Student ID: S3768288
Student Name: Kathleen Magbual

I certify that this is all my own original work. If I took any parts from elsewhere, then they were non-essential parts of the assignment, and they are clearly attributed in my submission. I will show I agree to this honor code by typing "Yes": *Yes*.

Predictive Modelling of Mice Protein Expression

Kathleen R. Magbual
COSC2760
S3768288@student.rmit.edu.au

June 10, 2020

Table of Contents

Executive Summary

The accurate identification of each of the eight (8) mouse class, namely , “c-CS-m”, “c-SC-m”, “t-SC-s”, “t-CS-m”, “c-SC-s”, “c-CS-s”, “t-SC-m”, and “t-CS-s” classified in terms of its genotype-behavior-treatment are constantly affected by the 77 different protein expressions.

Based on the problem at hand, the objective of this study is to explore, generate insights, and to identify which types of protein have more impact on the learning outcome of each of the mouse class, to further aide in applying an appropriate machine learning classifier to correctly predict each mouse class.

Two machine learning classifiers are tested and compared which outperforms the other with accuracy as the scoring metric. These classifiers are namely K Neighbors classifier and Decision Tree classifier. Each of the classifiers, were tested against a 70/30 proportion with 756 rows of training data were used for hyperparameter tuning phase and the remaining 324 rows of test data were used for the model evaluation.

To achieve the best fitting model, feature selection, cross validation and hyperparameter tuning were applied to the default/baseline classifier. The GridSearchCV splits the data into training set and validation set where the training set is mainly for hyperparameter tuning and validated against the validation set. This was done several times in this study, in both 5 and 10 folds, to avoid overfitting and underfitting and to achieve optimal parameters and accuracy.

Findings suggest that, a cross validated tuned KNN with optimal parameters of n_neighbors = 1 and p = 1 (Manhattan) is the best fitting classifier for the problem, it correctly predicted each mouse class. In addition, the selected 20 out of 77 important protein features by the Random Forest Importance were enough of a contribution to achieve the desired accuracy performance of the tuned KNN model. It is also important to note, cross-validation was applied to both the train and test sets to reduce overfitting.

It generated a higher testing accuracy of 97.2% compared to a Decision Tree classifier with only 77.6% test accuracy. In addition, both its classification report and confusion matrix correctly predicted each of the eight mouse class, it was also observed that its confusion matrix formed a diagonal matrix which means it was able to predict the correct classes. It also important to note the recall score of the tuned Decision Tree scored poorly in some of the mouse classes, thus it did not meet the goal of our objective.

I. Introduction

The identification of how the different types of protein, genotype, treatment, and behavior affect the learning outcome of each mouse is constantly challenged by rapid technological and social changes over the years. The Mice Protein Expression Data set was obtained from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression). The dataset consists of 1,080 observations and 82 features, 77 of those are different types of protein that affect each mouse type class. The other 4 features are genotype, treatment, behavior, and class. The class feature contains eight types of mouse classes based on genotype, wherein a mouse can be control or trisomic/down syndrome. Furthermore, based on treatment, some mice are injected with Memantine or Saline and lastly, based on behavior some mice have been stimulated to learn (context-shock or C/S) and others have not (shock-context or S/C).

Objectives:

The objective of this study is to explore, generate insights, and to identify which types of protein have more impact on the learning outcome, to further aide in applying an appropriate machine learning classifier to correctly predict each mouse class.

II. Methodology

2.1 Data Preparation

The dataset has 1,080 rows and 82 columns. The descriptive features include 77 numeric protein types and target feature "class". Prior feature selection and modelling the following data preparation should be applied:

- All missing values and unusual values were treated accordingly. Missing values were replaced with zero (0) to avoid inaccurate assumption of values instead of imputation of mean.
- Some descriptive features are deemed unnecessary for machine learning prediction such as ID feature "MouseID"
- The 77 numerical protein features were transformed using Min-Max scaling.
- It also to be noted that the dependent categorical features of the target feature, namely, genotype, treatment and behavior were not used in the predictive modelling, as the class feature is the summary of all the said features.

2.2 Predictive Modelling Methodology

It is to be noted that two (2) classifiers were used to predict the target feature

1. K-Nearest Neighbors (KNN)
2. Decision Tree (DT)

It will first proceed to perform a Feature Selection, comparing the results of different wrapper-based and filter-based feature selection methods namely, Hill Climbing Technique, F-score and Random Forest Importance to evaluate and select the most relevant features from the dataset.

Afterwards, a data sample was obtained based from the selected best features, in our case using the Random Feature importance of 20 selected best features. This data sample was split into the data and target features into training and test set following a 70-30 proportion. Cross Validation and Hyperparameter Tuning was performed for the KNN and Decision Tree classifiers. Specifically, cross validation of Stratified 5-KFold and Grid Search were performed to identify and validate the optimal parameters for each classifier with "Accuracy" as the performance metric.

Each of the classifier was fitted with the best set of hyperparameter values on the test data in another cross-validated way to identify which tuned classifier outperforms the others. Both phases involve cross-validation, thus effects of overfitting shall be quite limited.

Lastly, the classifiers were compared with its respective accuracy scores, classification reports specifically recall scores and its confusion matrices.

III. Results

3.1 Descriptive Statistics

To evaluate further on the data, summary statistics for categorical and numeric features (respectively) were identified. The below descriptive summaries aim to provide a better overview of each feature. Based on the summary statistics of categorical features below, the target feature "class" has 8 labels thus this can be classified as a multiclass target feature.

	MouseID	Genotype	Treatment	Behavior	class
count	1080	1080	1080	1080	1080
unique	1080	2	2	2	8
top	3418_10	Control	Memantine	S/C	c-CS-m
freq	1	570	570	555	150

Table 3.1.1: Summary statistics of categorical features

On the other hand, the summary statistics of numerical features show the each of the 77 protein has 1080 measurements for each mouse. It can also be observed that there are no unusual values in the numerical values.

	DYRK1A_N	ITSN1_N	BDNF_N	NR1_N	NR2A_N	pAKT_N	pBRAF_N	pCAMKII_N	pCREB_N	pELK_N	...	SHH_N
count	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	...	1080.000000
mean	0.424627	0.615388	0.318202	2.290688	3.833256	0.232520	0.181341	3.527284	0.211983	1.424713	...	0.226676
std	0.250022	0.253382	0.052098	0.367300	0.953532	0.043351	0.028652	1.306709	0.034413	0.472284	...	0.028989
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.155869
25%	0.287847	0.472764	0.287112	2.056486	3.149320	0.205530	0.164411	2.471432	0.190489	1.202789	...	0.206395
50%	0.366125	0.565494	0.316462	2.295648	3.738908	0.231082	0.182270	3.325505	0.210560	1.355423	...	0.224000
75%	0.487574	0.697500	0.348039	2.528035	4.425107	0.257225	0.197226	4.480652	0.234558	1.560931	...	0.241655
max	2.516367	2.602662	0.497160	3.757641	8.482553	0.539050	0.317066	7.464070	0.306247	6.113347	...	0.358289

Table 3.1.2: Summary statistics of numerical features

3.2 Data Exploration and Visualization of each feature

Each feature of the dataset was explored and visualized to further understand the data. Starting with the categorical features including the target feature --"class". Figure 3.2.1 shows the target feature "class" wherein there are 8 labels namely, "c-CS-m", "c-SC-m", "t-SC-s", "t-CS-m", "c-SC-s", "c-CS-s", "t-SC-m", and "t-CS-s". The proportion of each class are almost equal apart from "t-SC-m". In addition, looking at its dependent variables with graphs of Genotype, Treatment and Behavior also shows an almost equal result.

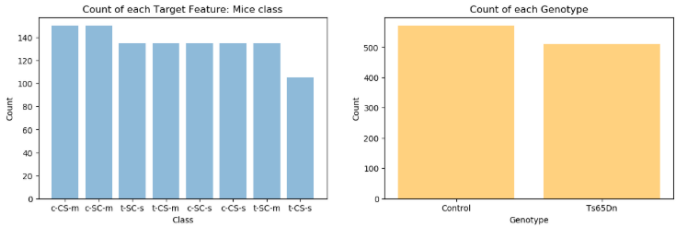


Figure 3.2.1: (L-R) Count bar chart of target feature and Count bar chart of Genotype feature.

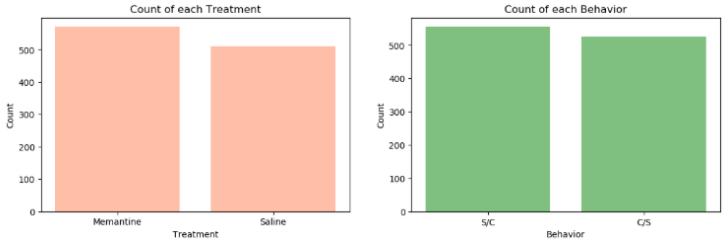


Figure 3.2.2: (L-R) Count bar chart of Treatment feature and Count bar chart of Behavior feature.

Moving on to the numerical features of 77 protein levels, in Figure 3.2.3 below shows the 10 highest mean protein values and on the left hand side is the 10 lowest mean protein values. The ten highest and lowest mean protein values were identified to check if these protein features have an impact on the prediction of target feature in the predictive modelling phase.

It can be observed that based on the mean values, the 10 highest proteins have a frequency range of 2 to 4 protein levels, which can be considered as the biggest protein contributors in each mouse class. Conversely, the lowest mean proteins have a frequency range of 0.0 to 0.02 protein levels which can be considered as a very minimal amount of contribution to the mice class.

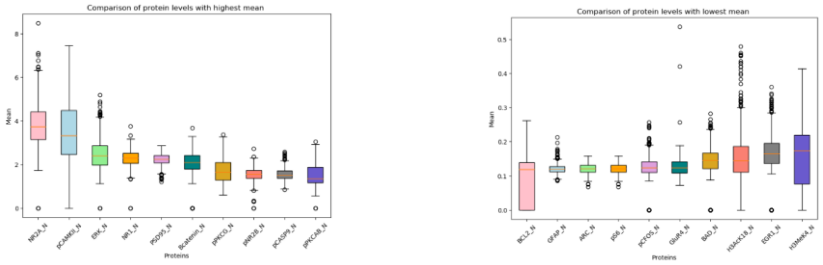


Figure 3.2.3: (L-R) Boxplot chart of the 10 highest mean protein values and Boxplot chart of the 10 lowest mean protein values.

Another comparison is between the 10 highest standard deviation values and 10 lowest standard deviation values which can be seen in Figure 3.2.4 below. 7 out of 10 of the highest standard deviation are also included in the 10 highest mean namely, pCAMKII_N, NR2A_N, ERK_N, pPKCG_N, Bcatenin_N, pPKCAB_N and NR1_N. The remaining three with the most outliers are not considered with high mean values as well. The 7 abovementioned proteins strongly suggest that they may have a big impact on the target feature mice class. This shall be further confirmed during the feature selection of the predictive modelling phase.

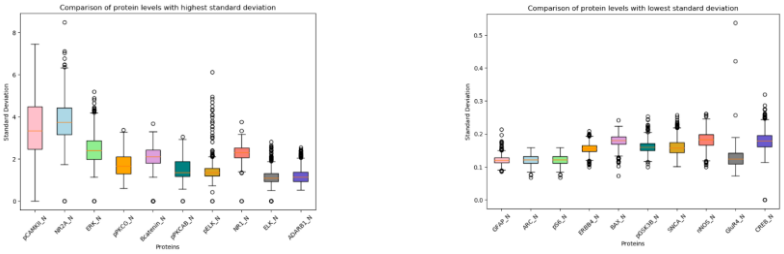


Figure 3.2.4: (L-R) Boxplot chart of the 10 highest standard deviation protein values and Boxplot chart of the 10 lowest standard deviation protein values.

3.3 Data Exploration and Visualization between pairs of features

The pearson correlation of 77 numerical protein features including the target feature was visualised in order to identify its relevance for further modelling. A pearson correlation matrix is shown in Figure 3.3.1 below shows that some features in shades of red are highly correlated as opposed to others with shades of blue.

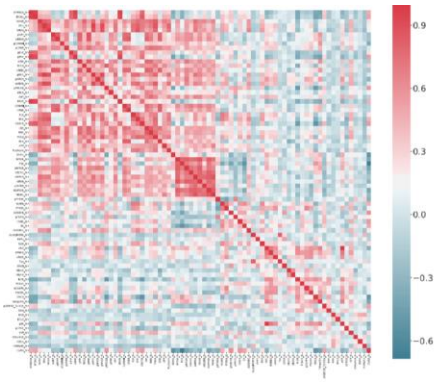


Figure 3.3.1: Pearson Correlation matrix of 77 numerical protein features and the target feature

The correlation coefficient has a range of −1 to +1. The larger the absolute value of the correlation coefficient, the stronger the relationship between two variables. An absolute value of 1 indicates a perfect linear relationship, on the other hand, an absolute value close to 0 indicates no linear relationship between two variables. Table 3.3.1 below shows the 10 highly correlated pairs of protein features and the target feature class, with SOD1_N and class having the highest correlation of 0.381. These highly correlated protein features are assumed to have a big impact in the target feature prediction. It is also important to note that none of the protein features included in the 10 highest mean and standard deviation values were not found as highly correlated. Furthermore, some of the protein features included in the 10 highest mean and standard deviation values such as ERK_N, NR2A_N and pPKCAB_N were considered as uncorrelated with a negative correlation coefficient.

10 highly correlated proteins and class

Dependent Variables: Protein	Independet Variable	Correlation	Is_correlated
SOD1_N	class	0.381	True
Tau_N	class	0.361	True
APP_N	class	0.345	True
H3MeK4_N	class	0.334	True
AcetylH3K9_N	class	0.326	True
pPKCG_N	class	0.299	True
pP70S6_N	class	0.298	True
Ubiquitin_N	class	0.259	True
H3AcK18_N	class	0.247	True
pCREB_N	class	0.227	True

10 uncorrelated proteins and class

Dependent Variables: Protein	Independet Variable	Correlation	Is_correlated
pERK_N	class	-0.263	False
CaNA_N	class	-0.256	False
GluR3_N	class	-0.25	False
pNUMB_N	class	-0.216	False
NR2A_N	class	-0.201	False
ERK_N	class	-0.189	False
GSK3B_N	class	-0.189	False
pPKCAB_N	class	-0.184	False
BRAF_N	class	-0.181	False
SYP_N	class	-0.167	False

Table 3.3.1: 10 highly correlated and uncorrelated variables between protein features and the target feature class.

As seen in Figure 3.3.2 below is scatter plot evidence of a large positive relationship between SOD1_N and class with a correlation coefficient of 0.381. The relationship between pERK_N and class shows otherwise with large negative relationship of -0.263.

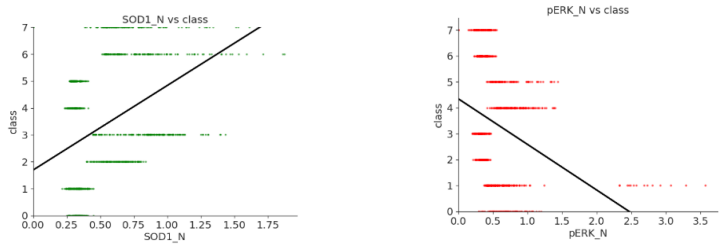


Figure 3.3.2: Scatter plot of the highest correlated pair of SOD1_N and class and Scatter plot of the uncorrelated pair of pERK_N and class.

The plots below depict that the proteins with high correlation coefficient have indeed a positive relationship. On the other hand, for exploratory purposes, it can be observed that a high correlation coefficient protein vs uncorrelated protein have a negative relationship.

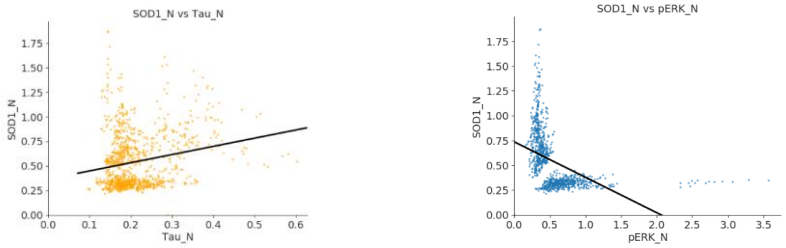


Figure 3.3.3: Scatter plot of highly correlated proteins, SOD1_N and Tau_N and Scatter plot of the correlated protein: SOD1_N and uncorrelated protein: pERK_N

Taking a closer look at the frequency values of highly correlated proteins and the target feature class, SOD1_N and class boxplot show that it has a frequency range of 0.25 to 1, one of the highest values in the group. Another highly correlated protein is Tau_N and class, wherein it shows a low frequency value between 0.2 to 0.3, however, it is important to note that the boxplot shows consistent equal values all throughout the different mice class. It can be observed that the specific proteins with high corrleation with the target feature class, APP_N and H3MeK4, have a low frequency range between 0 to 0.5, but it can also be observed that regardless of lower frequency range, it shows consistent equal contribution all throughout the different mice class.

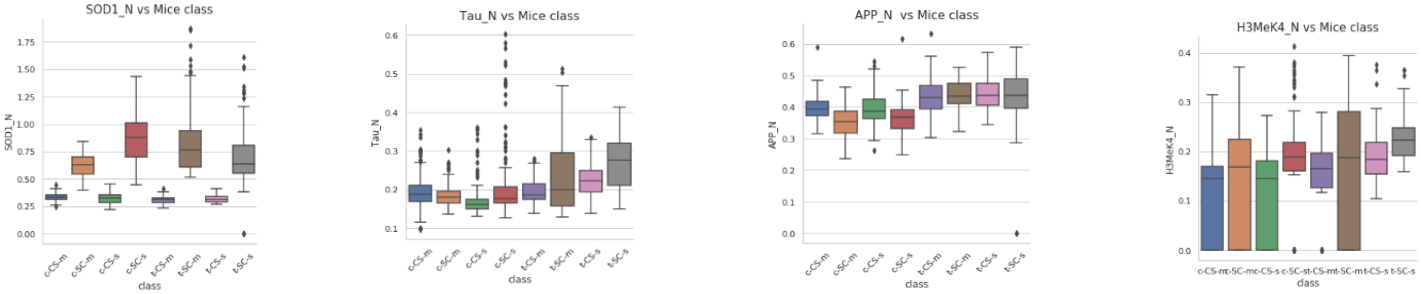


Figure 3.3.4: Boxplot of highly correlated proteins and target feature class

Moving on to the uncorrelated proteins with the target feature class, it can be observed that pERK_N and CaNA_N have high frequency range, however, pERK_N only has minimal contribution, on other hand, CaNA_N has an unequal distribution, thus it was categorized as uncorrelated. GluR3_N and pNUMB_N, both show low frequency range of less than 0.25 for GluR3_N, which is considered a very minimal contribution to the prediction of target feature. pNUMB_N has a range of less than 0.5 and at the same time it can also be observed that it has unequal contribution in each mouse class.

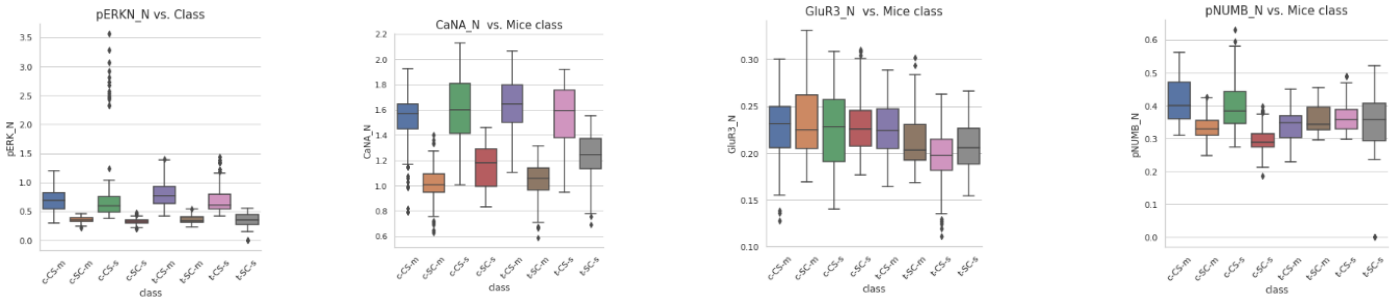


Figure 3.3.5: Boxplot of uncorrelated proteins and target feature class.

3.4 Predictive Modeling: Feature Selection

Before fitting a model, it is good to do feature selection to check whether reducing feature can help achieve better performance. The objective of feature selection is to use a number of features instead of using all features that may lead to poor performance in general. The feature selection methods that are going to be used are Hill Climbing Technique, Random Forest and F-Score methods.

First using a wrapper-based method such as Hill Climbing Technique, it can be observed that the chosen 20 features are not aligned with the abovementioned highly correlated features. Conversely, using filter methods such as Random Forest Importance and F-Score, the 20 most important protein features are selected. It can be observed in Figure 3.4.1 below that the top highly correlated protein SOD1_N is also the most important feature. Surprisingly, there are still features which are uncorrelated but is considered as highly important. The F-Score also shows the 20 most important protein features, again with SOD1_N as the most important feature.

Wrapper-based method

	Best Features Hill Climbing
0	BDNF_N
1	GFAP_N
2	pP70S6_N
3	P70S6_N
4	GluR3_N
5	MTOR_N
6	P38_N
7	pNR2A_N
8	pERK_N
9	Ubiquitin_N
10	pRSK_N
11	DYRK1A_N
12	APP_N
13	CDK5_N
14	PSD95_N
15	AKT_N
16	BAD_N
17	DSCR1_N
18	pPKCG_N
19	pS6_N

	Best Features RFI	Feature importances
0	SOD1_N	0.060493
1	pERK_N	0.039548
2	pPKCG_N	0.037327
3	CaNA_N	0.031963
4	APP_N	0.031757
5	DYRK1A_N	0.027084
6	pCAMKII_N	0.026414
7	Ubiquitin_N	0.026125
8	pS6_N	0.024124
9	Tau_N	0.022667
10	ARC_N	0.022472
11	pNUMB_N	0.021470
12	BRAF_N	0.021392
13	pPKCAB_N	0.021376
14	ITSN1_N	0.021158
15	P38_N	0.019845
16	S6_N	0.019381
17	pP70S6_N	0.019051
18	AcetylH3K9_N	0.017521
19	pGSK3B_N	0.016382

Filter-based methods

	Best Features F-Score	Feature Importances
0	SOD1_N	276.246
1	CaNA_N	240.972
2	Ubiquitin_N	161.164
3	pS6_N	138.894
4	ARC_N	138.894
5	P38_N	120.295
6	S6_N	98.047
7	pPKCAB_N	92.041
8	pGSK3B_N	86.820
9	pERK_N	86.405
10	SNCA_N	76.788
11	pMTOR_N	75.581
12	IL1B_N	71.075
13	pCAMKII_N	70.258
14	pNR2A_N	66.782
15	DYRK1A_N	62.490
16	ITSN1_N	61.513
17	pNUMB_N	59.717
18	APP_N	58.184
19	BRAF_N	53.398

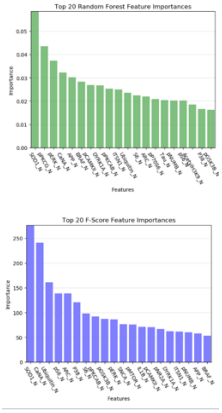


Figure 3.4.1: (L-R) Hill climbing technique, Feature importance bar chart using Random Forest Importance and F-Score method.

From the result below, it can be observed that using Random Forest Importance method gives the best performance of 98% to select 20 features.

	Feature Selection Method	Mean
0	F-Score	0.971
1	Random Forest Importance	0.981

Table 3.4.1: Summary of Feature selection mean CV score

3.5 Train and Test Split

The descriptive features of 20 selected best features from Random Forest Importance and the target feature "class" was split into training and test sets, following a 70-30 proportion.

- 756 rows of training data were used for hyperparameter tuning phase.
- 324 rows of test data were used for the performance comparison phase.
- A total of 1080 rows which is aligned with the original data.

3.6 Cross Validation and Hyper Parameter Tuning using KNN and Decision Tree

A cross validation was applied to reduce overfitting in the hyperparameter tuning phase of the 756 rows of **training data**. A cross validation using StratifiedKFold function with stratified 5-fold with no repetitions was specifically applied. This ensures that each fold has an equal proportion, especially the categorical target feature of different mouse classes.

To optimize the hyperparameters of KNN classifier, a GridSearchCV was performed with a dictionary of KNN parameters:

- Number of neighbors or 'n_neighbors' from a range of 1 to 7 with an increment of 1.
- Power parameter 'p' with a distance metric of 1 as Manhattan, 2 as Euclidean, or 3 as Minkowski.

On the other hand, to optimize the hyperparameters of a Decision Tree classifier, a GridSearchCV was also performed with a dictionary of parameters:

- A criterion of either gini or entropy, whichever seems fitting to maximize purity or information gain.
- Maximum depth or 'max_depth' of range from 1 to 16 in increments of 2, to identify the expansion of nodes until the all leaves are pure.
- Minimum samples leaf or 'min_samples_leaf' from 1 to 5.
- Minimum samples split or 'min_samples_split' from 1 to 8 to identify the minimum number of samples required to split an internal node.
- And class weight between None or balance to avoid bias on the majority class.

Lastly is to fit both the tuned KNN classifier and tuned Decision Tree classifier on the 324 rows in the **test data** in another cross-validated way, another StratifiedKFold function with stratified 10-fold with no repetitions to avoid class imbalance, overfitting and underfitting and to further identify which tuned classifier outperforms the other.

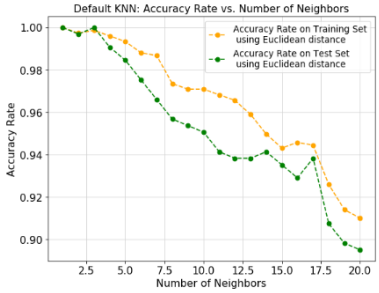
a. Default/Baseline KNN

It is important to note the current default parameters without hyperparameter tuning to be able to compare the changes to the cross-validated optimal parameters. The default parameters of the Baseline KNN were identified as n_neighbors = 5 and p = 2 (Euclidean).

	KNN Parameters	n_neighbors	p	distance metric
0	Baseline KNN	5	2	Euclidean

Table 3.5.1: KNN with default parameters

It can be observed below, that the Baseline KNN with default parameters, has a training accuracy of 0.993 and testing accuracy of 0.985 which is an obvious sign of overfitting, with values that are close to a perfect 1.0 and the testing accuracy being lesser than the training accuracy.



	KNN Classifier	Train Accuracy	Test Accuracy
0	Baseline KNN	0.993	0.985

Figure 3.5.1: Train and test accuracy of KNN with default parameter

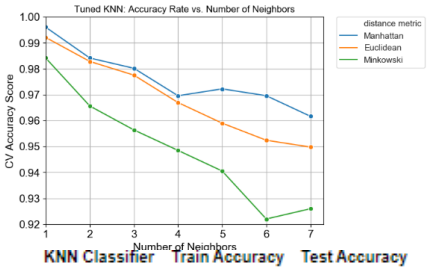
b. Tuned KNN

After the cross validated hyper parameter tuning, the optimal parameters of a tuned KNN are n_neighbors = 1 and p = 1 (Manhattan).

	KNN Parameters	n_neighbors	p	distance metric
0	Baseline KNN	5	2	Euclidean
1	Tuned KNN	1	1	Manhattan

Table 3.5.2: KNN with tuned parameters

From Figure 3.5.2 is the training accuracy of the cross validated tuned parameters, wherein it shows visual evidence that the optimal parameters of of n_neighbors = 1 and p = 1 (Manhattan) indeed performed with the highest accuracy. The tuned KNN has a training accuracy of 0.996 and testing accuracy of 0.972 which is slightly lower than the default parameters, perhaps due to noise variance.



	KNN Classifier	Train Accuracy	Test Accuracy
0	Baseline KNN	0.993	0.985
1	Tuned KNN	0.996	0.972

Figure 3.5.2: Train and test accuracy of KNN with tuned parameters

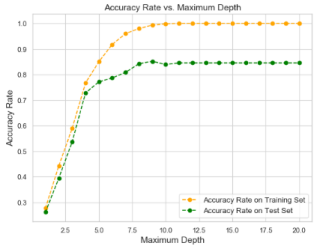
c. Baseline Decision Tree

The current default parameters of Decision Tree without hyperparameter tuning were identified as criterion = gini, max_depth = None, min_samples_split = 2 and min_samples_leaf = 1.

	Decision Tree Parameters	criterion	max_depth	min_samples_split	min_samples_leaf
0	Baseline Decision Tree	gini	None	2	1

Table 3.5.3: Decision Tree with default parameters

It can be observed below that the Baseline DT with default parameters, has a training accuracy of 1.00 and testing accuracy of 0.846, which is again an obvious sign of overfitting.



	Decision Tree Classifier	Train Accuracy	Test Accuracy
0	Baseline DT	1.000	0.846

Figure 3.5.3: Train and test accuracy of Decision Tree with default parameter

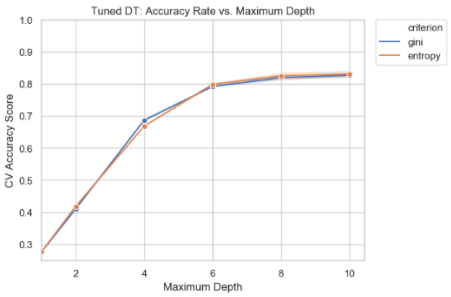
d. Tuned Decision Tree

After the cross validated hyper parameter tuning, the optimal parameters of a tuned Decision Tree criterion = entropy, max depth of 10, min_samples_split = 2, and min_samples_leaf = 1.

	Decision Tree Parameters	criterion	max_depth	min_samples_split	min_samples_leaf
0	Baseline Decision Tree	gini	None	2	1
1	Tuned Decision Tree	entropy	10	2	1

Table 3.5.4: KNN with tuned parameters

From Figure 3.5.4 is the training accuracy of the cross validated tuned parameters, wherein it shows visual evidence that the optimal parameters of criterion = entropy, max depth of 10, min_samples_split = 2, and min_samples_leaf = 1 indeed performed with the highest accuracy. The tuned DT has a training accuracy of 0.845 and testing accuracy of 0.778 which is lower than the default parameters, perhaps due to noise variance or since Decision Trees are notoriously known to overfit/underfit.



	Decision Tree Classifier	Train Accuracy	Test Accuracy
0	Baseline DT	1.000	0.846
1	Tuned DT	0.845	0.778

Figure 3.5.4: Decision Tree train accuracy with tuned parameters

Below is the graphical representation of the tuned parameters of the Decision Tree.

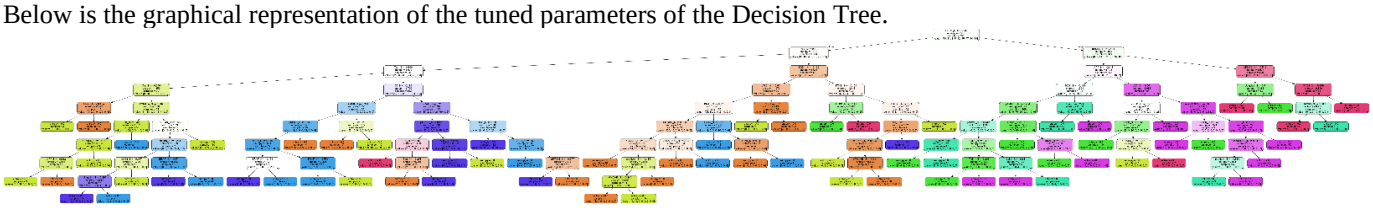


Figure 3.5.7: Tuned Decision Tree

IV. Discussion – Model Evaluation

4.1 Classification Report and Confusion Matrix

A Classification Report and confusion matrix of the tuned model with hyperparameters of the KNN and Decision Tree classifiers respectively are shown below. Since the target feature – mice class is considered a multiclass, it is more beneficial to look closely on the confusion matrix of each type of mouse class, 8 classes namely, “c-CS-m”, “c-SC-m”, “t-SC-s”, “t-CS-m”, “c-SC-s”, “c-CS-s”, “t-SC-m”, and “t-CS-s”.

a. Tuned KNN

It is evident based from both the classification report and the confusion matrix, that the tuned KNN model with optimal parameters together with the 20 important features from the Random Forest Importance method was successful in identifying each of the mouse class labels correctly. An ideal classifier would produce a pure diagonal matrix which would have all the points predicted in their correct class with the tuned KNN model achieved. Therefore, it can also be inferred that the included 20 protein features are enough to provide an accurate classifier.

Classification report for K-Nearest Neighbor

	precision	recall	f1-score	support
c-CS-m	1.00	1.00	1.00	51
c-CS-s	1.00	1.00	1.00	44
c-SC-m	1.00	1.00	1.00	35
c-SC-s	1.00	1.00	1.00	43
t-CS-m	1.00	1.00	1.00	40
t-CS-s	1.00	1.00	1.00	28
t-SC-m	1.00	1.00	1.00	41
t-SC-s	1.00	1.00	1.00	42
accuracy			1.00	324
macro avg	1.00	1.00	1.00	324
weighted avg	1.00	1.00	1.00	324

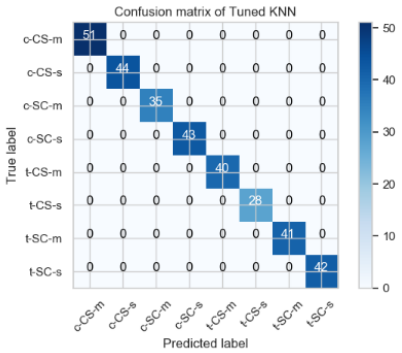


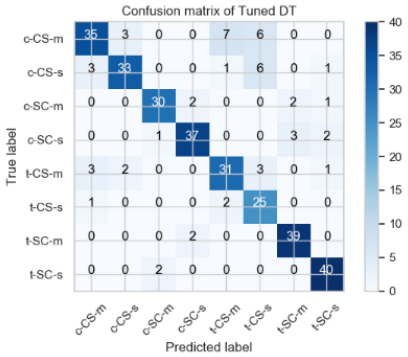
Figure 4.1.1.: Classification report and confusion matrix of the tuned KNN

b. Tuned Decision Tree

On the other hand, the tuned Decision tree model with hyperparameters had a few misclassifications in each of the mouse class. Specifically, the recall rate of the c-CS-m class is quite low with only 0.69, this is alarming as c-CS-m has the most number of mouse class in the dataset. Thus, the tuned Decision Tree with 20 features from RFI has failed to correctly classify each of the mouse class.

Classification report for Decision Tree

	precision	recall	f1-score	support
c-CS-m	0.83	0.69	0.75	51
c-CS-s	0.87	0.75	0.80	44
c-SC-m	0.91	0.86	0.88	35
c-SC-s	0.90	0.86	0.88	43
t-CS-m	0.76	0.78	0.77	40
t-CS-s	0.62	0.89	0.74	28
t-SC-m	0.89	0.95	0.92	41
t-SC-s	0.89	0.95	0.92	42
accuracy			0.83	324
macro avg	0.83	0.84	0.83	324
weighted avg	0.84	0.83	0.83	324



V. Conclusion

The objective of this study is to explore, generate insights, and to identify which types of protein have more impact on the learning outcome, to further aide in applying an appropriate machine learning modelling to correctly predict each mouse class. This has been achieved by the cross-validated tuned KNN model with 20 selected features from Random Forest Importance method.

Based on the generated insights, the findings suggest that cross validated tuned KNN with optimal parameters of n_neighbors = 1 and p = 1 (Manhattan) or in scikit learn KNeighborsClassifier(n_neighbors = 1, p = 1) correctly predict each mouse class. The selected 20 out of 77 important protein features by the Random Forest Importance also contributed to the performance of tuned KNN. Cross-validation was applied to both the train and test sets to reduce overfitting.

As a result, it generated a higher testing accuracy of 97.2% compared to a Decision Tree classifier with only 77.6% test accuracy. In addition, both its classification report and confusion matrix correctly predicted each of the eight mouse class, it was also observed that its confusion matrix formed a diagonal matrix which means it was able to predict the correct classes. It also important to note the recall score of the tuned Decision Tree scored poorly in some of the mouse classes, thus it did not meet the goal of our objective.

For future studies and research purposes, it is highly suggested to increase the sample size to reduce anomalies and to provide more accurate prediction of the mice class.

VI. References

Dealing with Multiclass data. [Online] Available from: https://towardsdatascience.com/dealing-with-multiclass-data-78a1a27c5dcc. [Accessed May 29, 2020].

How to interpret scikit's learn confusion matrix and classification report? [Online] Available from: https://stackoverflow.com/questions/30746460/how-to-interpret-scikits-learn-confusion-matrix-and-classification-report/30748053. [Accessed June 6, 2020].

Matplotlib. Creating annotated heatmaps. [Online] Available from:https://matplotlib.org/3.1.1/gallery/images_contours_and_fields/image_annotated_heatmap.html#sphx-glr-gallery-images-contours-and-fields-image-annotated-heatmap-py. [Accessed May 29, 2020].