REPORT ON MODELING A CLASSIFIER

Identify a subset of proteins that are distinguished between eight discrete classes

Kaushal Gawri[1]

s3777121@student.rmit.edu.au

DATE OF REPORT – 8TH JUNE 2020

[1]Bachelors of Computer Science, School of Science, Computer Science and Information Technology, RMIT University, Melbourne, Australia

ABSTRACT

I report two different classification model, the Random Forest and Decision Tree modelling of thee expression levels of proteins critical to learning in some mice for the given mouse model of Down Syndrome in which the proteins were responsible in producing detectable signals in the nuclear fraction of the cortex. This work exhibits the performance evaluation for various Random Forest and Decision Tree configurations by choosing optimum hyper parameters and best fit features to built the best model with highest accuracy without overfitting or underfitting the model and hence comparing the classification accuracy for both the models. The Random Forest derived entails 100 trees in the forest with 21 selected features to yield the maximum accuracy.

INTRODUCTION

The dataset provided to us, consists of a total of 1080 observations per protein and is confiscated from the UCI data repository. The dataset provided was in the form of xls format and it was recommended to convert it to suitable CSV(a.k.a comma-separated values) file using the Microsoft Excel. Each observation/measurement among the 1080 provided is considered as an independent sample(here, a mouse) .The dataset provided comprise of 82 different attributes, among which there are expression levels of 77 different proteins that basically were responsible in producing detectable signals in the nuclear fraction of the cortex. Mouse models are considerably a standard instrument within the examination of various human afflictions, in principal investigation and in preclinical assessment of potential therapeutics. Their utilization in the sub-atomic, cell and behavioural trials can provide a good understanding about the standard components of a particular quality, how these are altered in affliction and how do they emphasise a malady process and in expansion, the information on pharmaceutical action, ampleness and responses to a particular protein. This particular disorder has been regarded by various (in both fundamental and clinical investigation) as unreasonably complex test for effective pharmaceutic arbitration. Down Syndrome is considered as one of the foremost common hereditary innate causes of learning deficits, the dataset consists of some behavior of the mice who either have been fortified to learn and some who haven't,. It could be a hereditary annoyance of significant complexity due to the trisomy of the long arm of human chromosome 21 and the ensuing level of expression of few subsets of the qualities it encodes. By and by, protein expression modelling is additionally turning into an incontestably stead technique in microbial cell production lines as the learning of the three-dimensional structure of a protein would be a valuable direct to require care of issues on the protein era.

Protein expression has been reported by many authors in the literature, but to the best of my knowledge, there aren't many instances regarding the use of some machine learning models for classifying the mice protein expressions. In this particular case, I will be building two different classification models using the data set provided and would be predicting the future theories related to it by using appropriate algorithms, in my case I would be building two different models, a random forest classifier and a decision tree classifier over the class along with the other attributes including Genotype, Treatment and Behavior. I would be training 2 different models over all these four features which will be followed by accomplishing the feature selection using one of the techniques and tuning the hyper parameters used by model to ultimately get the most accurate model. This will be followed by comparing both the models over all the attributes, but specifically for the class which is my main focus(as the class attribute is basically a combined feature of Genotype, Treatment and Behavior).

According to the description of the data set provided, around fifteen observations were recorded of each protein per sample(here, mouse). The eight classes of mice are depicted based on some features such as genotype, behaviour and treatment. Genotype classifies mice to be either control or trisomic whereas behavior depicts that a few mice have been fortified to learn(context-shock) and others have not(shock-context) and in order to evaluate the impact of the drug memantine in recovering the ability to learn in trisomic mice, a few mice have been infused(or injected) with the remedy and other have not.

The following information is provided to us regarding the attributes provided in the dataset:-

Mouse ID – Used as a count for the mouse(Not required in our analyses)

Different types of Proteins(77 in total) – The names of most of the proteins are followed by a '_' stating that they were calculated in the nuclear fraction.

Genotype – Either control(c) or trisomy(t)

Treatment – Either memantine(m) or saline(s)

Behavior – Either context shock(CS) or shock-context(SC)

(Context shock(CS) - the mice were fortified to learn and Shock-Context(SC) - the mice were not fortified to learn)

Class - Depicting different combinations of the Genotype, Treatment and Behavior

A table indicating all the possible values for the "class" attribute, is displayed below:-

c-CS-s	control mice, stimulated to learn, injected with saline
c-CS-m	control mice, stimulated to learn, injected with memantine
c-SC-s	control mice, not stimulated to learn, injected with saline
c-CS-m	control mice, not stimulated to learn, injected with memantine
t-CS-s	trisomy mice, stimulated to learn, injected with saline
t-CS-m	trisomy mice, stimulated to learn, injected with memantine
t-SC-s	trisomy mice, not stimulated to learn, injected with saline
t-SC-m	trisomy mice, not stimulated to learn, injected with memantine

METHODOLOGY:-

The first and foremost step to start my analysis would be to retrieve the data from the UCI webpage provided. Given, the dataset is provided in xls format, I am required to convert it to suitable csv format, by using the appropriate software, here I would be using Microsoft Excel. After transforming it to a CSV file, I would be making the use of pandas data frame to read the data from the file as required. The data provided to us consists of 1080 measurements and 82 different sets of attributes, out of which 81 are useful to process our analysis. Among these 81 attributes, 77 attributes are numerical and continuous which will be used later on to train our machine learning model. The rest of the four columns are categorical, though I would be building the model on all the four attributes, but my main focus would be to build a high accuracy model(without overfitting it) to formulate the "class" of the certain measurements as a classification problem.

Now, I was required to start the data cleaning process for which I am supposed to :-

1. Check for any null values present in the dataset
2. Perform sanity checks for any impossible values present in the data.
3. Check for any extra whitespaces present in the data.

➔ Check for extra white spaces were performed using the pandas.Dataframe.value_counts() which basically checks for different types of values present in the categorical features.
On performing the required tasks, we found out that there were no extra whitespaces present in the categorical features of our datasets.

➔ Check for impossible values were completed using the pandas.Dataframe.value_counts() which sanity checks all the numerical data and tells us about the different values present. Using it I found out that there are no impossible values present, though after closely examining the numerical data, I did find out some outliers.

Till now, I can say that the data provided to us is mostly clean. It will be followed by checking for the missing values.

➔ Checks for missing values was done by invoking the pandas.Dataframe().isnull().sum() which tells us about the number of missing values present. It was further confirmed by using pandas.Dataframe().info() function and through that I got to know that there were quite a few null values present in the numerical data. The null values were then replaced by the mean of the numerical data to complete our data cleaning process. This was confirmed by using the pandas.Dataframe().describe() which clearly stated a slight increase in the statistical values like mean, standard deviation etc.

With this, the data cleaning process was completed and the focus was shifted to explore some of the most suitable attributes.

Data Exploration:-

The next part of the process included exploring :-

- Each column (or at least 10 columns if there are more than 10 columns), using appropriate descriptive statistics and graphs (if appropriate). For each explored column, I would be stating some conclusions that I can derive from the explored column after the visualization was completed.
- The relationship between all pairs of attributes (here, as the number of attributes is quite high, I would be exploring a satisfactory number of columns) with a plausible hypothesis clearly stating the reason for choosing them and stating if we reject or fail to reject the null hypothesis.

Univariate Exploration :-

To explore each column separately, I would be initially focussing on creating some meaningful plots, like the histograms which might be helpful in stating some meaningful conclusions derived from the shapes of the plots.

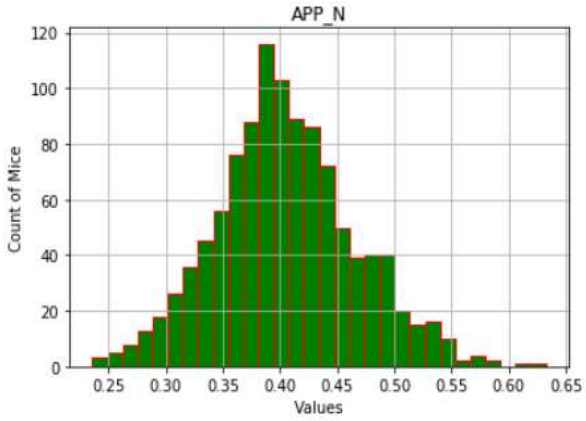


Figure 1 : - Symmetrical Distribution of APP_N

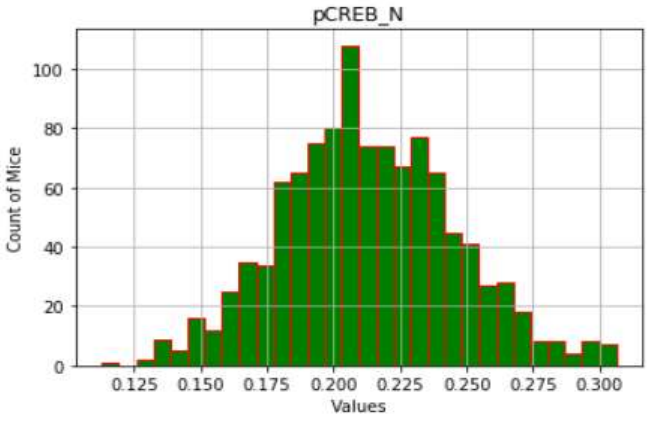


Figure 2 : - Symmetrical Distribution of pCREB_N

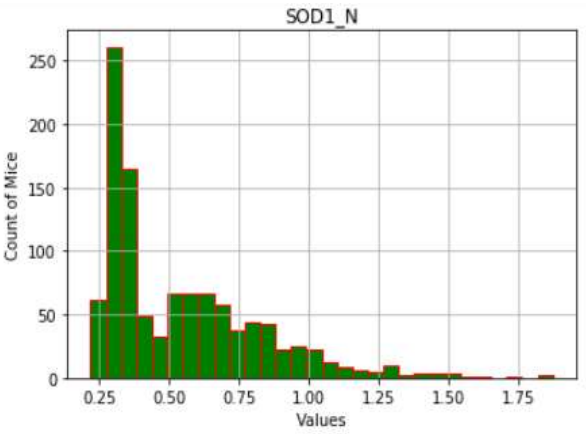


Figure 3 : - Positively Skewed Distribution of SOD1_N

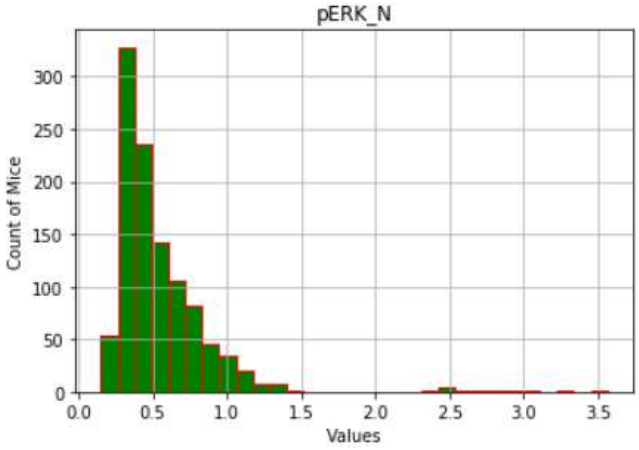


Figure 4 : - Positively Skewed Distribution of pERK_N

We basically tend to use histogram when we have continuous numerical measurements and we want to understand the distribution of values and check the outliers. The values in the histogram are displayed in the form of bins which are basically the continuous range of values for a particular attribute.

Figure 1 displays a histogram plotted over a continuous numerical data of protein “APP_N”. The shape of the given histogram is mostly symmetric which states that the distribution is close to being normal. From the histogram, I can conclude that the mean value for protein “APP_N” lies around 0.40.

Similarly, Figure 2 displays a histogram plotted over a continuous numerical data of protein “pCREB_N”. The shape of the given histogram is mostly symmetric which again states that the distribution is close to being a normal distribution. From the histogram, I can conclude that the mean value for protein “pCREB_N” lies around 0.21.

I would like to follow my data exploration process by talking about another attribute, similarly a continuous numerical data for protein “SOD1_N”. The shape of the given histogram is skewed, the skewness is towards the right, so I can conclude that it is positively skewed. As for positively skewed, the mode is the highest peak, we can conclude that it will be the mean which will be the maximum. In this case, the mean is greater than mode. Hence, mean>mode.

This was followed by exploring another attribute, similarly a continuous numerical data for protein “pERK_N”. The shape of the given histogram is skewed, the skewness is towards the right, so I can conclude that it is positively skewed. As for positively skewed, the mode is the highest peak, we can conclude that it will be the mean which will be the maximum. In this case, the mean is greater than mode. Hence, mean>mode which is similar to the other case.

Now, plotting a histogram not only helped me in gaining an insight about these statistical values like the mean, it can also help in detecting the outliers. As seen from the above plots, figure 1 and figure 2, having the symmetric distribution have a negligible to zero outliers as visible on the plot. But comparing it to the right skewed graphs, that is figure 3 and figure 4,

we can detect a lot more outliers for protein “SOD1_N” and “pERK_N”. As we can detect from the graph, the outliers for protein “SOD1_N” are basically present at a value of greater than 1.50 whereas for protein “pERK_N” have some outliers at a value of greater than 2.5.

Now, even though it is a good idea to plot histograms for numerical data, it is quite confusing to decide the value of mean, mode and median just by looking at the plot as the precision isn’t much. Though we get an insight of which value is greater amongst the three, mean, mode and median, it is quite difficult to compute the exact value. Also, depending on the number of bins, if we don’t use appropriate value, there is a chance that we might not be able to detect the outliers.

So, to overcome the above situation, I would be creating different boxplots and violin plots, to get a more suitable statistical values which might be useful in the later stages.

Considering, violin plots and boxplots are quite similar, the main reason for using both the plots is that violin plots provides information about the probability density of the data at different values whereas it is quite easy to detect the outliers from the boxplots along with the mean and median values.

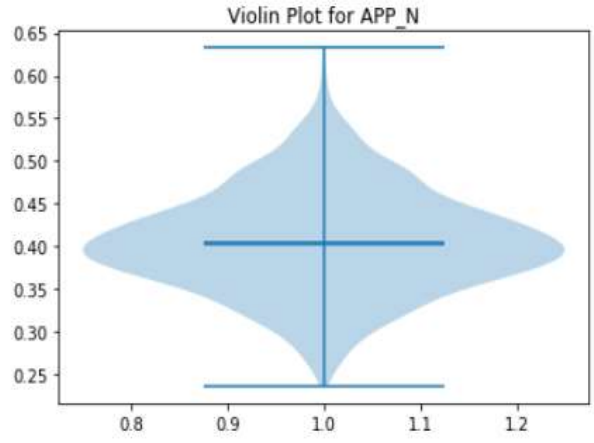


Figure 5 : - Violin Plot displaying density distribution Of APP_N

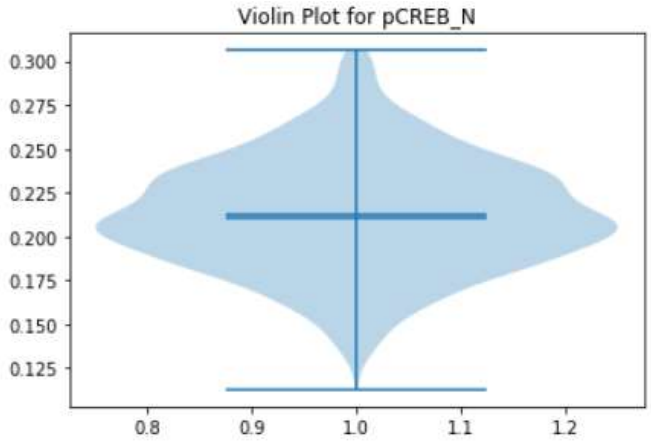


Figure 6:- Violoin Plot displaying density distribution Of pCREB_N

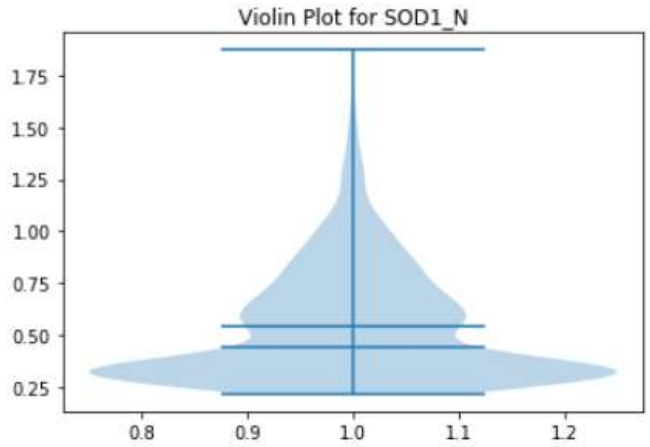


Figure 7 : - Violin Plot displaying density distribution Of SOD1_N

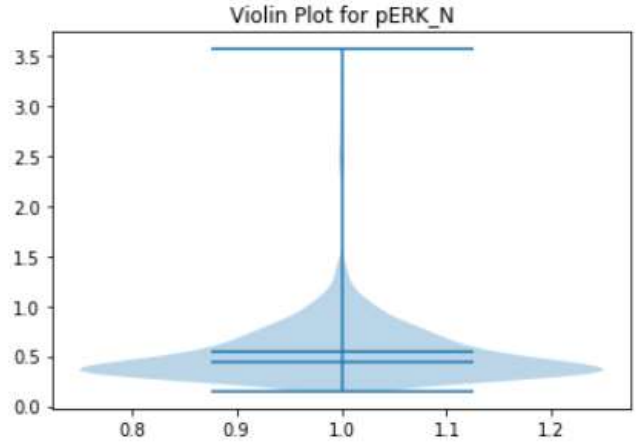


Figure 8:- Violoin Plot displaying density distribution Of pERK_N

Considering the above figures, it is quite visible from the violin plots for protein APP_N and protein pCREB_N that the mean and median value for the proteins are almost equal which further states the symmetry in the data for both the proteins. Expanding the reasoning from the histograms, violin plots provide us with a conclusive evidence that the distribution of both the proteins are almost normal with a mean and median value of around 0.40.

From the last two figures, we can see two different line for the mean and median of the sample, and a skewness towards the higher values(>>0) which states the positive skewness though most of the probability density is distributed for a given range with the peak value(that is most number of measurements) for a value of around 0.32 and around 0.40 for both the cases. From the figure, we can also conclude some of the statistical values like the mean and the median for both the distribution,

with mean and median of around 0.55 and 0.45 for the both the plots, which implies the value of mean is greater than the median and hence the positive skewness is justified.

A few box plots were also made for a number of variables that provided us with not only a clear insight over the mean and median values but also helped us in detecting the outliers in a more efficient way.

Some bar graphs and pie charts were plotted to get an insight about how the data is distributes amongst different groups of genotypes, treatment, behavior and ultimately class. This helped me in concluding that most of the distribution consisted of a greater number of mice with Control genotype as compared to Ts65Dn whereas more mice were provided with a treatment(here, injected with) Memantine as compared to Saline. Also, a few a greater number of mice were not fortified to learn(i.e. Shock Context) as compared to the ones who were(i.e., Context Shock). Almost identical values were seen for different classes though controlled mice who are injected with saline and memantine and stimulated to learn had the greatest number of measurements. Though there was not much of a difference seen, the given plots in the notebook justify the statements made above.

Most of the plots, including all the histograms, violin plots and box plots suggested that though most of the attributes of different proteins had some sort of symmetric distribution, there were quite a few attributes having some kind of skewness(positive). Though a number of plots were made but they only provided me with understanding of only a single attribute, rather I would want to explore the presence of few relations for multiple attributes which ultimately lead me to multivariate data exploration.

Multivariate Exploration :-

The multivariate exploration included plotting two protein over a histogram to compare the symmetry present in the distribution which ultimately provides me with an insight on how the difference in some of the statistical values like the mean varies, with my null hypothesis being:-

H01 – There is no statistically significant difference between the means of different protein for the trisomic mice who are fortified to learn(CS) and who aren't fortified to learn(SC) and are injected with Memantine

H02 – It is expected that it is possible to divide the trisomy mice from the control mice ,i.e., to separate both

HA1 - There is no statistically significant difference between the means of different protein for the trisomic mice who are fortified to learn(CS) and who aren't fortified to learn(SC) and are injected with Memantine

HA2 - It is expected that it is not possible to divide the trisomy mice from the control mice.

The same null hypothesises were used on different pair of proteins, and a few are discussed below. To get an insight on all the expected plots, please refer to the Jupyter notebook.

To study my hypothesis in more detail I first plotted multiple histograms for two different proteins on a single graph to compare the symmetry of two different proteins which will further help me in comparing the means for the same. Given, the proteins either had a normal distribution or positively skewed, it is a tedious task to first divide the proteins into two categories and then compare the normal distribution with positively skewed(Impossible to compare normal-normal or skewed-skewed). For the same reason, it was not quite achievable to compare two different proteins solely on the basis of histogram, given according to my hypothesis, I further wanted to compare the mean values of different trisomic mice depending on if they are forced to learn or not, I plotted the data for a particular protein grouped by different classes(8 in total) on a box plots.

Some of the plots that helped me in concluding my hypothesis are provided below:-

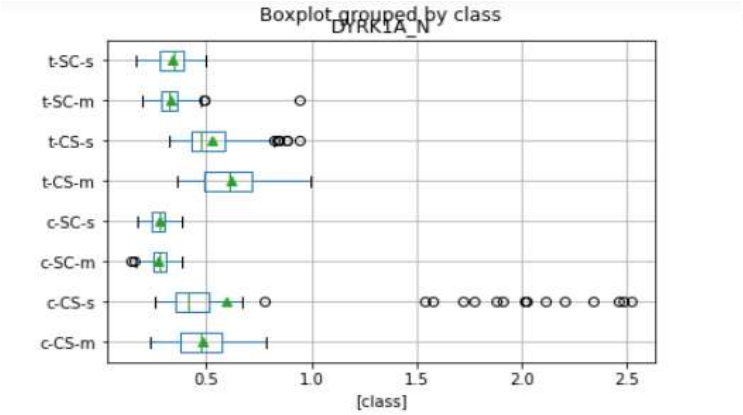


Figure 9 :- Boxplot of DYRK1A_N grouped by class

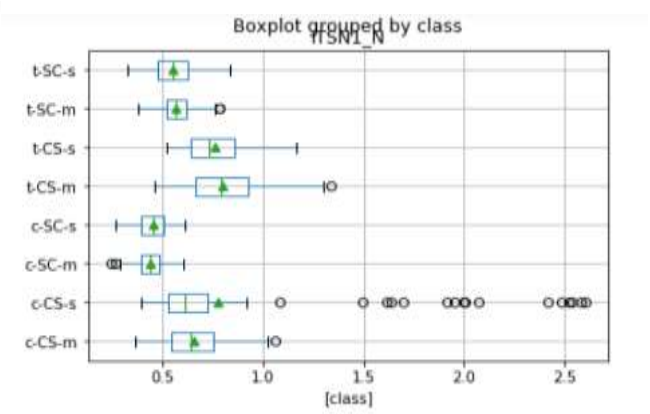


Figure 10 :- Boxplot of ITSN1_N grouped by class

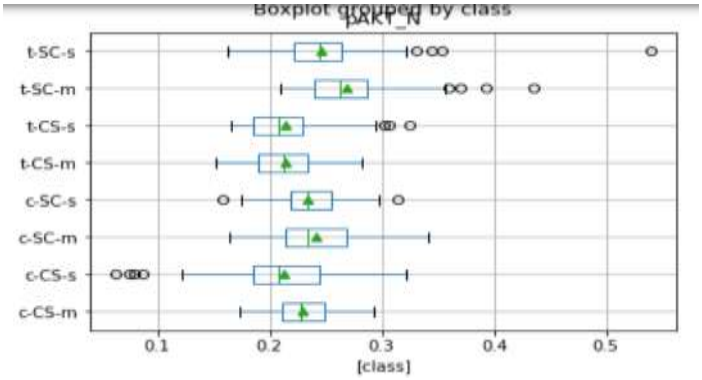


Figure 10 :- Boxplot of pAKT_N grouped by class

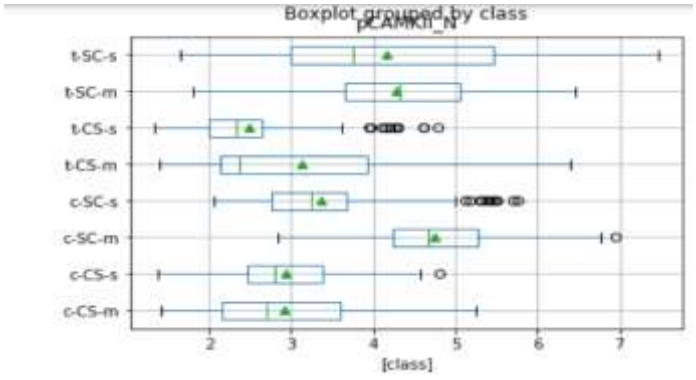


Figure 11 :- Boxplot of pCAMKII_N grouped by class

Conclusion for first null hypothesis :-

From the plots provided above, we can see in the boxplot the different means of a particular protein. We can see that there is a significant difference between the mean of some of the proteins for the trisomic mice depending on if they are simulated to learn or not. This further state that trisomic mice learn a lot better with the presence of some proteins like DYRK1A_N, whereas presence of proteins like pCAMKII_N doesn't help in letting the mice learn. Based on this, we can say that the p-value must be less than 0.05 due to which I can conclude that I reject my first null hypothesis of there being no significant difference between the mean values of such mice.

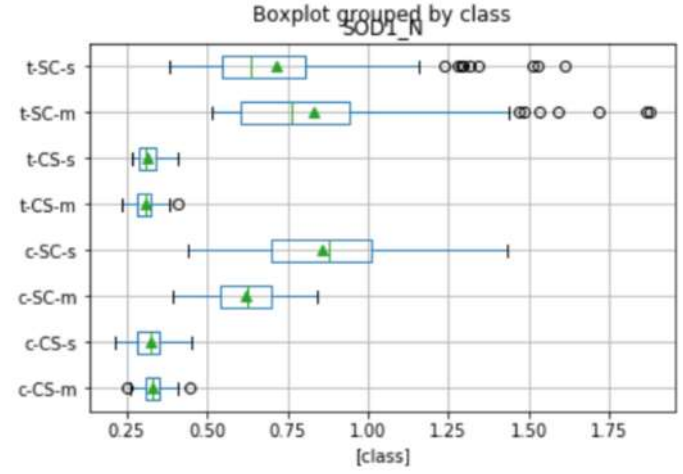


Figure 12 :- Box Plot of SOD1_N grouped by class

Conclusion for second null hypothesis :-

Also, from the figure shown, we can see that there is a huge difference between the mean values for different classes for trisomic and control mice, and hence we can conclude that it is feasible to divide the mice based on if they are trisomic and control mice. Presence of such proteins like SOD1_N, helps in differentiating the two Genotypes of mice and as, we conclude our second null hypothesis to be true, I declare that I fail to reject my null hypothesis, given the p values calculated would obviously be less than 0.05, which is enough to conclude that I fail to reject my null hypothesis two.

A few more plots for different test cases are provided in the notebook attached, but it is enough to discuss about some of the proteins that help us in rejecting our null hypothesis.

Hence, I reject my null hypothesis one and fail to reject my null hypothesis two.

With this, the data exploration for both univariate and multivariate exploration is concluded with plotting some histograms, violin plots, box plots and bar graphs.

In my opinion, it is enough to start the next step of the process, i.e Data Modelling.

METHODOLOGY AND DISCUSSION :-

Data Modelling:-

In this task, I am required to model the data either by treating it as a classification or a clustering task. Given, in my knowledge, there ain't any models build on the given data set of different mice protein with taking classification into consideration and hence I would be building two different classification models, the random forest classifier and the decision tree classifier specifically on the class attribute, but I would be training them on other categorical values of Genotype, Treatment and Behavior to conclude the importance of some given features.

Random Forest Classifier: -

As we already know, a Random forest is an ensemble of some decision trees, generally trained using the Bagging method, typically with the help of some indeed useful hyper parameters like the value of max_samples being set to the size of the training set. Instead, I would be using the RandomForestClassifier class, which is more convenient and an optimized version of decision trees. Though I would be building two different model, the one with the RandomForestClaasifer and the other with the Decision Tree classifier and would be comparing the accuracy of both the models and provide justification for which is the better build. Basically, a random forest is a collection of unpruned decision trees. Though it is often used in a very large training datasets and with a very large number of input variables, I would be trying to achieve the maximum accuracy without overfitting the data with the help of it on my dataset. The random forest algorithm introduces extra randomness when growing trees instead of searching for the very best feature when splitting a node, it searches for one of the best features presents amongst a random subset of features. This helps in increasing the diversity, which in turn trades a higher bias with lower variance generally ending up yielding an overall better model.

Splitting the Data: -

The process of training our model starts by splitting the data into training and testing sets. Before splitting the data, I was required to achieve the training data, with the input data attributes(all the numerical columns), i.e, proteins to be the data used by the model to find patterns and predict to the target, which is the "class" attribute for first model. This is achieved with the help of train_test_split imported from the model_selection package of sklearn machine learning library of python. The training of the dataset to build predictive model was accomplished using python with the help of appropriate algorithms. Now, to split the data, we need to choose the test_size of data, though the split is said to be ideal with a 80:20 ratio, which is basically justified using the Pareto principle, but I would be splitting the data in the ratio of 60:40 with test_size being 0.4, as the dataset provided is not that large, with only 1080 observations, we might not get accurate result while using this data with 1080 observations as splitting it to 20% would reduce the number of observations even more, which might overfit the random forest model. Hence, to reduce the overfitting, the data would be split into 60:40 ratio with the test 40% of the data being used as the testing set while we train on the 60% of the data.

Choosing the suitable model: -

The next step will include importing the RandomForestClassifier from sklearn.ensemble, and making a classification using the particular algorithm. Initially, the hyper parameters for the classifier include the value of,

n_estimators, to be equal to 100 (which is the default value), which is the number of trees in the forest.

random_state, to be equal to 1, which controls the sampling of the features to consider when looking for the best split at each node.

n_jobs, to be equal to 1, which is basically the default value of the number of jobs(here, fit and predict) to be ran in parallel.

Hyper-parameter Tuning: -

With the use of the current hyper parameters and all the features responsible to train the model(all the proteins, that is all the continuous numerical feature, which are 77 in total), the accuracy recorded for the model was around 98.37%. Though, an accuracy of 98.37% is quite good to predict on the future data, still I wanted to tune the hyper parameters, to achieve a better accuracy without overfitting model. This was initially done using the GridSearchCV which provides an exhaustive search over specified parameter values. The specified parameter values were in turn taken from the values that were generated through the RandomizedSearchCV, which provided me with some values for different parameters. These values were then used in the GridSearchCV to train the model, which used around 4320 different combinations with the value of cross-validation generator(determines the cross-validation splitting strategy), of 3, making the total value of 12960 different parameter combinations. This took around 9 hours to train a single model, which in-turn is very exhaustive. Though, I gained the best fit

values for the hyper parameters, with most of the values being stated to default and added the hyper parameter of max_depth = 12 and increased the n_estimators to 1000, which are respectively the number of maximum depth of the tree, that is the nodes will be expanded until 12 leaves generated and 1000 different trees were used to successfully build the model. This was earlier done manually to increase the accuracy of the model, by with the help of hit and trial method, different values and different hyper-parameters were used. After the hyperparameter tuning was completed, I was successful in achieving an accuracy of around 98.84% which is around 0.50% greater than the initial value.

Concluding it, I found out tuning the hyper parameters, either with the help of some algorithms like GridSearchCV/Cross-Validation technique/RandomizedSearchCV or by manually adjusting the values of the hyper parameters and introducing new parameters to the model, the accuracy of the initial model can be increased. Thus, hyperparameter tuning, is indeed an important part of building a better machine learning model.

Feature Selection: -

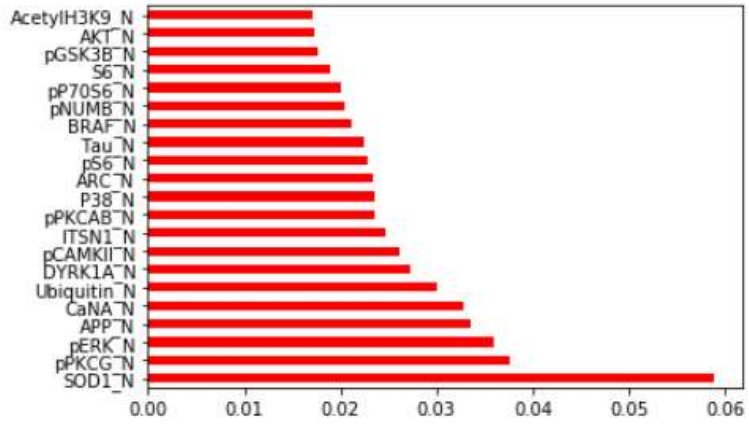
In the present investigation, the next task involved selecting the best fit features(a.k.a attributes) for my model, which will indeed help in increasing the accuracy. To achieve it, I first imported the necessary libraries from the sklearn.feature_selection , the SelectFromModel package. In all the feature selection procedures, it is indeed a good practice to select the features by examining only the training set to avoid overfitting our model. This was followed by getting all the importance values of the given attributes, the protein's in our case, using the feature_importances_, which will basically provide us with the importance values of all the features. This was followed by building a data frame for the same. The next included making an instance of SelectFromModel by passing the classifier to it as a parameter and an appropriate value of threshold(which is the minimum value of the importance value for all the attributes to be considered). This was then fitted on the training set and a new data set was created from the sfm model to test the accuracy with the most important features. This was then followed by creating a new classifier with similar hyper parameters, and then this new classifier was then allowed to fir over the newly generated training data. This was then followed by predicting the values for the "class" attribute over this newly generated data of important features and then the accuracy score was tested against the same. Upon running the model, I observed that the accuracy of the model is increased, from the initial value of 98.84%, now the accuracy stands out to be around 99.074% which is around 0.25% more. The threshold value was then adjusted manually using the hit and trail method, through which I discovered that the accuracy remains constant to a value of 99.074% if and only if 0.014<=threshold<=0.017. Hence, the average value of threshold(=0.0155) was selected, to train the SFM model.

Conclusion for Random Forest Model : -

After the completion of all the above steps, I was able to train my model over the target variable of "class" with eight different values to predict on future data without overfitting the model using only the suitable features(21 features were used for the final model) with an accuracy of around 99.074% which is indeed splendid.

Similarly, to get an insight of how many features would be required and how accurate my model can be to predict over other target variables like Genotype, Treatment and Behavior, I build the same random forest classifiers, after completing the hyper parameter tuning using the manual hit and trial method(using an algorithm like GridSearchCV will get complex and very tedious), the initial accuracy for all the three features was respectively 98.61% for Genotype, 98.84% for Treatment and 100% for Behavior(which might have resulted in overfitting, though even after using different hyper parameters and using different test sizes, the accuracy remained constant). This was then followed by selecting the important features which didn't yield anything as the most accurate model required the use of all the available features.

This further state that it is comparatively easier to separate the mice who have been already trained to learn based on different "class" whereas it is quite tough to classify the genotype, treatment and behavior as it will require several variables.



Decision tree Classifier: -

In the present investigation, next I was required to build another classification model, so that I can compare the model building time, features required and the accuracy for both the model. As, random forest is like a subset for Decision Tree Classifier, the next model that I built was hence a Decision Tree Classifier. The model building process included following similar steps, which required me to split the data in training and testing, tuning the hyper parameters either manually or by using suitable technique(depending on the training time) and selecting the most important features to get a best fit and most accurate model by reducing the probability of overfitting or underfitting the model.

Splitting the data: -

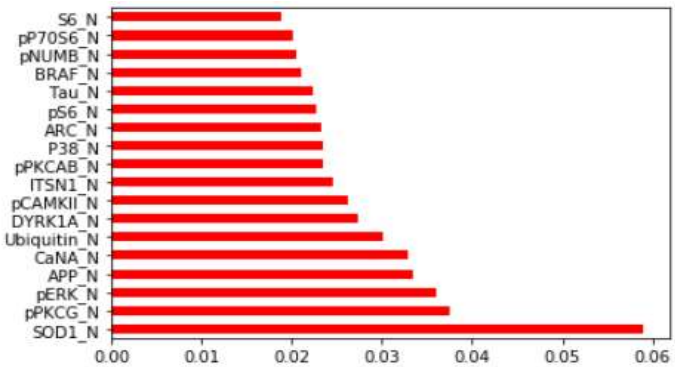
The first task included splitting the data into training and testing with the target variable of "class" attribute. This time, I chose a size of 80:20, which is the most preferred, as the scikit learn library will given us 80% of the returns(bigger chunk of data) with the 20% of the work compared to rolling my own model. After splitting the data in desired portions, I am required to build a decision tree classifier with appropriate hyper parameters, which are derived by manually tuning the current ones and introducing the new ones to our classifier. For my current target variable, class, I decided to use the max_depth to be equal to 12, that is the number of nodes, just like the RandomForestClassifier and the default value of random_state which is 1.

Training the data and Feature Selection: -

The model is then fitted against the training data and then the values are predicted against the testing data. The accuracy that I get after completing the hyper parameter tuning is around 86.57% which is much lesser than what I get by using the RandomForestClassifer. To improve the accuracy, I was required to select the most important features which was carried out by using the SelectFromModel(SFM) of the feature_selection module in the sklearn library. A threshold value of 0.018 is used, as the most accurate model with the least number of features was built. It was then followed by fitting the training data and deriving a subset of data comprising of the most accurate features. Now, using the newly created data with the most important features, a new classifier is made with similar hyper parameters and is thus fitted against the newly fitted training data. It is then used to predict the future values through the newly created testing data and hence the accuracy is computed for the same. Turned out, the accuracy I get after the feature selection is somewhat around 88.88%, at least 2.3% more than the initial value. Though, the accuracy is improved, it is still considered to be way lesser than what it was with the RandomForestClassifier.

Hyper – parameter Tuning: -

Similarly, to get an insight of how many features would be required and how accurate my decision tree model can be to predict over other target variables like Genotype, Treatment and Behavior, I build the same decision tree classifiers, after completing the hyper parameter tuning using the manual hit and trial method(using an algorithm like GridSearchCV will get complex and very tedious), the initial accuracy for all the three features was respectively 91.43% for Genotype, 92.59% for Treatment(initially build a model with just 86.8% accuracy) and 98.61% for Behavior. This was then followed by selecting the important features which then increased the initially values of accuracy for Genotype to 91.66%, for treatment to 93.51% and for Behavior to 99.3%.



This shows that the accuracy increased by using some selected features which was not the case while using RandomForestClassifier.

CONCLUSION:-

In the current investigation, I have reported modelling of expression rates of 77 proteins which are considered essential to learning in the mouse model of Down Syndrome using two different classification techniques, Random Forest and Decision Tree. The dataset with 1080 independent samples of protein was selected and confiscated from the UIC Machine Learning Repository for the preceding modelling. Before starting the modelling, basically there were two null hypothesis stating,

H01 – There is no statistically significant difference between the means of different protein for the trisomic mice who are fortified to learn(CS) and who ain't fortified to learn(SC) and are injected with Memantine

H02 – It is expected that it is possible to divide the trisomy mice from the control mice, i.e to separate both

After exploring different combinations of the 77 different proteins provided to us, there were few test cases in the given sample, which with appropriate visualization provided us with an inference that majorly there was some significant difference between the mean values of at least two different proteins which enforce us to reject our first null hypothesis.

After exploring different combinations of the 77 different proteins provided to us, there were a few test cases in the given sample, which with appropriate visualization provided us with an inference that there were a few groups of protein, through which it is quite attainable to divide the trisomic and the control mice because of some significant difference in the statistical values like mean. Hence, we conclude that we fail to reject our second null hypothesis.

The present investigation suggested optimum use of two different classification techniques, the Random Forest Classifier and the Decision Tree classifier, with some major difference of around 10.2% in the accuracy of both the classifiers. The accuracy of random forest was higher as compared to the accuracy of decision tree with a difference of 10.2% as stated above, while the random forest classifier used 21 features to train the final model to lead the best accuracy, decision tree used just 18, which is 3 less than the number of features used by the RF model. The usage of hyper parameters for both the classifiers were comparable though there were some minor differences because our aim was to build the model with most accuracy and without overfitting it. The current investigation demonstrated the optimum usage of random forest and decision trees architecture by varying its various attributes such as the maximum depth of the tree, the number of trees required to build the model and the choice of variables(here, features) to build the most accurate model.

The significant difference in the accuracy for both the classifiers, though Random Forest being a sub division of Decision, could be because Random Forest leverages the power of multiple decision trees and does not rely on the feature importance given by just a single decision tree, due to which it outperformed the decision tree based model. Also, it is being proven that the decision tree model gives high importance to a particular set of features whereas the random forest chooses features randomly during the training process. Therefore, random forest, like decision tree doesn't depend highly on any particular set of features. Thus, this randomized feature selection makes random forest much more accurate than a decision tree which is furthermore proved by the accuracy of our model with class being the target variable. Though the training time is much more in a random forest classifier as compared to decision tree, it was not a major concern for the current investigation as no such constraints were provided.

Hence, by keeping such facts in mind, I can conclude that the derived Random Forest Classifier model more efficiently classifies the protein samples as compared to the Decision Tree Classifier into the eight different classes with minimum error and without overfitting the model. The result suggests that the Random Forest Classifier has the potential to manifest as one of the best model to predict on future protein samples.

REFERENCES : -

1. https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression
2. https://www.researchgate.net/
3. https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e
4. https://blog.prepscholar.com
5. https://www.kaggle.com