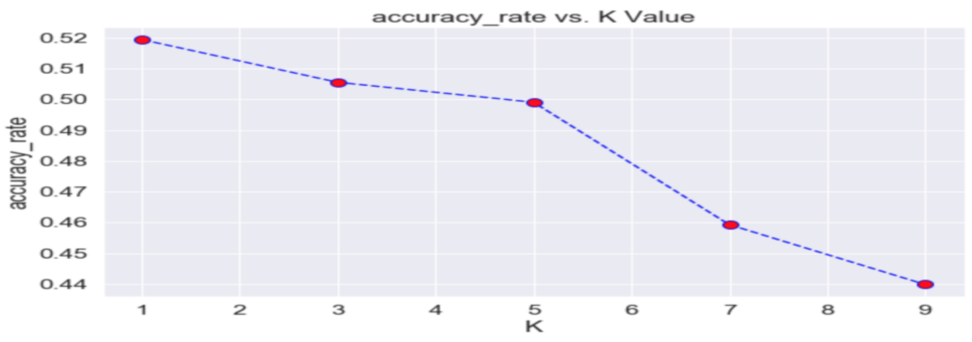
  The first part of pre-processing is separating input features (proteins) from output features (environment variables)

- **Standard Scaler and Normalization**

  K-Nearest Neighbors does require quite a bit of pre-processing as it works on calculating distances hence we first need to make sure we can sending all the numeric variables, then those numeric variables needs to be normalized or scaled so that calculating distances can be achieved conveniently and they are all on the same scale. For this purpose, I have performed both standard scaler as well as normalization on my input features which were proteins.

- **Finding best K**

  K is basically the number of votes (neighbors) on the basis of which we decide from which class does our unknown data point belongs to. This is a hyperparameter and it is probably the most important parameter in KNN hence to accommodate this we ran model with different k's to select the best k value. The K value usually taken is an odd number so that we could have a deciding vote.



We can see that the accuracy is best with the least k value, but we don't want to pick k as 1 as it is unstable and gives judgment based on 1 vote hence we will continue with k as 3.

- **Feature Selection**
  For training the model and reducing dimensionality by removing redundant features we used hill climbing algorithm.

- **Train Test Split**
  Now before passing the data to the model we will split our data into 80% training and 20% testing set as it helps in validating the performance and we could tune/pick our model based on that 20% testing/validating set.

Defining Model and Fitting

After pre-processing the data, we will feed the data to the model initially with the default parameters and having K as 3. Since the data is only about 1000 rows 24 best features (proteins) it didn't take much time in model training.

Predictions and Results

After training the data we predicted the test set which was set aside initially (20%) on our trained model using the evaluation metrics defined in the start. We received an accuracy of about 99% with an F1 score of 0.99 as well.

We don't need to optimize the parameters in case of KNN as we already got a high accuracy of 99%.

Decision Tree

The second model used for classification is decision tree. Decision tree is widely famous due to its robustness to noise, handling missing values, low computational cost and interpretability Decision tree basically uses a new feature at each step to try to reduce the entropy and maximise information gain by placing each new unknown data point in one of the class based on feature. The criterion, max_depth and min_samples_split are some of the key hyperparameters in decision tree which needs to be tuned accordingly.

Pre-processing

It does not require much pre-processing like KNN, but we do need to perform separation of input output features, feature selection again as the important features for this model might be different then KNN and train test split which is the same as mentioned for KNN. Even here we will take 80% training and 20% validation (test) set.

Defining Model and Fitting

We will initially train the decision tree model with the default parameters. Total records are 1080 with 24 features.

Predictions and Results

After training the data we predicted the test set which was set aside initially (20%) on our trained model using the evaluation metrics defined in the start. With each new iteration we generally receive a bit varied result averaging to about 80% to 84% accuracy and 0.80 F1 score.

Hyperparameter Tunning (Grid Search with Cross Validation)

In the case of decision tree, we obviously need to tune our parameters using grid search with cross validation. Cross validation is important as our data is quite limited (less) so we use 5 folds which divides the data into 5 parts and use each part as training and test at least once. After performing grid search our best parameters came out to be **{'max_depth': 14, 'max_leaf_nodes': 92, 'min_samples_split': 2}** and we got an accuracy of **83%** And F1 score of **0.83.**

Results

Model Name	**Accuracy**	**F1 Score**
K Nearest Neighbors	**99%**	**0.99**
Decision Tree (optimized)	**83%**	**0.83**

Discussions and Conclusions

After applying both the models and getting the results we can clearly see that KNN outperformed the Decision Tree by a huge margin with a high accuracy of up to 99% in contrast to the 83% accuracy of decision tree. In real life we generally prefer a model that is more robust, convenient for us to use, requires less pre-processing and is more computationally efficient. Decision tree does fit these criteria's while on the contrary KNN is way more computationally expensive but since we have a dataset of 1080 measurements and few features it allowed our KNN to overcome its drawbacks and perform exceptionally well. Hence, we will be sticking with KNN for this problem.

For the future we would like to use more advanced machine learning methods like Random Forest or maybe Neural networks as well but since our problem does not require all that we don't need to worry about that right now.

References

https://rmit.instructure.com/courses/67430/files/11552799?module_item_id=2314376

https://rmit.instructure.com/courses/67430/files/12058206?module_item_id=2369057

https://medium.com/machine-learning-101/k-nearest-neighbors-classifier-1c1ff404d265