# Practical Data Science Assignment 2

## Report Date:

**10.06.2020**

## Author

**Muhammad Ali Tariq s3779826**

## Table of Contents

# Report

## Abstract

This report covers the data science experiment on the Mice Proteins Dataset that are discriminant between the classes. For the experiment, we were required to clean the data and build two data models. We also had to read columns and then explain what the relationships are between columns. Once All the basic data cleaning and analysis is finished, we were asked to create two machine learning model of either two classification models or two clustering models. For this report, two classification models are created and compared. The task is to create and train and improve the models as much as possible and to analyse and suggest a model at the end of this experiment and justify why it is better than the other.

## Introduction

The purpose of this experiment is to understand which proteins help learn stimulate context shock and possibly determine a better machine learning model to generate such data.

The Dataset that was selected for this Assignment was dataset number 3, which represented information about Mice Proteins. The aim is to identify subsets of proteins that are discriminant between the classes and look for their effect on context shock learning of mouse.

## Methodology

### Retrieving and preparing the data

The data was loaded appropriately in pandas and preparation of the data was started. The preparation was done by the following:

1. Data types were checked.
2. Extra-whitespaces and typos were checked (As the most data was numerical this was just skimmed through)
3. Sanity Checks were done (Mostly Data was correct)
4. Null values in the dataset were checked
5. Null values were filled up with the mode values

**Data types** were checked by dataframe_dtypes command. This was necessary as the data types should be known, and we can prepare the data accordingly. To perform this efficiently, skiprows=1 was done, as using that was returning all data types as an object.

**Extra whitespaces and typos** were checked by using the values_counts() function, which tells all the values in the dataset, as most of the dataset was numerical and didn't needed this they were check separately. This was performed on all the categorical data. All the attribute columns had no Typos.

**Sanity checks** were performed by for x in data_p: print(data_p[x].value_counts()), which printed all the values inside the columns.

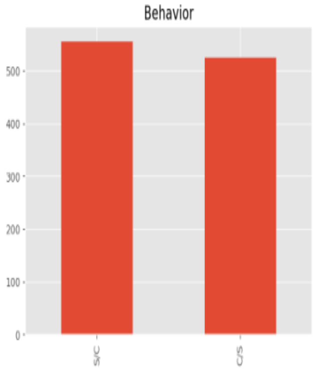
**Null values** were checked in the dataset by isnull function, which wre further made a sum of by using the sum function, which tells the total number of null values for each column in the dataset. As this command didn't provide the whole null values in the column, they were further checked by info() function, which returned total number of null values in all the columns.

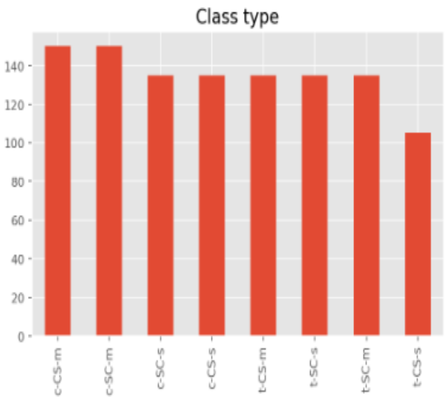
Null values were filled up by using the fillna() function. First the null column was taken, and then the null column was taken and mode value was checked for the column, by value_counts().idxmax(). That mode value was used to fill up Nan values in the dataset.

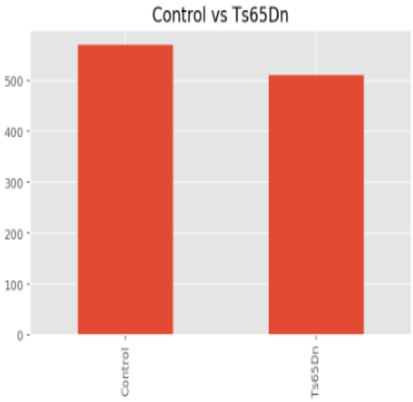
Now, the data was thoroughly checked by filling up the all the null values, if any null value still exists in the column. The data was assumed to be prepared and cleaned.

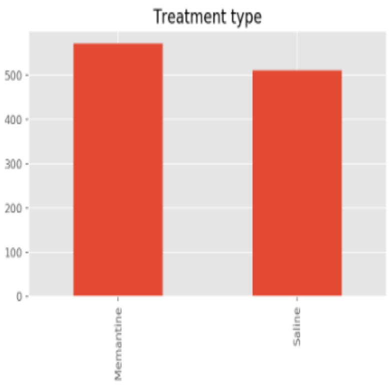
## Exploring Each Column

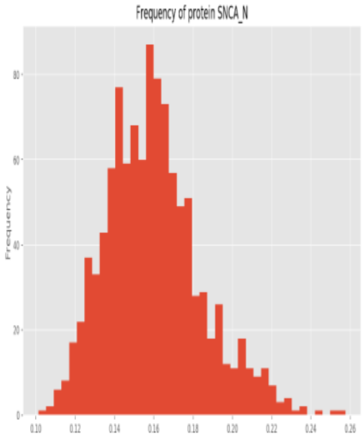
For exploring each column, the first 4 columns that were explored were the attributes in the dataset. The following attributes were checked Genotype, Class, Behavior and Treatment. The number of each types of variable existed was checked inside the column.











The treatment type which show the frequency of each treatment. Rest of the data was numerical, and it was hard to select a graph to represent their visualization. The graph I selected was histogram. It represented each protein quantity between certain intervals.

The following columns were explored pAKT_N, BRAF_N, BCL2_N, pELK_N, H3AcK18_N and SNCA_N. These visualizations explored the whole columns.

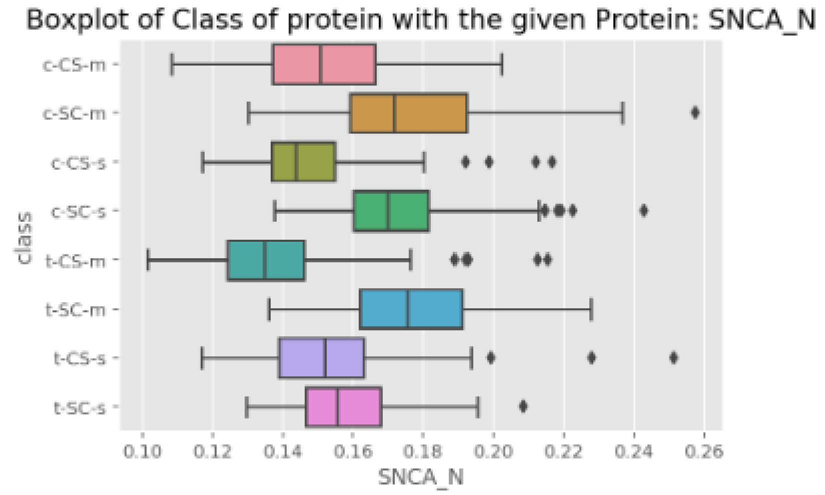
## Relationships between Columns

### Relationship 1

For exploring relationships between column, hypothesis question was made to explore, and the hypothesis were explored by looking at the boxplots made. By looking at the boxplots the conclusion on the hypothesis were made, and it was concluded that the hypothesis was correct or not.

Main Hypothesis = If the protein helps mice to learn stimulate shock or context shock.

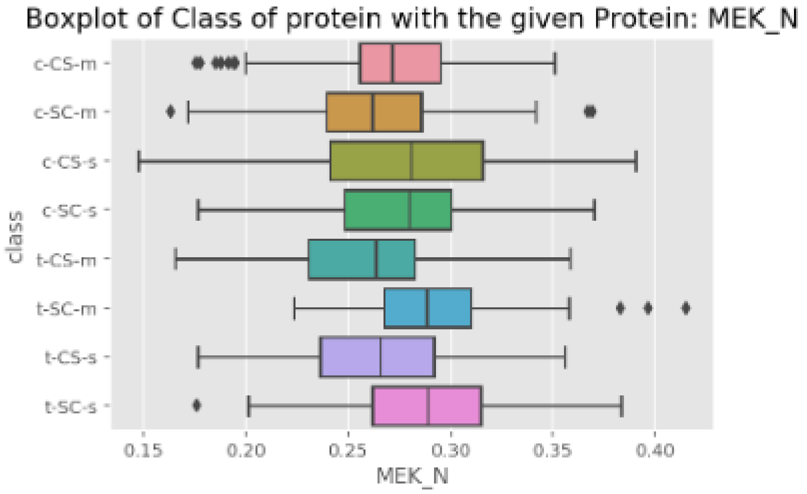
Hypothesis = Mice which are stimulated to learn will have more protein present than others, as it is assumed that SNCA_N proteins help learn stimulate context shock.



The hypothesis is proven wrong, as we can see mice stimulating to learn has lower than the one not stimulated to learn. The median of protein present in the mice c-CS-s (Stimulated to learn) is lower than a mice c-SC-s (not stimulated to learn), and for c-CS-m (stimulated to learn), median of protein present is lower than c-SC-m (not stimulated to learn). Hence, it is the same case for non-control mice (lower protein present for not stimulated to learn. According to this hypothesis, we can say less SNCA_N protein present inside a mouse is more stimulated to learn context shock.

### Relationship 2

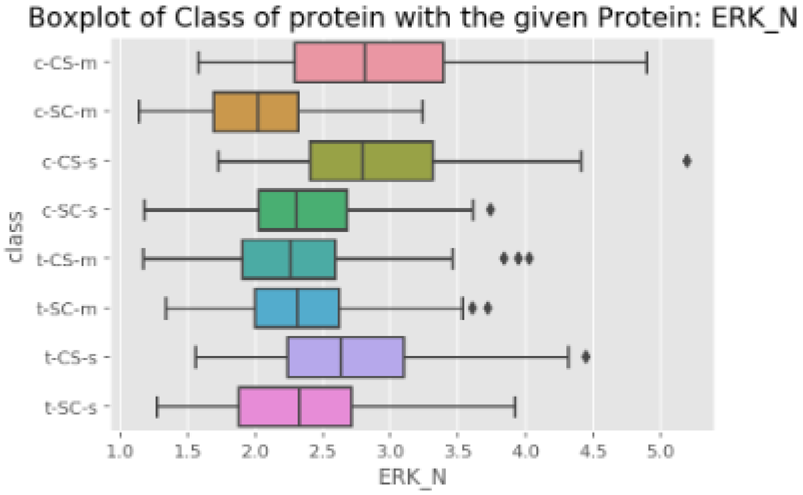
Hypothesis = Mice which are stimulated to learn will have more protein present than others, as it is assumed that MEK_N proteins help learn stimulate context shock.



The hypothesis is nor right nor wrong, as we can see mice stimulating to learn has more median and spread than the one not stimulated to learn in some cases, and its less in some cases. The median and spread for c-CS-m (stimulated to learn), median of protein present is more than c-SC-m (not stimulated to learn). Hence, it is a different case for control mice (more protein present for not stimulated to learn. According to this hypothesis, we can say more MEK_N protein present inside a mouse is maybe more stimulated to learn context shock.

**Relationship 3**

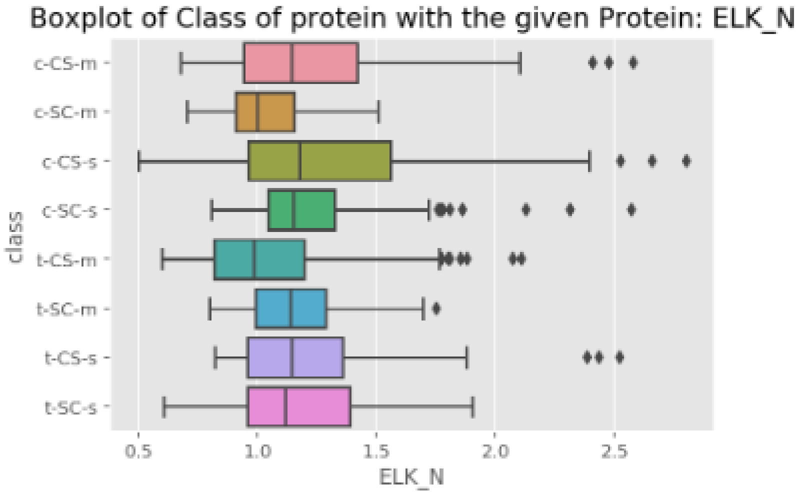
Hypothesis = Mice which are stimulated to learn will have more protein present than others, as it is assumed that ERK_N proteins help learn stimulate context shock.



The hypothesis is proven right, as we can see mice stimulating to learn has more median and spread than the one not stimulated to learn. The median and spread for c-CS-m (stimulated to learn), median of protein present is more than c-SC-m (not stimulated to learn). Hence, it is the same case for control mice (less protein present for not stimulated to learn. According to this hypothesis, we can say more ERK_N protein present inside a mouse is more stimulated to learn context shock.

**Relationship 4**

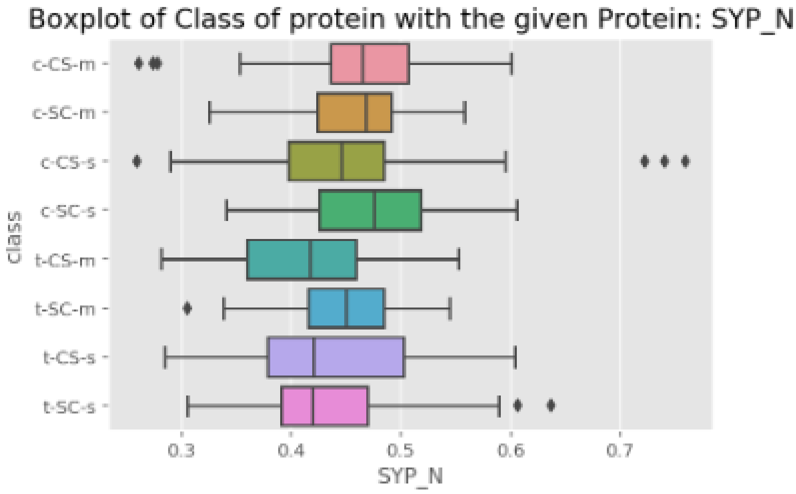
Hypothesis = Mice which are stimulated to learn will have more protein present than others, as it is assumed that ELK_N proteins help learn stimulate context shock.



The hypothesis is nor right nor wrong, as we can see mice stimulating to learn has more median and spread than the one not stimulated to learn in some cases, and its less in some cases. The median and spread for c-CS-m (stimulated to learn), median of protein present is more than c-SC-m (not stimulated to learn). Hence, it is a different case for control mice (more protein present for not stimulated to learn. According to this hypothesis, we can say more ELK_N protein present inside a mouse is maybe more stimulated to learn context shock.

**Relationship 5**

Hypothesis = Mice which are stimulated to learn will have more protein present than others, as it is assumed that SHH_N proteins help learn stimulate context shock.



The hypothesis is proven right, as we can see mice stimulating to learn has more median and spread than the one not stimulated to learn. The median and spread of protein present in the mice c-CS-s (Stimulated to learn) is more than a mice c-SC-s (not stimulated to learn), and for c-CS-s (stimulated to learn), median of protein present is more than c-SC-m (not stimulated to learn). Hence, it is the same case for control mice (less protein present for not stimulated to learn. According to this hypothesis, we can say more SYP_N protein present inside a mouse is more stimulated to learn context shock.

### Overall Conclusion

Overall, after exploring the relationships between the columns, I will conclude that the main hypothesis is right, as proteins do help mouse to lean context shock. We explored 10 different types of proteins with the class to identify if proteins help mouse to learn context shock or not. The conclusion was based on this exploration where 4 relationships stated that these types of protein does not help to learn, whereas 5 stated they do help and 1 was nor right or wrong. Based on this numbers, we can conclude that proteins do help learning context shock. To generate a clearer conclusion, we need to more proteins in the dataset, which will further answer the hypothesis question.

### Solution for Clearer Answer

As we only tested for 10 proteins, which were chosen at random, we can test for all the proteins present inside the dataset, as it will be a more correct answer. Here, if different proteins were chosen, maybe we could have a different conclusion, for such type of things, we always need to test everything.

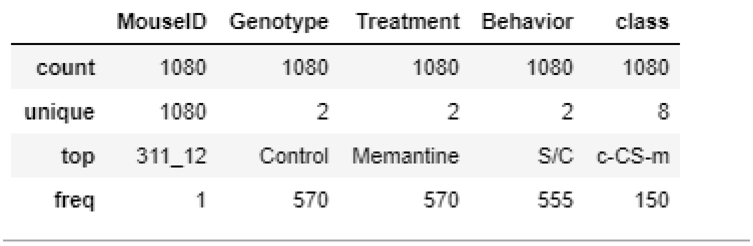
## Data Modelling

‘MouseID’ was dropped as it was not needed for data modelling.

### Model 1 Feature Engineering

-There were 77 features identified, column [0,77]. (All the proteins)

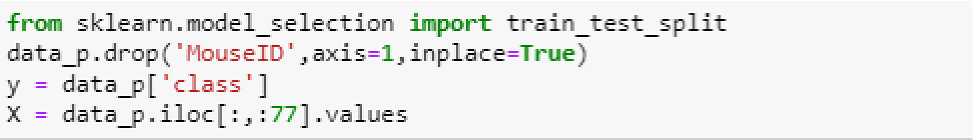
-Four Target Columns identified: Genotype, Class, Treatment and Behavior. (Four classifiers were made to include all targets)



| | MouseID | Genotype | Treatment | Behavior | class |
|---|---|---|---|---|---|
| count | 1080 | 1080 | 1080 | 1080 | 1080 |
| unique | 1080 | 2 | 2 | 2 | 8 |
| top | 311_12 | Control | Memantine | S/C | c-CS-m |
| freq | 1 | 570 | 570 | 555 | 150 |

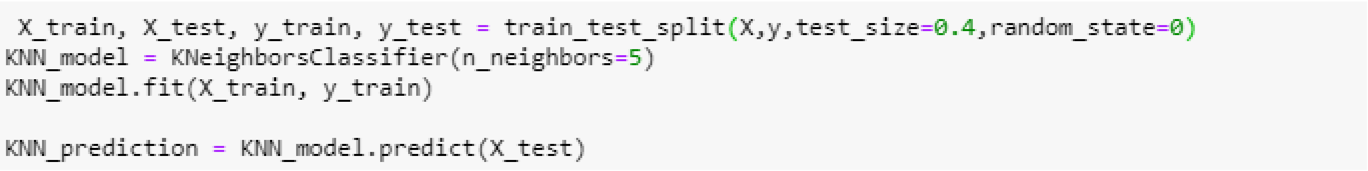
-Model 1 Algorithm: K-neighbours Classifier

For Model 1, 4 classifiers were created of the same algorithm, as there were 4 different targets.



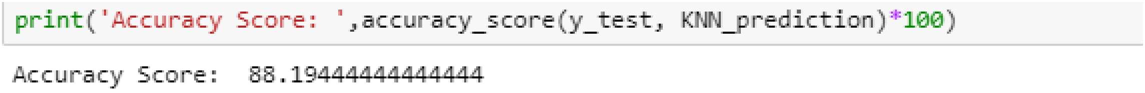
```
from sklearn.model_selection import train_test_split
data_p.drop('MouseID',axis=1,inplace=True)
y = data_p['class']
X = data_p.iloc[:,:77].values
```

**Model 1 Training 1:**



```
 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.4,random_state=0)
KNN_model = KNeighborsClassifier(n_neighbors=5)
KNN_model.fit(X_train, y_train)

KNN_prediction = KNN_model.predict(X_test)
```

Default parameters were used, and the accuracy was scored was accounted to be 89%.

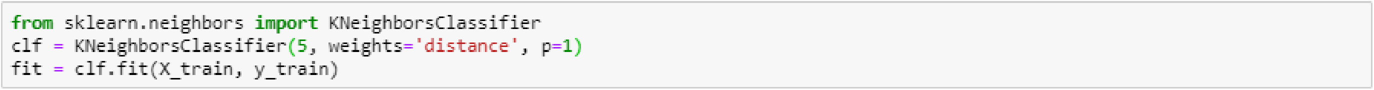


```
print('Accuracy Score: ',accuracy_score(y_test, KNN_prediction)*100)
```

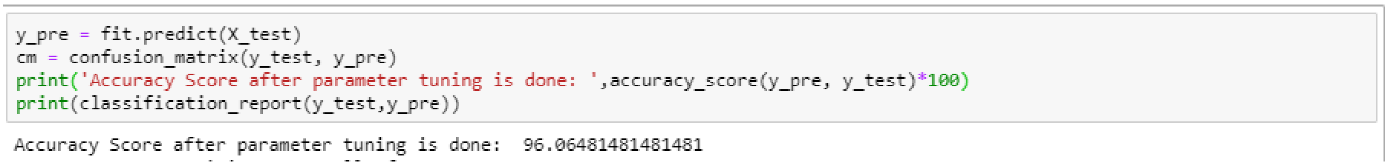
```
Accuracy Score:  88.19444444444444
```

**Model 1 Training 2:**

This time training was done by using other parameters weights and p=1, which enhanced the accuracy score to 96%.



```
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(5, weights='distance', p=1)
fit = clf.fit(X_train, y_train)
```



```
y_pre = fit.predict(X_test)
cm = confusion_matrix(y_test, y_pre)
print('Accuracy Score after parameter tuning is done: ',accuracy_score(y_pre, y_test)*100)
print(classification_report(y_test,y_pre))
```

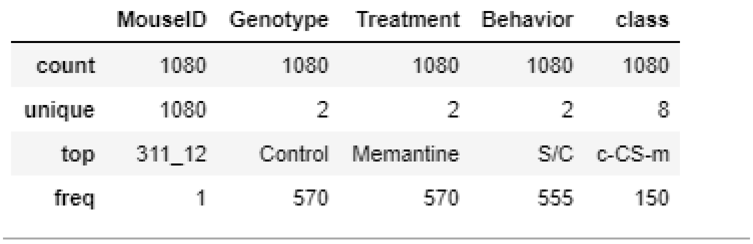
```
Accuracy Score after parameter tuning is done:  96.06481481481481
```

As we already noted that the accuracy was increased by using such parameters, the rest of the classifiers for model one was trained with these parameters from the start.

**Model 2 Feature Engineering:**

-There were 77 features identified, column [0,77]. (All the proteins)

-Four Target Columns identified: Genotype, Class, Treatment and Behavior. (Four classifiers were made to include all targets)

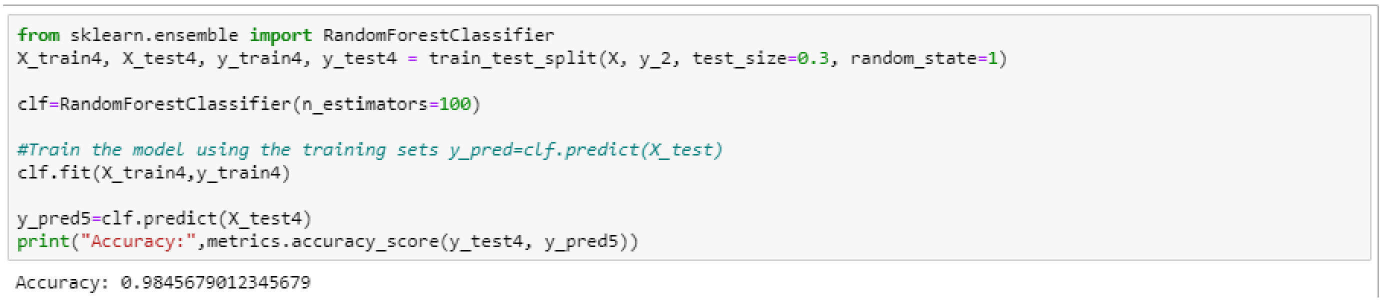


| | MouseID | Genotype | Treatment | Behavior | class |
|---|---|---|---|---|---|
| count | 1080 | 1080 | 1080 | 1080 | 1080 |
| unique | 1080 | 2 | 2 | 2 | 8 |
| top | 311_12 | Control | Memantine | S/C | c-CS-m |
| freq | 1 | 570 | 570 | 555 | 150 |

-Model 2 Algorithm: Random Forest.

For Model 2, 4 classifiers were created of the same algorithm, as there were 4 different targets.

### Model 2 Training 1:

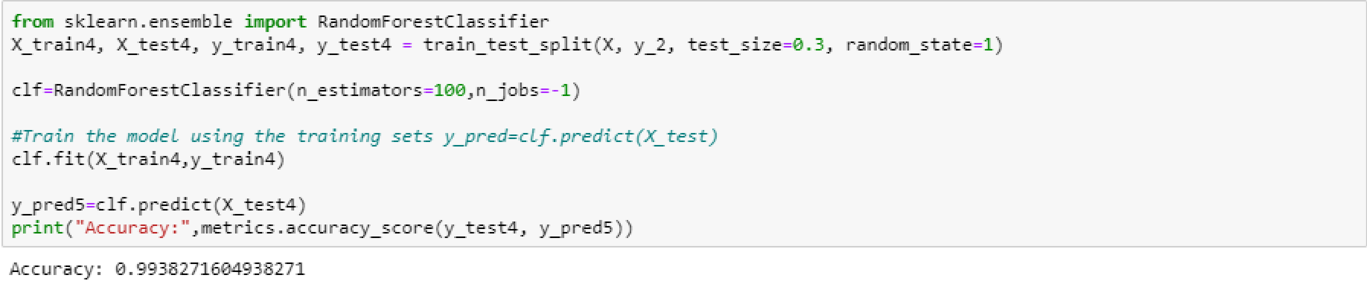


```
from sklearn.ensemble import RandomForestClassifier
X_train4, X_test4, y_train4, y_test4 = train_test_split(X, y_2, test_size=0.3, random_state=1)

clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train4,y_train4)

y_pred5=clf.predict(X_test4)
print("Accuracy:",metrics.accuracy_score(y_test4, y_pred5))
```

```
Accuracy: 0.9845679012345679
```

Default Random Forest parameters were used, and the accuracy was scored was accounted to be 98%.

### Model 2 Training 2:



```
from sklearn.ensemble import RandomForestClassifier
X_train4, X_test4, y_train4, y_test4 = train_test_split(X, y_2, test_size=0.3, random_state=1)

clf=RandomForestClassifier(n_estimators=100,n_jobs=-1)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train4,y_train4)

y_pred5=clf.predict(X_test4)
print("Accuracy:",metrics.accuracy_score(y_test4, y_pred5))
```

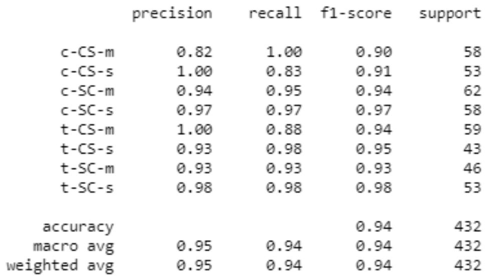
```
Accuracy: 0.9938271604938271
```

This time another parameter was added, ‘n_jobs=-1’, which further enhanced the accuracy score.

As we already noted that the accuracy was increased by using such parameters, the rest of the classifiers for model one was trained with these parameters from the start.

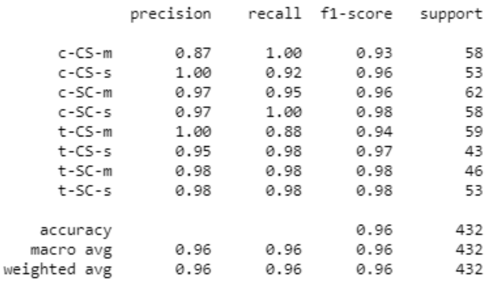
### Results:

For Model 1 During Training 1 Class classifier:



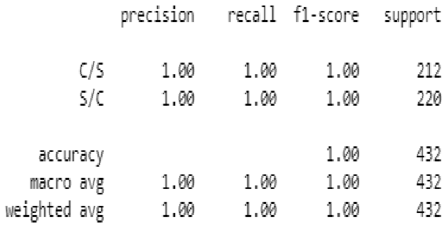
```
              precision    recall  f1-score   support

      c-CS-m       0.82      1.00      0.90        58
      c-CS-s       1.00      0.83      0.91        53
      c-SC-m       0.94      0.95      0.94        62
      c-SC-s       0.97      0.97      0.97        58
      t-CS-m       1.00      0.88      0.94        59
      t-CS-s       0.93      0.98      0.95        43
      t-SC-m       0.93      0.93      0.93        46
      t-SC-s       0.98      0.98      0.98        53

    accuracy                           0.94       432
   macro avg       0.95      0.94      0.94       432
weighted avg       0.95      0.94      0.94       432
```

For Model 1 During Training 2 Class classifier:



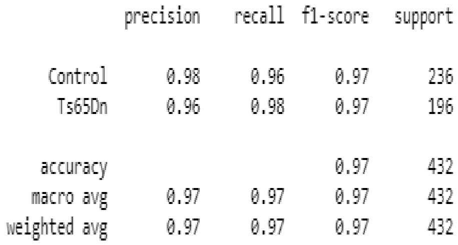
```
              precision    recall  f1-score   support

      c-CS-m       0.87      1.00      0.93        58
      c-CS-s       1.00      0.92      0.96        53
      c-SC-m       0.97      0.95      0.96        62
      c-SC-s       0.97      1.00      0.98        58
      t-CS-m       1.00      0.88      0.94        59
      t-CS-s       0.95      0.98      0.97        43
      t-SC-m       0.98      0.98      0.98        46
      t-SC-s       0.98      0.98      0.98        53

    accuracy                           0.96       432
   macro avg       0.96      0.96      0.96       432
weighted avg       0.96      0.96      0.96       432
```

Behavior:



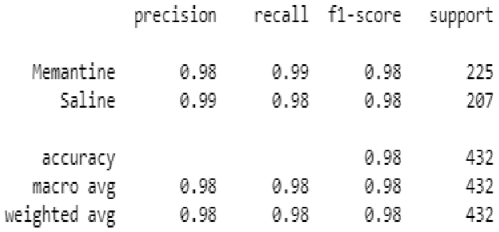
```
              precision    recall  f1-score   support

         C/S       1.00      1.00      1.00       212
         S/C       1.00      1.00      1.00       220

    accuracy                           1.00       432
   macro avg       1.00      1.00      1.00       432
weighted avg       1.00      1.00      1.00       432
```

Genotype:



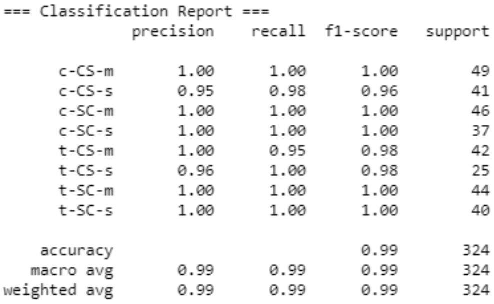
```
              precision    recall  f1-score   support

     Control       0.98      0.96      0.97       236
      Ts65Dn       0.96      0.98      0.97       196

    accuracy                           0.97       432
   macro avg       0.97      0.97      0.97       432
weighted avg       0.97      0.97      0.97       432
```

Treatment:



```
              precision    recall  f1-score   support

   Memantine       0.98      0.99      0.98       225
      Saline       0.99      0.98      0.98       207

    accuracy                           0.98       432
   macro avg       0.98      0.98      0.98       432
weighted avg       0.98      0.98      0.98       432
```

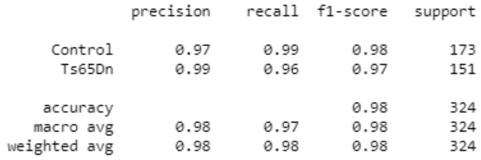
For Model 2 During Training 1 Class Classifier:



```
=== Classification Report ===
              precision    recall  f1-score   support

      c-CS-m       1.00      1.00      1.00        49
      c-CS-s       0.95      0.98      0.96        41
      c-SC-m       1.00      1.00      1.00        46
      c-SC-s       1.00      1.00      1.00        37
      t-CS-m       1.00      0.95      0.98        42
      t-CS-s       0.96      1.00      0.98        25
      t-SC-m       1.00      1.00      1.00        44
      t-SC-s       1.00      1.00      1.00        40

    accuracy                           0.99       324
   macro avg       0.99      0.99      0.99       324
weighted avg       0.99      0.99      0.99       324
```

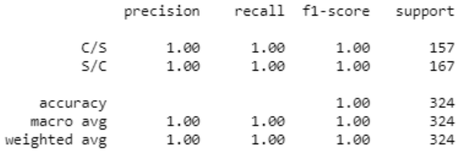
The classification report during both trainings were approximate the same, as there was not much difference between accuracy scores as well.

Genotype:



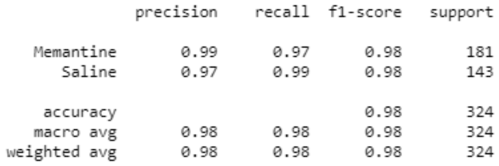
```
              precision    recall  f1-score   support

     Control       0.97      0.99      0.98       173
      Ts65Dn       0.99      0.96      0.97       151

    accuracy                           0.98       324
   macro avg       0.98      0.97      0.98       324
weighted avg       0.98      0.98      0.98       324
```

Behavior:



```
              precision    recall  f1-score   support

         C/S       1.00      1.00      1.00       157
         S/C       1.00      1.00      1.00       167

    accuracy                           1.00       324
   macro avg       1.00      1.00      1.00       324
weighted avg       1.00      1.00      1.00       324
```

Treatment:



```
              precision    recall  f1-score   support

   Memantine       0.99      0.97      0.98       181
      Saline       0.97      0.99      0.98       143

    accuracy                           0.98       324
   macro avg       0.98      0.98      0.98       324
weighted avg       0.98      0.98      0.98       324
```

## Discussion

The making of two data models and analysis were performed successfully. The data cleaning was done with appropriate methods and was successfully prepared to be explored. The exploration was performed and the relationship between the columns were found. Hence, the two models were created. Model 1 used K-neighbour classification whereas the model 2 used Random-Forest. When the models were first build the accuracies were reasonable, but to make them more accurate and higher parameter tuning was performed. After parameter tunning both accuracies generated were very high. Initially, the accuracies were in the high 80 %. Parameters tuning was performed and therefore accuracies were even higher.

The model 1 had an accuracy for the class classifier of 96% whereas, model 2 had an accuracy of 99%. The treatment classifier had an identical accuracy with both models. The model 2 accuracy for genotype classifier was more (97.5%) than the model 1, which was 96%. The behavior classifier had almost same accuracy for both models, as model one was 99%, which was approximately one, whereas model 2 had an accuracy of exactly 1.

For Model 1 K-neighbour algorithm was used and the parameters were searched that were most suitable for the model.

Model 1 parameters:

-weights=distance (weight matrices are based on the centroid distances)

-p=1  (This is equivalent to using manhattan_distance (l1))

The classifier was checked on both, this provided the highest accuracy, so both were selected as the best parameters for this dataset.

For Model 2 Random-Forest algorithm was used and the parameters were searched that were the most suitable for the model.

Model 2 parameters:

-n_jobs=-1

The classifier was tested with the parameter and it provided a better accuracy.

For both Model 1 and Model 2 after parameter tunning the improvement in the accuracies was significant. Both models had very similar accuracies, as model 1 had accuracy around 96-99%, while the other one had accuracy around 97-100%.

Both models had similar accuracies but model 2 provides slightly better accuracy than model 1. There are many other reasons that model 2 is better such as it can handle high dimensional spaces as well as large number of training examples. Hence, there is still room for improvements:

1. We could have created a visualization which will help understand the model better.
2. Use all parameters to get the best possible accuracy, but that will take time.
3. Train data with oversampling.

## Conclusion:

To conclude, it can be said that the dataset shows the potential of machine learning and how proteins can help mouse to learn context shock. We can create models that will help us learn to identify subsets of proteins that are discriminant between the classes.

Overall, the data modelling and whole experiment was successful. The data preparation was very smoothly done, but there could have been a way to get rid of the outliers, which would have made the dataset much cleaner. The data was explored, and relationships were formed between the columns. Although, for data modelling the accuracies generated were reasonable, but to get even high results parameter tuning was performed, and after parameter tuning the difference between the accuracies was significant. Therefore, it can be concluded that no matter how high the accuracy is, the model can always be improved.