Classification of Attributes of Mice Down Syndrome Through Protein Expression

June 05, 2020

Hikansh Kapoor
S3803669@student.rmit.edu.au
Affiliations: Master of Information Technology, RMIT University

Table of Contents

Abstract

Down Syndrome is a chromosomal abnormality related to intellectual disabilities that affects 0.1% of live births worldwide (Sara S. Abdeldayem, 2018). It usually occurs only when and individual has an extra copy of chromosome 21 and this results in an overexpression of genes that can interfere with normal human responses. Therefore, by studying these proteins and classifying the data, we can get a good amount of information about the drugs that would assist in recovering from the syndrome. In this report, I provide deep analysis of protein expressions and identifying major proteins critical for mice learning ability (experimental). Also, I would classify the mice behaviour, genotype, treatment type and class by using 2 different models (RFC and KNN). These models are then compared to see which one stands out and perform better classification than the other. We can then use these models to predict future data and identify effective drugs to help learning process for people with Down Syndrome.

Introduction

Down Syndrome (DS) is considered to be one of the most common genetic congenital causes of learning deficits (Kamath, 2016). It is related to intellectual disability and physical growth delays (Irving C, 2008). It is caused by the presence of chromosome 21 (partial of full). The chromosome 21, also known as Hsa21 encodes almost 500 gene models (Sturgeon X, 2011). The mouse chromosome 16 closely resembles human chromosome 21. In this report, I have worked with the data provided by UCI for mice protein expression. I have examined 77 expression levels or proteins to identify the several attributes and also the suitable levels for successful and failed learning in a mouse. The data consists of a total of 72 mice those were tested and 15 measurements were taken of each protein per mouse. It is to be noted that each measurement can be considered as an independent sample. Total of 1080 measurements are listed in the dataset.

Main focus of this experiment was to classify different attributes of a mouse given the expression levels of proteins. The attributes consist of Behaviour, Genotype, Treatment type and Class of mouse. The possible values for each attribute are shown below in tables.

Genotype	Behaviour	Treatment
Control (c)	Context-Shock (CS)	Memantine (m)
Trisomy (t)	Shock-Context (SC)	Saline (s)

Class
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-SC-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 whole project and data analysis are carried out in 3 simple steps:

1. Data Pre-Processing
2. Data Exploration
3. Data Modelling and Testing

We would now look at different steps individually and try to explain everything that has been done in the process.

Data Pre-Processing

If we look at the description of the dataset on UCI website, it clearly states that the dataset has missing values. And as we know, missing values are to be removed before we actually start analysing the data. I dealt with missing data in 2 simple steps:

1. Dropping any column if it has more than 75 missing values. In this step, I used dropna() method of pandas' library to drop any column with more than 75 null values. This is because, fixing so many null values in that column would not make much sense and we can focus on other columns for now.
2. Secondly, I used the mean value of every column to fill the null values present in the column by simply using the fillna() method.

Once, we are done with this step of handling missing values, we can step forward and start exploring the dataset as there was not anything that had to be done to clean the data more than what I did.

So, lets step to Data Exploration.

Data Exploration

This is a quite interesting step as I have used different columns to visualize. But the main difficulty that we could have faced was the number of columns that we have in the dataset. As there are a total of 77 protein expression columns, we can't really visualize all of them together and it doesn't make much sense to choose randomly from 77 columns and then visualize 10-15 of them.

What I have done here is I applied Feature selection before Data Exploration to find out the top features that are essential for the model. I have used SelectFromModel to find out every protein that is important for data modelling and then have visualized based on the output of feature selection just to build some useful plots. In feature selection, I used the threshold for importance to be 0.014 as it yields the best results and high accuracy when we trained the model.

Now was the time for data visualization using seaborn library and creating some fascinating graphs and plots. I started by plotting individual columns and did so for top 10 important proteins as per feature importance.

All the 10 plots can be found in the Jupyter notebook, but here I'm just presenting you for the top 5 proteins.

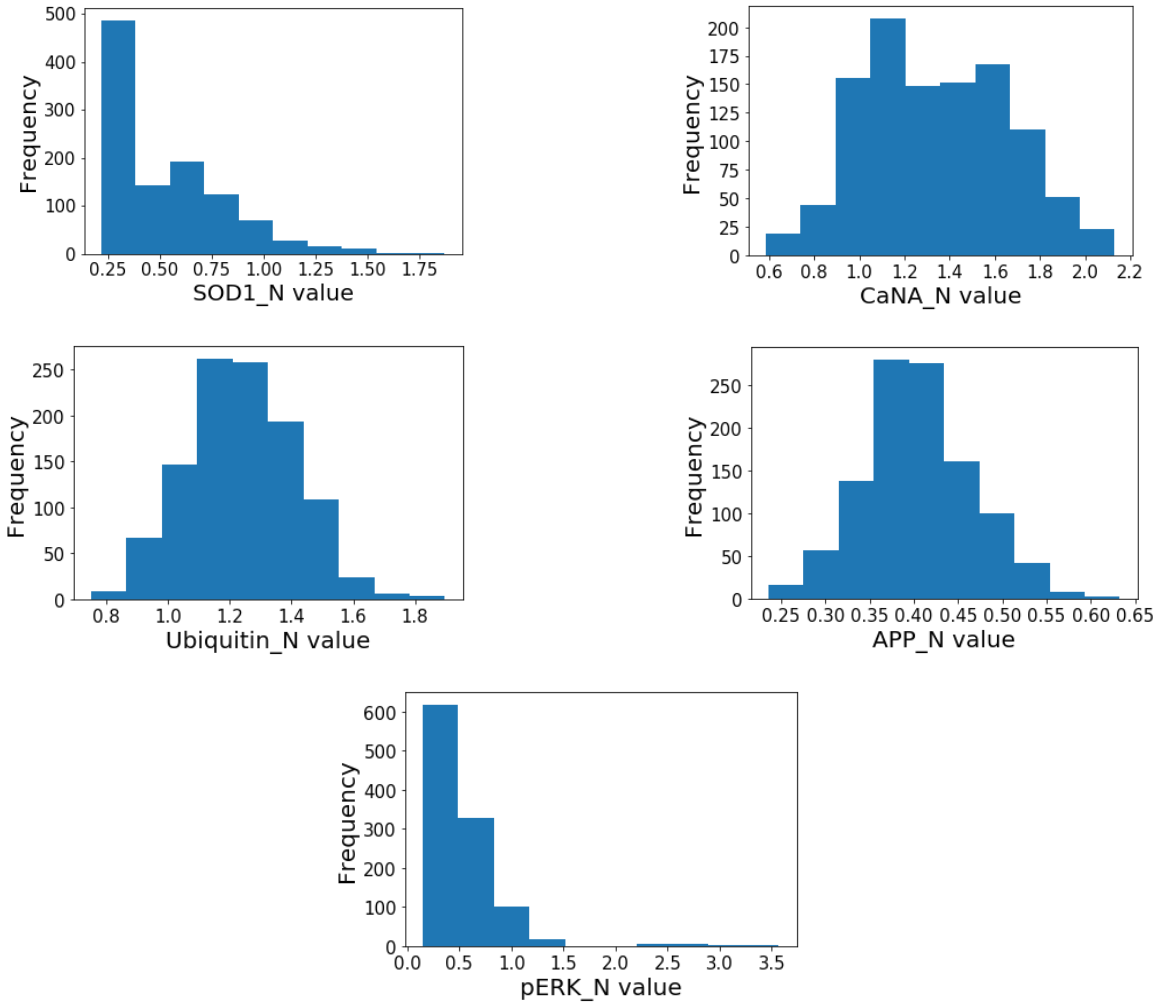

Fig 1 – Individual plots of top 5 features

Looking at individual plots, one can observe that the data was evenly spread across almost all the proteins but we can still see some outliers in the last plot for pERK_N protein. Lets try to plot some useful relationships as it would be good for us to understand the data in terms of other columns as well.

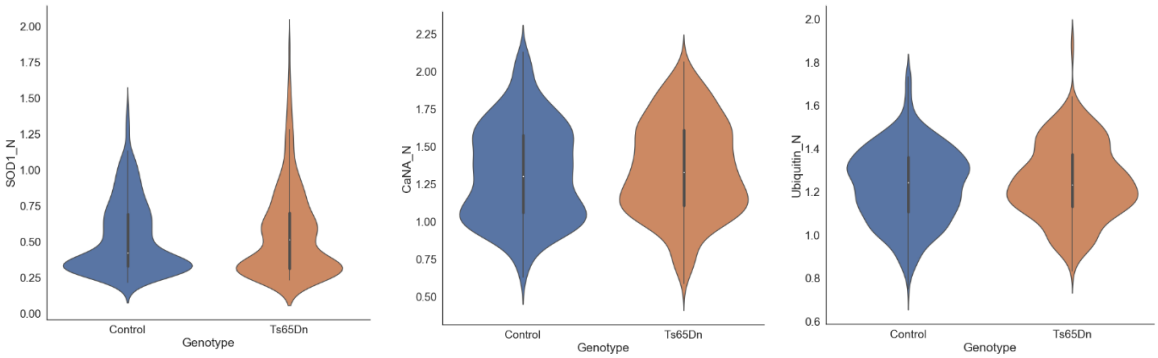

Fig 2 – Mouse Genotype vs top three proteins

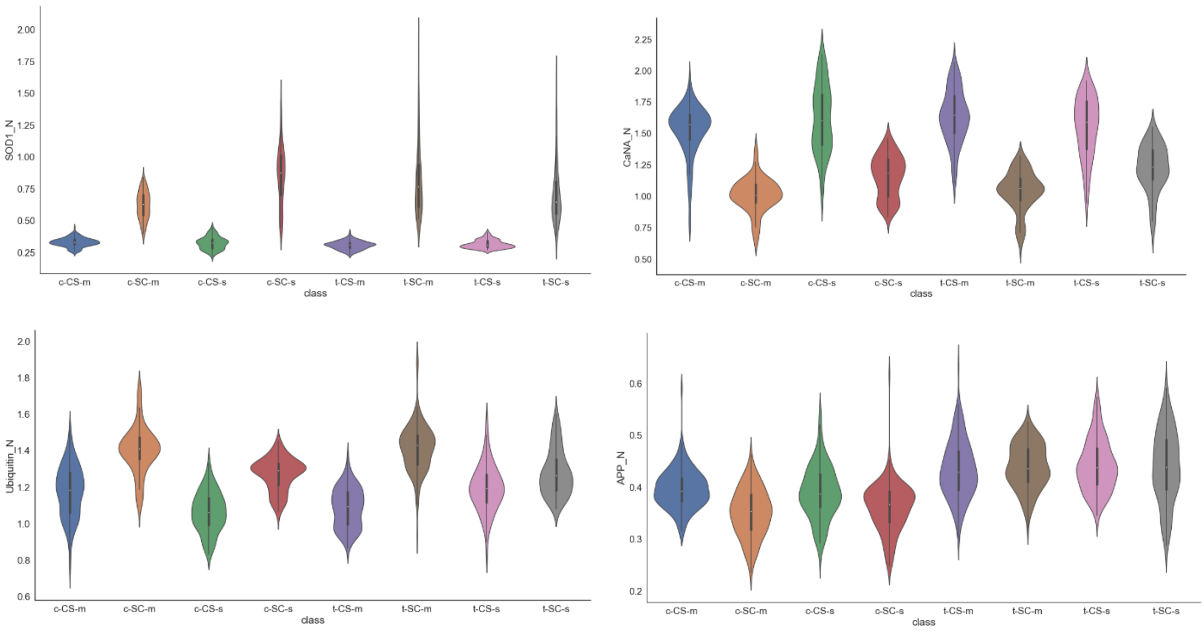

Fig 3 – Mouse Class vs top four proteins

I have included some useful plots in the report (All 10 plots available in the notebook) in figures 2 and 3 where we can see the distribution of every protein in terms of the genotype of mouse as well as the class it belongs to. Both the target columns are explored in terms of top proteins by importance and we can see the various properties of proteins in every plot. For instance, in the case of genotype, all the three proteins shown here are almost equally scattered for both values, 'Control' and 'Ts65Dn' . The violin plots look almost equivalent in each plot for both of these values.

Talking about the relationship with classes, APP_N protein looks quite consistent than others as it is equally distributed as well as scattered for every class type. Other proteins show variance in terms of their distribution and that might be due to the nature of that protein expression and other factors like behaviour and treatment that are not considered for the data exploration step but are obviously targeted in the data training and model building.

Overall, if we talk about exploration, we can conclude that all of our data (columns) in the dataset is in the same range that is between 0 to 3. Hence, we can move forward with the same data and start building our model.

Data Modelling

Before we actually talk about modelling and how I did it. I would like to discuss the models that I used and their background. This will answer the question why I chose these models for classification.

Models Background

Two completely different and efficient models were selected for the purpose of data modelling. One being Random forest classifier and the other being K-nearest neighbours. Both of them are used for classification of target variables based on the inputs provided.

Random Forest Classifier (RFC) is the most flexible and easy to use algorithm. By the name itself one can get an idea about the algorithm that it uses trees for classification i.e. Decision tree is a crucial part of the algorithm and it is based upon that only. RFC creates multiple decision trees, gets predictions from them and finally select the best solution based on voting. It also provides very good indicator of the feature importance as I have used in the notebook (Navlani, 2017).

Similar to RFC, I have used K-nearest neighbours (KNN). It is also a classification algorithm, but it works differently. KNN algorithm fairs across all parameters of considerations and It is commonly used for its easy of interpretation and low calculation time (SRIVASTAVA, 2018). It works on classifying the nearest neighbours as the same class label and the far ones to the different label. The implementation of the algorithm is quite easy and hence it is widely used.

Feature Selection

As discussed already, I have used feature selection before the data exploration just to visualize important data because of so many columns. I used SelectFromModel to select the important features from the dataset for classification via RFC. Putting it in a simple way, I built the classifier first and then selected the top features that satisfy the threshold value of 0.014 to yield the best output and accuracy. I got almost 23 features that satisfied the threshold and then I put all of them in a separate data frame so that I can work on them.

Training the Model & Hyper Parameter Tuning

As soon as I had my features selected, I trained my RFC model and printed the accuracy for the test d ata set. It came out to be "0.9767981438515081". Now was the time for parameter tuning. I used GridSearchCV for this purpose. Grid search uses a simple concept of training and testing the given parameters and then selecting the best one according to the scoring criteria that was accuracy in my case. I tried some parameters to be tested and by this Grid Search returned the best Hyper parameters for the model. This time the accuracy came out to be "0 . 983758700696055". More than 1% increase is quite good and hence our parameter tuning was successful as well. The same thing wa s done in case of KNN model as well and Grid search was used there too to tune the parameters. Now, as I had tuned the models, I built a classifier for every target variable in our dataset i.e. one for all: Genotype, Treatment, Behaviour, Class. Finally, I trained them and printed the accuracy for every target and compared with the other model too.

Results

Random Forest Classifier

In the case of RFC, I got great accuracy for all the target variables in our data.

1. **Target 'Class':** The model achieved a high accuracy of 0. 983758700696055 in this case. The confusion matrix is as follows:

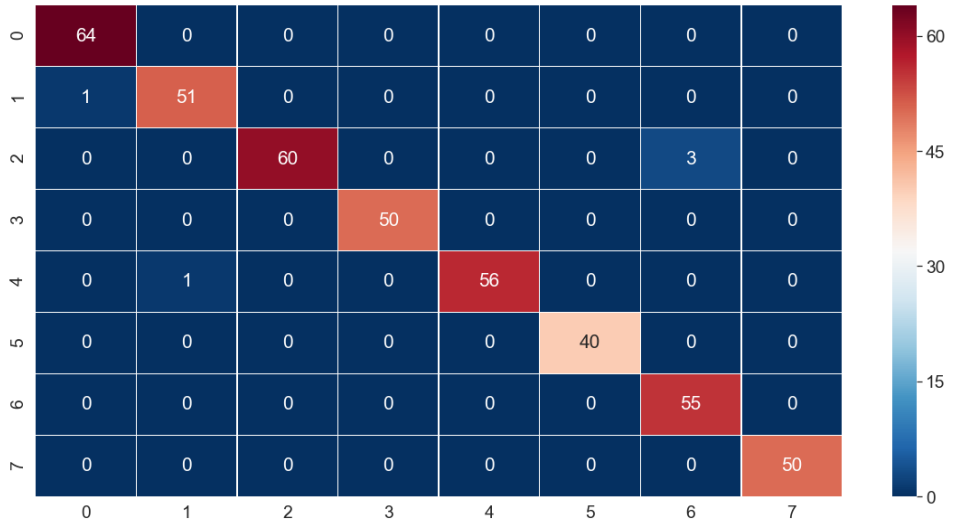

Fig 4 – Confusion Matrix (RFC 'class')

2. **Target 'Genotype':** The model achieved an accuracy of 0.976798143851508 in this case. The confusion matrix is as follows:

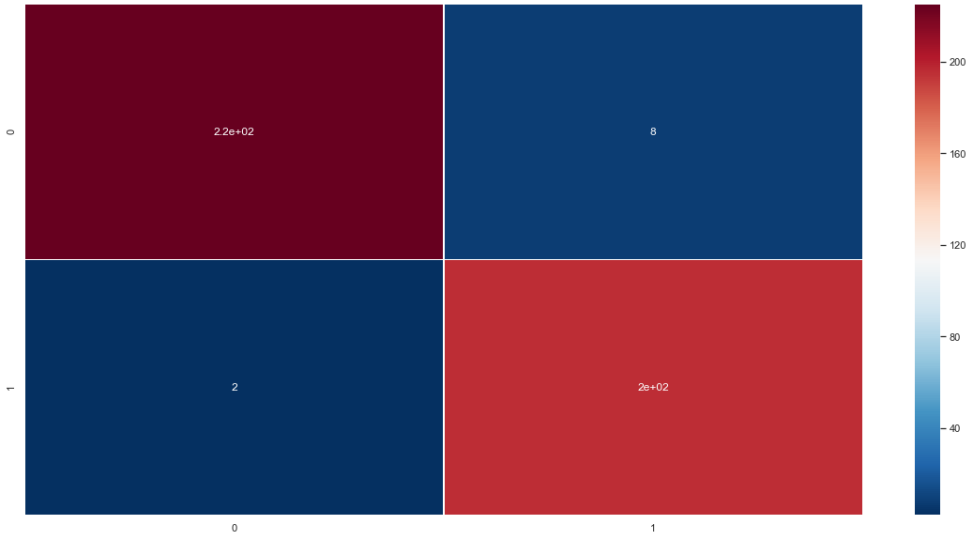

Fig 5 – Confusion Matrix (RFC 'genotype')

3. **Target 'Behavior':** This time the model was 100% accurate and the confusion matrix can be found in the notebook too.

4. **Target 'Treatment':** This time again the model got a high accuracy of 0.988399071925754. You can please refer to the accuracy report in the notebook along with the confusion matrix.

K Nearest Neighbours

In the case of KNN, I got a little lower accuracy if compared to that of RFC but overall this model also produced a great result.

1. **Target 'Class':** The model achieved an accuracy of 0.974477958236659 that is almost 1% less than that obtained by RFC. The confusion matrix is as follows:

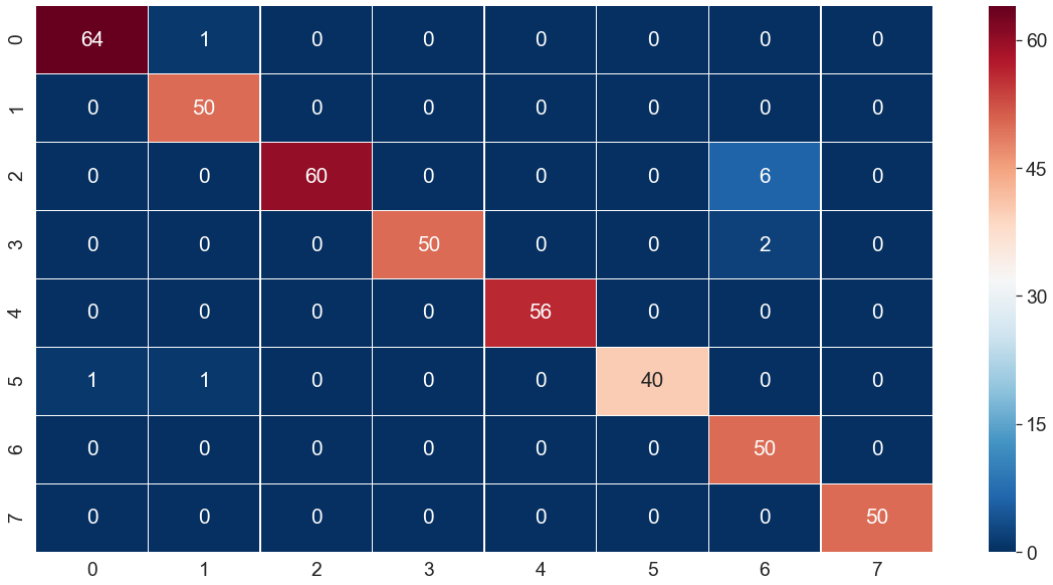

Fig 6 – Confusion Matrix (KNN 'class')

2. **Target 'Genotype':** This model achieved an accuracy of 0.9164733178654292 in this case that is again quite lower(6% approx.) than that already achieved by RFC. The confusion matrix is as follows:

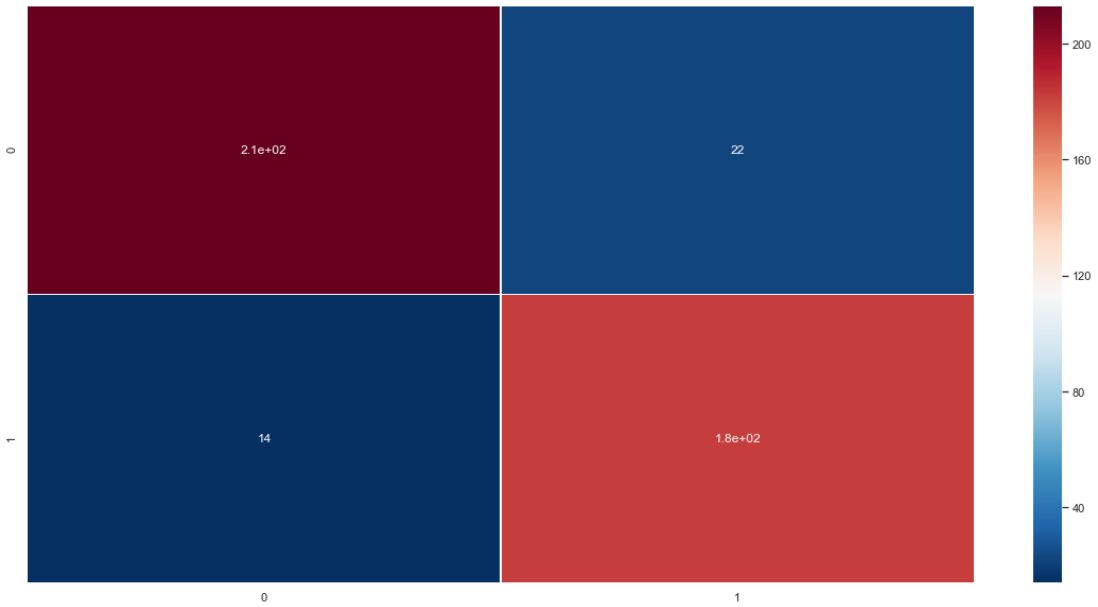

Fig 7 – Confusion Matrix (KNN 'genotype')

3. **Target 'Behavior':** In this case, the model was again 100% accurate just as in the case of RFC. One can refer to the confusion matrix shown in notebook.

4. **Target 'Treatment':** This time again the model got an accuracy of 0.9535962877030162 that is lower than that of RFC by almost 3%. You can please refer to the accuracy report in the notebook along with the confusion matrix.

These were the results obtained during the model testing. Confusion matrix for some target variables are not provided in this document and can be referred to via jupyter notebook. Also, I have constructed an accuracy report for each model and target variable so please refer to the notebook to have a look at it as well.

Discussion

In this section of the report we can try and compare the accuracies of these two models that I have build and see which one is efficient and accurate. I've plotted some graphs to compare the accuracies and to visualize it in a better way that RFC outperforms KNN for 3 out of 4 target variables.

In every graph shown in figure 8, we can observe that the blue bar is either equal to or more than that of the orange bar. That shows how dominating was RFC over KNN in case of accuracy as RFC represents the blue bar and the orange bar is for KNN. In case of treatment, both of them yield 100% accuracy and hence the bars are equal.

By this, we can infer that Random Forest performed really well in terms of accuracy. But, KNN outperformed RFC when it comes to Efficiency as it took really less time than RFC to build, train and test the model. This can be analysed by running the code and observing how long it takes for every model to build, train and test.

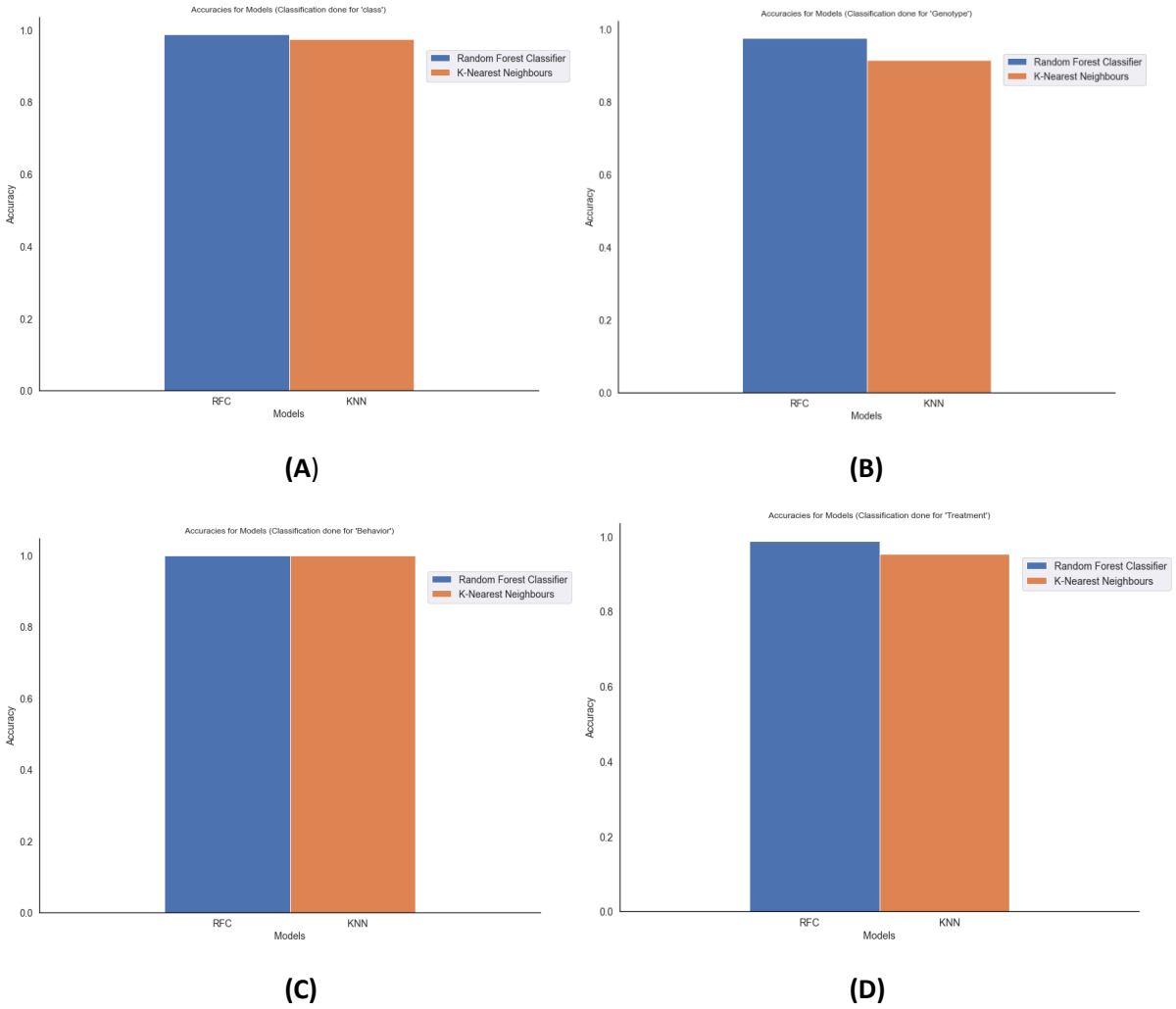

Fig 8 – Comparison of accuracies. (A) For 'class'. (B) For 'Genotype'. (C) For 'Behavior'. (D) For 'Treatment'

Conclusion

To conclude this report, we can point out some major observations recorded during the whole process of data analysis. Firstly, we saw how effective it is to use Hyper parameter tuning in order to enhance the accuracy of the models. Secondly, we learnt how to conduct feature selection based on a new approach of SelectFromModel where we use pre built model and then select the best features from it. Finally, we observed the dominance of Random forest over K nearest neighbour as it outperformed the latter one in all the tests. In the end, we have 2 really efficient and accurate models that can predict several attributes of a mouse, given the protein expression levels for the experiment of Mouse Down Syndrome.

References

Irving C, B. A. R. S. B. J. W. C., 2008. *Twenty-year trends in prevalence and survival of Down syndrome,* s.l.: s.n.

Kamath, R., 2016. *Random Forest Modeling For Mice Down Syndrome Through Protein,* s.l.: s.n.

Navlani, A., 2017. Understanding Random Forests Classifiers in Python.

Sara S. Abdeldayem, M. M. E., 2018. *Deep feature selection for Identification of Essential Proteins of Learning and Memory in Mouse Model of Down Syndrome,* s.l.: s.n.

SRIVASTAVA, T., 2018. Introduction to k-Nearest Neighbors: A powerful Machine Learning Algorithm (with implementation in Python & R).

Sturgeon X, G. K., 2011. *Transcript catalogs of human chromosome 21 and orthologous chimpanzee and mouse,* s.l.: s.n.