Mice Protein Expression Data Set

REPORT

Table of Contents:

Introduction:

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.

Executive Summary:

Data Source: https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression

Data Description:

The data set consists of the expression levels of 77 proteins/protein modifications that produced detectable signals in the nuclear fraction of cortex. There are 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.

The eight classes of mice are described based on features such as genotype, behavior and treatment. According to genotype, mice can be control or trisomic. According to behavior, some mice have been stimulated to learn (context-shock) and others have not (shock-context) and in order to assess the effect of the drug memantine in recovering the ability to learn in trisomic mice, some mice have been injected with the drug and others have not.

Classes:

* c-CS-s: control mice, stimulated to learn, injected with saline (9 mice)
* c-CS-m: control mice, stimulated to learn, injected with memantine (10 mice)
* c-SC-s: control mice, not stimulated to learn, injected with saline (9 mice)
* c-SC-m: control mice, not stimulated to learn, injected with memantine (10 mice)
* t-CS-s: trisomy mice, stimulated to learn, injected with saline (7 mice)
* t-CS-m: trisomy mice, stimulated to learn, injected with memantine (9 mice)
* t-SC-s: trisomy mice, not stimulated to learn, injected with saline (9 mice)
* t-SC-m: trisomy mice, not stimulated to learn, injected with memantine (9 mice)

Attribute Information:

1. Mouse ID
2. 78 Values of expression levels of 77 proteins; the names of proteins indicating that they were measured in the

nuclear fraction. For example: DYRK1A_n
79. Genotype: control (c) or trisomy (t)
80. Treatment type: memantine (m) or saline (s)
81. Behavior: context-shock (CS) or shock-context (SC)
82 .Class: c-CS-s, c-CS-m, c-SC-s, c-SC-m, t-CS-s, t-CS-m, t-SC-s, t-SC-m

Target Feature:
In this project target feature is "class" column which has 8 classes of mice.

Project Objective:

* Our goal is to predict the class of the mice out of 8 characters in the class column.
* The objective of this case study is to fit and compare two different classifiers to predict the class of the mice.
Methodology

For this classification problem following two classifiers are considered to predict the target feature.

- K-Nearest Neighbours (KNN)
- Decision trees (DT)

Data Preparation:

Dataset is usually in raw format so first we are processing dataset to remove any inconsistencies in the data. Data types of all the features will be checked for the correctness of data. Next, we are checking for the missing values and outliers in dataset. Unique values of categorical features will be checked for any typos, extra whitespaces. Range of numerical features will be checked to perform sanity check and identify any inconsistencies in data.

Data Exploration:

Once data is processed then in data exploration part will try to get some meaningful insights from the data. In this part one variable, two variable plots are done to understand the data and how descriptive features are related to target feature. For plotting categorical features bar charts are used, for categorical and numerical pair boxplot is used to obtain relationship between features.

Data Modelling:

In this part first the data is transformed using encoding and scaling techniques. target feature is encoded manually from 0 to 7 for each class since this is multilevel classification problem. To make sure that all the features are in same range they are normalized using Min Max Scaler.

To compare different model performances, hold out sampling method is used in which we split the dataset into training and test sets with a 60:40 ratio.

Feature Selection using Hill Climbing:
In this part we will obtain the maximum number of features which will give the more score with respect to target feature.

Hyperparameter Tuning & Model Fitting:
Before selecting any algorithm, it is necessary to tune the hyperparameters from which the optimal values for hyperparameters will be obtained. To remove any bias from the dataset we are using 5-fold stratified cross-validation technique to tune the hyperparameters.

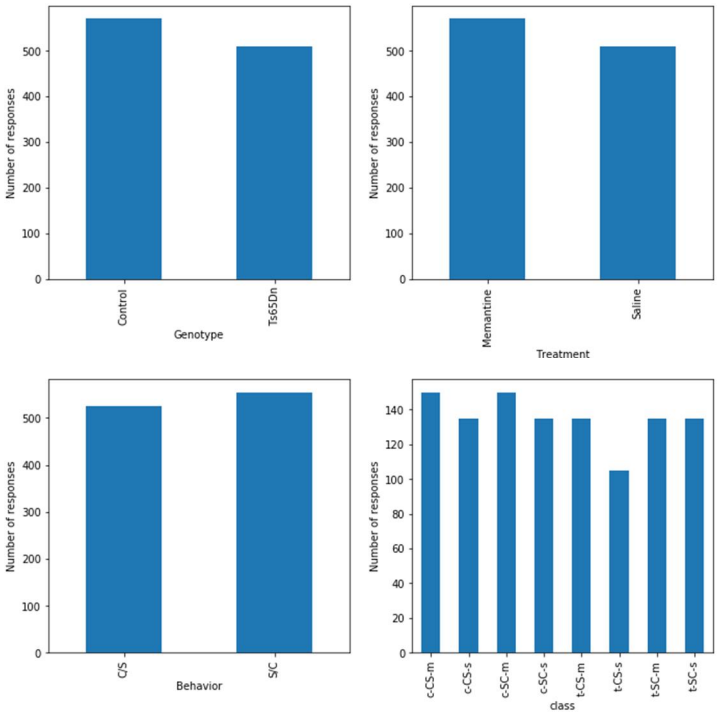
Task 1: Retrieving and Preparing the Data

- **Data_Cortex_Nuclear.csv** file is imported from the UCL machine learning repository for the mice protein expression using the function **read_csv()**. It is separated by "," as it is comma separated value file.
- It has total of 82 Columns and 1080 Rows in the entire dataset.
- After the data is loaded, now displaying the data to verify whether the loaded data is equivalent to the data in the source file.
- Later the data types of all attributes are obtained using the function dtypes().
- Once the data loaded is verified by observing the loaded data frame we proceed further.
- Removing the ID like Columns which is "MouseID" in the data set as it is not useful for our analysis.
- Now checking the number of missing values in each column by using the function isna().sum().
- From the above step we found that there are many missing values in the original data set.
- So, for all the missing values we are replacing it with the `**mean**` of that corresponding column to the data frame for the further analysis.
- Once all the missing values are replaced, further inspection of data is done for any other errors, but everything is proper now, So data is ready for exploration.

Task 2.1: Explore each column

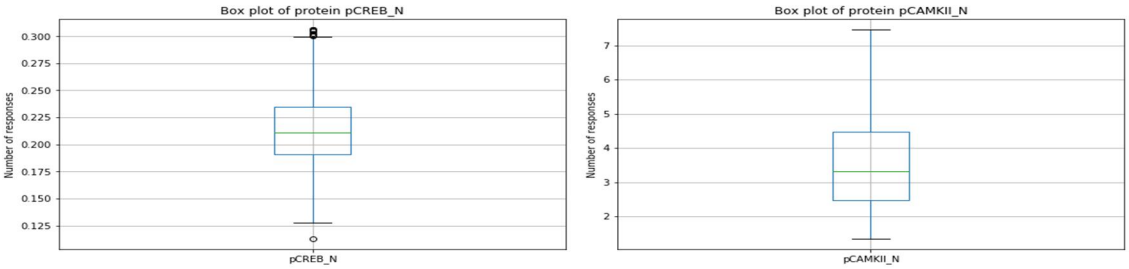
Here each column is explored to understand the column values and their relationships

The following figures of each columns shows that,
* Genotype column has more control mice than trimosy mice. Similarly, memantine treatment is more than saline.
* Shock contest mice are more than contest shock.
* Column class have different numbers with c-CS-m and c-SC-m are more than all other classes.



Protein `pCREB_N` has median of 0.215 and it is equally distrbuted values. Upper and lower portion of box plot shows that it has both low and high values.

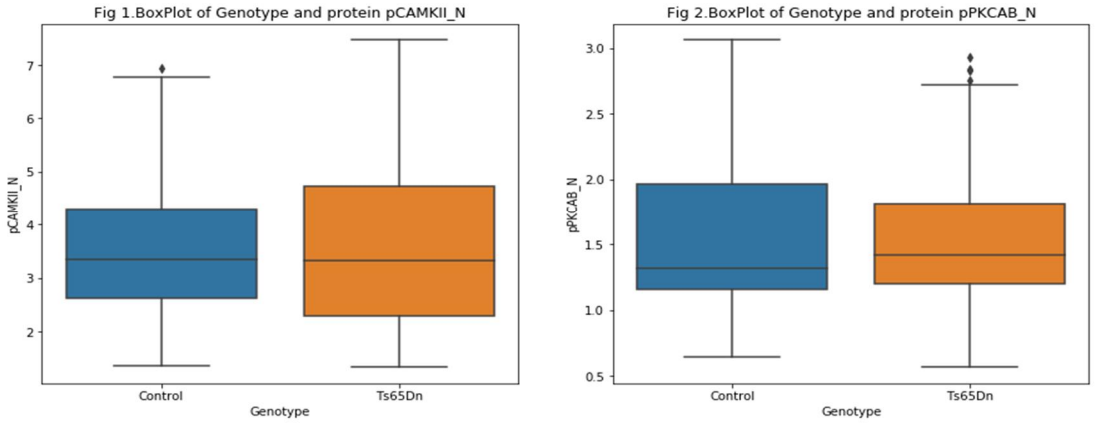
Protein `pCAMKII_N` has median of 3.2 .It has more higher values than the lower values and there are no outliers in this protein. All values lie in the range.



Task 2.2: Explore the relationship between pairs of Attributes

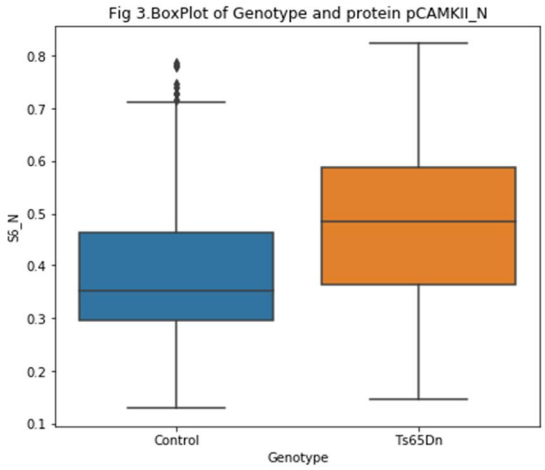
* **Hypothesis-1**: In fig 1, Genotype of control mice median is more than trisomy mice for the protein pCAMKII_N . From the figure we cannot get the evidence as they seem to be equal. So, we reject the hypothesis.

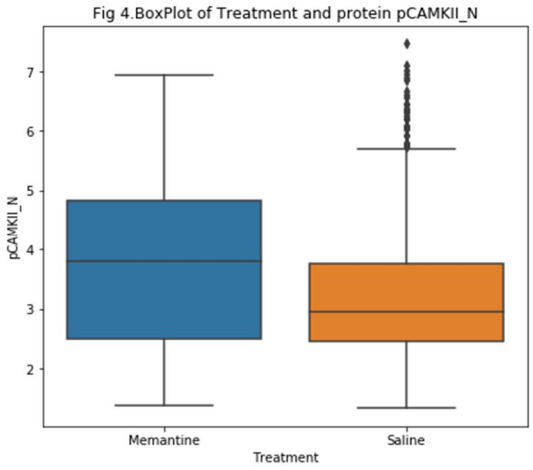
* **Hypothesis-2**: In fig 2, Genotype value of control mice more than Trisomy mice for protein pPKCAB_N. From the figure 2 we accept the hypothesis.



* **Hypothesis-3**: In fig 3, Genotype of value of control mice is less than trisomy mice for the protein S6_N. From the figure we can see that it is true hence we accept the hypothesis.

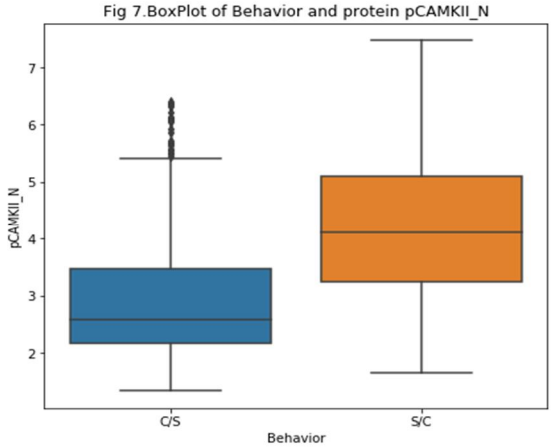
* **Hypothesis-4**: In fig 4, Treatment by Memantime value is less than the saline for the protein pCAMKII_N. From the figure we can see that it is not true hence we reject the hypothesis.

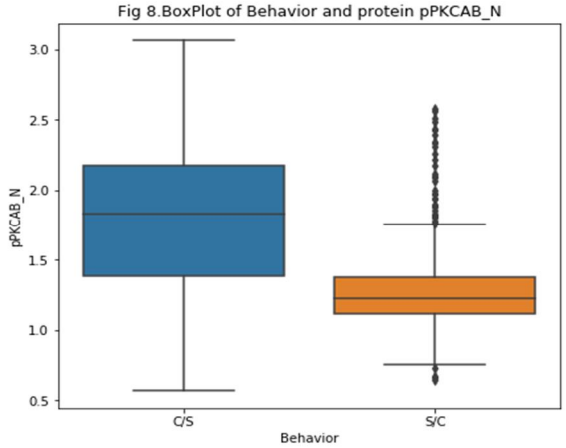




* **Hypothesis-7**: In fig 7, Behavior of context-shock mice value is less than the shock-context mice as the protein pCAMKII_N increases. This is true from the bar graph, so we accept the hypothesis.

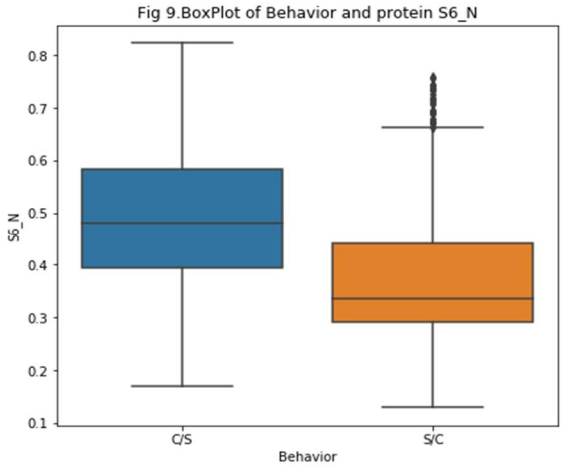
* **Hypothesis-8**: In fig 8, Behavior of context-shock mice is more than the shock-context mice as the protein pPKCAB_N increases. This is also true from the graph, so we accept the hypothesis.

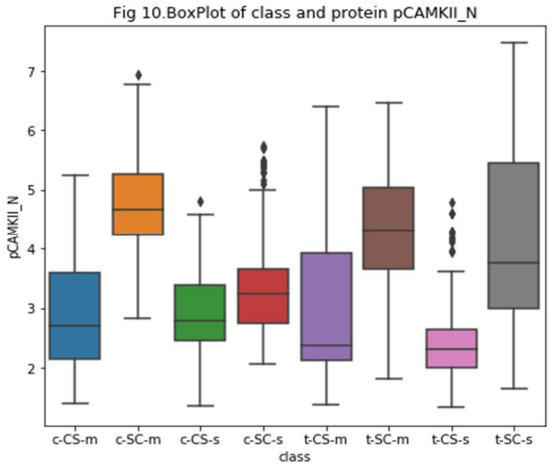




* **Hypothesis-9**: In fig 9, Behavior of context-shock mice median is more than the shock-context as the protein S6_N increases. It is true from the graph and we accept the hypothesis.

* **Hypothesis-10**: In fig 10, trisomy mice, stimulated to learn, injected with saline is more than any other classes as the protein pCAMKII_N increases.





After exploration of all the columns and finding few relationships between the columns now we proceed further of data preparation.

Now the columns ‘Genotype', 'Treatment', 'Behavior’ been removed from the dataset as they are redundant. All these columns combined to form the class column, so it is not required for our analysis. These columns are dropped from the dataset for further analysis.

After removing all missing values, redundant columns, making sanity checks now the column ‘Class’ has 8 unique values which have to be encoded.

c-CS-m': 0, 'c-SC-m': 1,
c-CS-s': 2, 'c-SC-s': 3,
t-CS-m': 4, 't-SC-m': 5,
t-CS-s': 6, 't-SC-s': 7

All 8 class levels are encoded manually as shown above. Now all the dataset is pre-processed and ready for further analysis. Once everything is ready now the target variable i.e. “class” column in this data set is separated from the data set and assigned to target variable and rest of the data set is assigned to variable “data”.
To make the data uniform for our analysis we are scaling the data set using MinMax scaler so that the values vary between 0 and 1 for all the columns.
Now all the pre-processing steps are done, and dataset is ready for modelling.

Task 3: Data Modelling

Splitting data into Train and Test data.

Now the entire data set both data and target variable are used for modelling.

The data and target are splitted into train and test data for training the model and testing the trained model. We used test_size =0.4 which means 40% of entire data is made as test data and rest 60% is made as training data.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(Data,target,test_size=0.4,random_state=0)
```

X_train, y_train consists of Data variables and X_test, y_test consists of target variables. They are splitted in the ration of 60:40 for training and testing, respectively.
Since the target variable has 8 values to predict this is a classification problem. So, we use two classifiers KNN and Decision Tree for this problem to classify the target variable.

K Nearest Neighbour:

```
clf = KNeighborsClassifier()
fit = clf.fit(X_train, y_train)

y_pre = fit.predict(X_test)
```

KNN with default parameter has 5 neighbours. Classifier is fitted with the training data and training target. Then it is fitted with the test data to get the confusion matrix as shown below.

```
cm = confusion_matrix(y_test, y_pre)
print('Confusion Matrix:\n',cm)
```

```
Confusion Matrix:
 [[58  0  0  0  0  0  0  0]
 [ 2 48  0  0  0  3  0  0]
 [ 0  0 60  2  0  0  0  0]
 [ 0  0  3 54  0  0  1  0]
 [ 9  0  0  0 49  0  0  1]
 [ 2  0  0  0  1 40  0  0]
 [ 0  0  2  0  0  0 44  0]
 [ 0  0  1  0  0  0  0 52]]
```

```
print('Classification Report:\n\n',classification_report(y_test,y_pre))
```

```
Classification Report:

              precision    recall  f1-score   support

      c-CS-m       0.82      1.00      0.90        58
      c-CS-s       1.00      0.91      0.95        53
      c-SC-m       0.91      0.97      0.94        62
      c-SC-s       0.96      0.93      0.95        58
      t-CS-m       0.98      0.83      0.90        59
      t-CS-s       0.93      0.93      0.93        43
      t-SC-m       0.98      0.96      0.97        46
      t-SC-s       0.98      0.98      0.98        53

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

The classification report states that accuracy is 0.94 i.e. is 94% which is very good for default parameters. We can also see the parameters for each target variable for our analysis.

Tuning the KNN Model:

Initially we start with 1 neighbour with `weights = uniform`. we are achieving 98% of accuracy. So, we increased the neighbours to 2 and weights as distance, now we are achieving 99% accuracy.
For 2 neighbours with weights as `uniform` and `distance` we are almost getting the same accuracy so we are using Neighbours=2 and weights=distance

```
clf1 = KNeighborsClassifier(2, weights='distance')
fit = clf1.fit(X_train, y_train)
```

The confusion matrix for above parameters is as shown below:

```
cm = confusion_matrix(y_test, y_pre)
print('Confusion Matrix:\n',cm)
```

```
Confusion Matrix:
 [[58  0  0  0  0  0  0  0]
 [ 1 52  0  0  0  0  0  0]
 [ 0  0 61  1  0  0  0  0]
 [ 0  0  0 58  0  0  0  0]
 [ 1  1  0  0 56  1  0  0]
 [ 0  0  0  0  0 43  0  0]
 [ 0  0  0  0  0  0 46  0]
 [ 0  0  0  0  0  0  0 53]]
```

The confusion matrix for KNN with parameters tuned has the best results compared to default KNN values. Here the prediction accuracy increased, i.e. the diagonal elements number is increased than the default KNN model. Now let us see the classification report for the exact numbers for each class variable.

```
print(classification_report(y_test,y_pre))
```

	precision	recall	f1-score	support
c-CS-m	0.97	1.00	0.98	58
c-CS-s	0.98	0.98	0.98	53
c-SC-m	1.00	0.98	0.99	62
c-SC-s	0.98	1.00	0.99	58
t-CS-m	1.00	0.95	0.97	59
t-CS-s	0.98	1.00	0.99	43
t-SC-m	1.00	1.00	1.00	46
t-SC-s	1.00	1.00	1.00	53
accuracy			0.99	432
macro avg	0.99	0.99	0.99	432
weighted avg	0.99	0.99	0.99	432

The classification report shows that accuracy is 99% which is extremely good for any model. We can also see that for few class variables we are getting **recall** as 1 that is 100%. Only few class labels has few error in prediction which we can further analyse how to reduce the error by other methods.

Feature Selection by Hill Climbing method for KNN

Loading the shuffle package to randomize the order of features, then we start trying from the 1st feature. Then, we will keep this feature as selected if this feature existing already selected features can lead to higher accuracy score than that when only using existing already selected features.
Here we are selecting the best features for the KNN algorithm in the entire dataset using the Hill Climbing technique.
The best features are selected based on the score until the maximum score is obtained for the number of columns. once the maximum possible score is obtained then those features indexes are obtained in the list from which we form one dataset and it is used to train the model for best score with less features is always an advantage.

After applying the hill Climbing method for feature selection, we got combination of 33 features which is giving maximum score.

```
print("There are " + str(len(new_Ind)) + " features selected:")
print([new_Ind])
```

```
There are 33 features selected:
[[31, 43, 26, 74, 58, 59, 61, 51, 36, 57, 10, 53, 34, 71, 46, 15, 45, 19, 76, 63, 62, 55, 67, 68, 39, 69, 73, 66, 28, 42, 60, 2
0, 12]]
```

Now with these Indices we get the corresponding columns from the data set and new dataset was made with only 33 features i.e. “data_fea”.

Fitting the KNN default model for feature selected dataset.

Now again fitting the Default and tuned KNN models for this feature selected dataset to check the efficiency. For this fitting again train and test data are separated with the “data_fea” dataset with same ratio of 40% test data. Fitting for default Model with 33 features selected data. Now after fitting the model when checking the classification report we found that it is giving 94% efficiency which is very good.

```
print(classification_report(y_test1,y_pre1))
```

	precision	recall	f1-score	support
c-CS-m	0.89	0.98	0.93	58
c-CS-s	0.96	0.94	0.95	53
c-SC-m	0.87	0.95	0.91	62
c-SC-s	0.92	0.84	0.88	58
t-CS-m	0.96	0.88	0.92	59
t-CS-s	0.95	0.93	0.94	43
t-SC-m	0.98	0.96	0.97	46
t-SC-s	0.98	1.00	0.99	53
accuracy			0.94	432
macro avg	0.94	0.94	0.94	432
weighted avg	0.94	0.94	0.93	432

Fitting the tuned KNN model for feature selected dataset.

Now after getting 94% for default KNN, now its parameters are tuned to check the accuracy of the model for 33 features selected by Hill Climbing technique. Here KNN with 2 neighbours and weights as distance is giving almost 100% accuracy. This is an excellent accuracy for just 33 features out of 82 features in the entire dataset.

```
print(classification_report(y_test,y_pre))
```

	precision	recall	f1-score	support
c-CS-m	1.00	1.00	1.00	58
c-CS-s	1.00	0.98	0.99	53
c-SC-m	1.00	0.98	0.99	62
c-SC-s	0.98	1.00	0.99	58
t-CS-m	0.98	1.00	0.99	59
t-CS-s	1.00	1.00	1.00	43
t-SC-m	1.00	1.00	1.00	46
t-SC-s	1.00	1.00	1.00	53
accuracy			1.00	432
macro avg	1.00	1.00	1.00	432
weighted avg	1.00	1.00	1.00	432

Decision Tree

Now Decision tree classifier is used to predict the target. Here the same train and test data is used which was used for the default KNN classifier without feature selection.
Now train data is fitted to the model with the default parameters of the decision tree. After fitting the training data, now model is used to predict the target. Now from the confusion matrix we can observe that diagonal values are much higher than others in that rows, which means that it is performing well. After getting the classification report we can see that it is giving the accuracy of 78% which is not bad.

```
cm = confusion_matrix(y_test, y_pre)
print('Confusion Matrix:\n',cm)
print(classification_report(y_test,y_pre))
```

```
Confusion Matrix:
 [[48  7  0  0  1  2  0  0]
 [ 4 36  5  0  3  4  0  1]
 [ 1  0 53  0  0  0  7  1]
 [ 0  1  2 47  0  0  7  1]
 [ 4  3  0  0 51  1  0  0]
 [ 2  6  0  0  4 30  1  0]
 [ 2  2  7  7  0  0 27  1]
 [ 3  2  3  1  0  0  0 44]]
              precision    recall  f1-score   support

       c-CS-m       0.75      0.83      0.79        58
       c-CS-s       0.63      0.68      0.65        53
       c-SC-m       0.76      0.85      0.80        62
       c-SC-s       0.85      0.81      0.83        58
       t-CS-m       0.86      0.86      0.86        59
       t-CS-s       0.81      0.70      0.75        43
       t-SC-m       0.64      0.59      0.61        46
       t-SC-s       0.92      0.83      0.87        53

    accuracy                           0.78       432
   macro avg       0.78      0.77      0.77       432
weighted avg       0.78      0.78      0.78       432
```

Now the decision tree is tuned with parameters to get the maximum efficiency. Tuning is done by varying the max_depth from 0 to 11 and min_samples_splits from 0 to 10, with criterions as `entropy` and `ginni`. After testing all the combinations, we found that it is giving maximum accuracy for entropy with max_depth=11 and samples_splits=3.

From the classification report we can see that it is giving 81% after tuning.

```
cm = confusion_matrix(y_test, y_pre)
print('Confusion Matrix:\n',cm)
print(classification_report(y_test,y_pre))
```

```
Confusion Matrix:
 [[50  7  0  0  0  0  1  0]
 [ 4 37  1  0  5  4  1  1]
 [ 0  2 52  1  0  0  7  0]
 [ 0  0  2 51  0  0  5  0]
 [10  1  0  0 47  1  0  0]
 [ 5  1  0  0  4 33  0  0]
 [ 1  1  4  3  0  0 36  1]
 [ 2  0  1  3  1  0  1 45]]
              precision    recall  f1-score   support

       c-CS-m       0.69      0.86      0.77        58
       c-CS-s       0.76      0.70      0.73        53
       c-SC-m       0.87      0.84      0.85        62
       c-SC-s       0.88      0.88      0.88        58
       t-CS-m       0.82      0.80      0.81        59
       t-CS-s       0.87      0.77      0.81        43
       t-SC-m       0.71      0.78      0.74        46
       t-SC-s       0.96      0.85      0.90        53

    accuracy                           0.81       432
   macro avg       0.82      0.81      0.81       432
weighted avg       0.82      0.81      0.81       432
```

Feature Selection by Hill Climbing method for Decision tree

Now best features for the decision tree is obtained from the Hill Climbing technique. We can find that Hill climbing technique choosed 27 characters as it is giving maximum score.

```
print("There are " + str(len(new_Ind)) + " features selected:")
print([new_Ind])
```

```
There are 27 features selected:
[[31, 43, 26, 74, 58, 59, 61, 51, 57, 10, 53, 34, 46, 15, 45, 62, 67, 5
2, 22, 41, 4, 60, 18, 50, 1, 16, 64]]
```

After selecting the 27 important features for the decision tree classifier now we are fitting the model with the new 27 features dataset to check the accuracy.

* After fitting the model, we can get the accuracy of the 80% with default decision tree and for tuned decision tree parameters are tuned to get the maximum accuracy for those 27 features obtained from the hill climbing. At max_depth=9, we are getting the maximum accuracy of 83%.

For Default Decision Tree

	precision	recall	f1-score	support
c-CS-m	0.83	0.83	0.83	58
c-CS-s	0.78	0.74	0.76	53
c-SC-m	0.86	0.79	0.82	62
c-SC-s	0.83	0.76	0.79	58
t-CS-m	0.83	0.88	0.85	59
t-CS-s	0.72	0.72	0.72	43
t-SC-m	0.66	0.80	0.73	46
t-SC-s	0.88	0.87	0.88	53
accuracy			0.80	432
macro avg	0.80	0.80	0.80	432
weighted avg	0.81	0.80	0.80	432

For Tuned Decision Tree

	precision	recall	f1-score	support
c-CS-m	0.78	0.86	0.82	58
c-CS-s	0.73	0.68	0.71	53
c-SC-m	0.85	0.89	0.87	62
c-SC-s	0.84	0.90	0.87	58
t-CS-m	0.84	0.83	0.84	59
t-CS-s	0.82	0.77	0.80	43
t-SC-m	0.89	0.87	0.88	46
t-SC-s	0.90	0.83	0.86	53
accuracy			0.83	432
macro avg	0.83	0.83	0.83	432
weighted avg	0.83	0.83	0.83	432

Now after all these results we do 5-fold cross validation for most accurate accuracy for both KNN and Decision tree classifiers.

5-folds cross validation for KNN

```
[fold 0] score: 0.99074
[fold 1] score: 0.99074
[fold 2] score: 1.00000
[fold 3] score: 1.00000
[fold 4] score: 1.00000
```

5-folds cross validation for Decision Tree

```
[fold 0] score: 0.85185
[fold 1] score: 0.83796
[fold 2] score: 0.84722
[fold 3] score: 0.79167
[fold 4] score: 0.75463
```

Table for KNN

Table for Decision Tree

Model	Accuracy
Default KNN	94%
Tuned KNN	99%
Feature Selected Default KNN	94%
Feature Selected	99%

Tuned KNN	

Model	Accuracy
Default Decision Tree	78%
Tuned Decision Tree	81%
Feature Selected Default Decision Tree	80%
Feature Selected Tuned Decision Tree	83%

Discussion:

We have seen that the models performing well for this classification problem which has 8 classes. The KNN model with features of 33 selected from Hill climbing technique giving the accuracy of the prediction 99%. We can further investigate by using different feature selection methods to get the smaller number of features for the same accuracy. Similarly, Decision tree model also can be improved by using different modes of tuning and feature selection models. If we can get the high accuracy of models with less number of features then, we can save the execution time, processing time and memory

Conclusion:

Taking into consideration the performance of both classifiers it can be concluded that KNN is giving the highest accuracy with all 33 features i.e. 99%. It can be noted that Decision Tree with 27 features giving efficiency of 83%. Out of 77 features KNN model is giving highest accuracy considering only 33 features is the best model we can use to classify the class of the 8 different mice.

So, we recommend KNN with feature selection is the best model for this Classification dataset.

References:

- Dr. Yongli Ren, ' week3-Summarisation-Clear-1', PowerPoint slides , COSC2670, RMIT University, viewed 10 June 2020, <https://rmit.instructure.com/courses/67430/files/11273016?module_item_id=2268981>.
- Dr. Yongli Ren, ' week4', PowerPoint slides , COSC2670, RMIT University, viewed 10 June 2020, <https://rmit.instructure.com/courses/67430/files/11273015?module_item_id=2268981>.
- Dr. Yongli Ren, ' week5', PowerPoint slides , COSC2670, RMIT University, viewed 10 June 2020, <https://rmit.instructure.com/courses/67430/files/11273015?module_item_id=2314378>
- matplotlib.org. (2002). matplotlib.pyplot.subplots — Matplotlib 3.2.1 documentation. [online] Available at: https://matplotlib.org/3.2.1/api/_as_gen/matplotlib.pyplot.subplots.html [Accessed 10 Jun. 2020].
- pandas.pydata.org. (n.d.). pandas.DataFrame.unstack — pandas 1.0.3 documentation. [online] Available at https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.unstack.html [Accessed 10 Jun. 2020].