Student ID: S3806039

Student Name: Suman Reddy Regatti

Classification of Mice Based on Protein Expression

(10 June 2020)

Affiliations: Master of Data Science, RMIT University, S3806039@student.rmit.edu.au

Table of Contents

Abstract/ Executive Summary

Expression levels of 77 proteins are measured in the cerebral cortex of 8 classes of Control and Down syndrome mice exposed to context fear conditioning .The aim of this project is to identify the subset of proteins that can identify 8 mice classe(s) leading to the success and failure of mice learning. Mouse models are used to study genetic Down syndrome in mice which is a cause of mental disability. In this project, we have used supervised machine learning methods such as KNN and Decision Tree classifier to identify which proteins are critical to mice learning after being exposed to context fear conditioning (CFC).

Introduction

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. The dataset contains a total of 1080 measurements per protein. Each measurement was 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)

Methodology

Retrieving Data

Dataset is imported into local directory from UCI Machine Learning Repository **(https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression).** The attribute information for the Mice Protein Expression Dataset with 1080 observations and 82 columns are as follows

- Mouse ID
- Values of expression levels of 77 proteins; the names of proteins indicating that they are measured in the nuclear fraction. For example: DYRK1A_n etc.
- Genotype: control (c) or trisomy (t)
- Treatment type: memantine (m) or saline (s)
- Behavior: context-shock (CS) or shock-context (SC)
- 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

Data Preparation

The following Pre-processing steps to prepare the data for exploration and Modelling

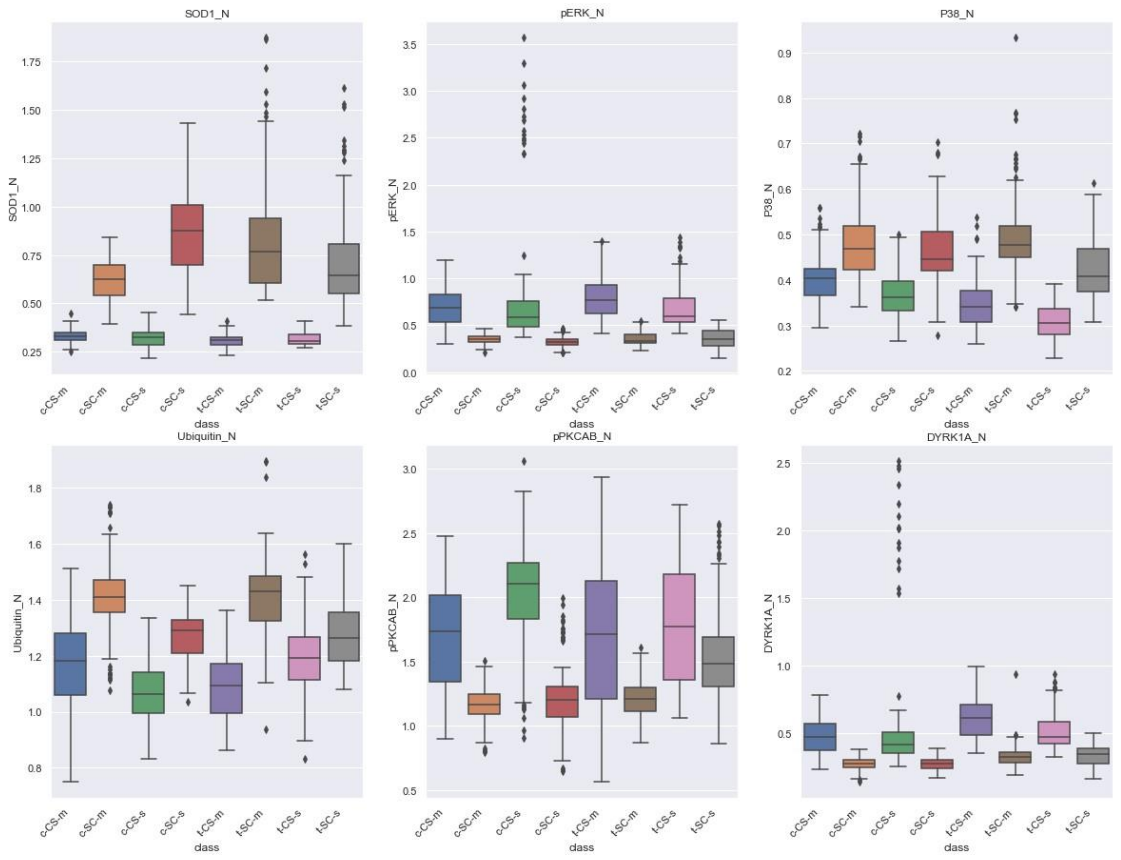
- After importing raw data, we check if the data is imported with appropriate data types.
- All the 77 variables which has protein expression values are numerical. However some of them are imported as categorical since they have missing values.
- Genotype, Treatment and Behaviour and class are categorical columns.
- Dropping Mouse Id from the raw dataset since it can cause to data leakage.
- Check typos on categorical columns
- Check for missing values on all columns and identified that 3 observations have missing values in more than 40 descriptive feature. So removing them as they will not be of much use.
- After doing the above we are left with missing values in 9 proteins. We impute the missing values with the mean value of each protein.
- Then we get descriptive statistics to get an indication of unusual values in the 77 numerical features.
- plotting boxplots on all the 77 numerical columns to suggests that there are outliers in many features. We decide to keep these outliers as they are based on univariate analysis and we are not sure whether they are true outliers are not.

Data exploration

Next we perform data exploration to get insights into the data and check if there are any meaningful relationships. We have identified few strong relationships based on the exploration as below.

Are some of the predictors able to separate the Behavior?

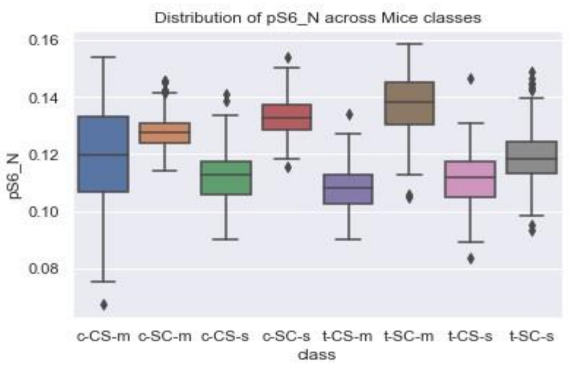
Analysis: SOD1_N separates the CS mouse very nicely. CS mouse seems to have low protein expression values where as SC mouse seems to have higher values.

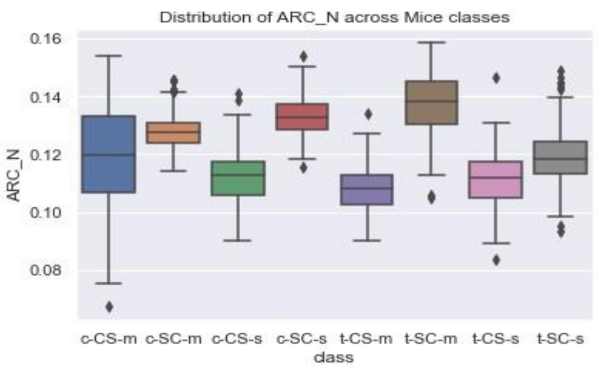


Exploring the proteins which are highly correlated

ARC_N vs PS6_N : Do they share same information with respect to target feature?

Analysis: ARC_N and PS6_N are highly correlated. Plotting the distribution of protein expression values across different Mice classes. We observe that both of them follow the same distribution. So it is evident that one of the variables is redundant.

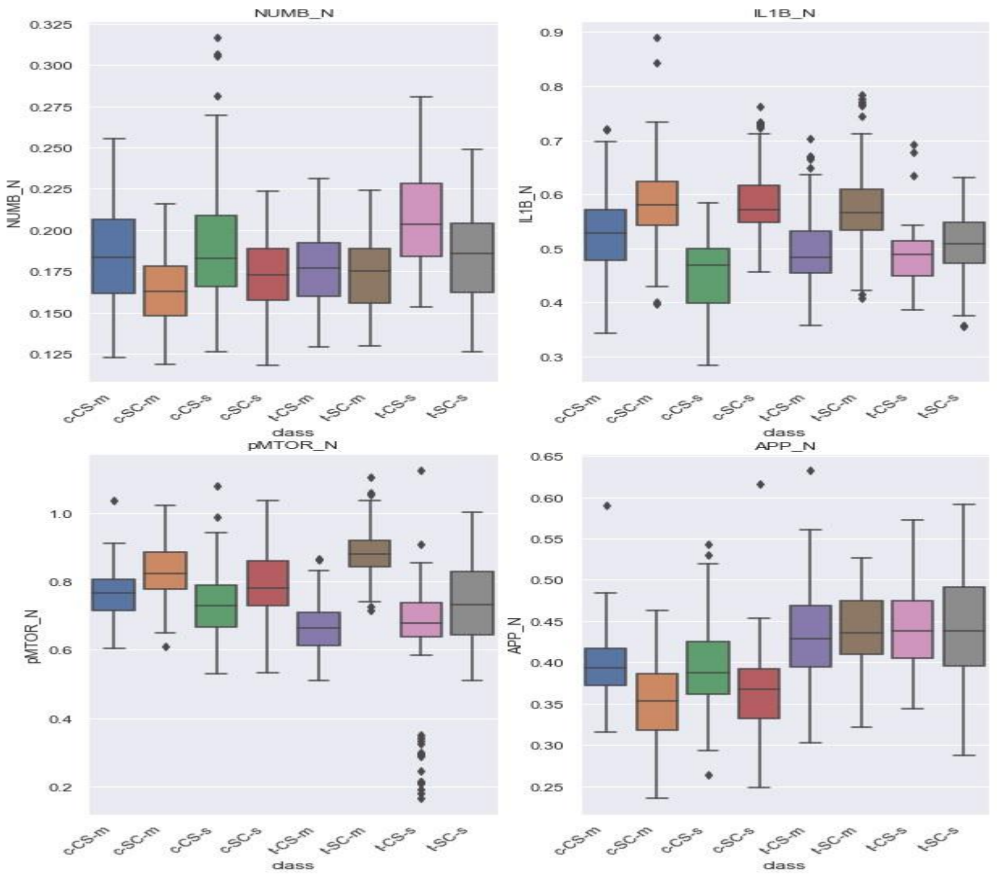




Are some of the predictors able to separate the Genotype?

APP_N has higher values for Ts65Dn mouses compared to Control mouse.

However relationship is not very strong. pMTOR_N has high values for t-SC-m compared to other mice classes.



Feature Selection

We have 77 descriptive numerical features in our dataset. All 77 might not be useful in predicting the target feature. Our focus in this section is select a subset of features which are most relevant to the target feature and remove any redundant or irrelevant features.There are filter based and wrapper based feature selection methods. Filter based feature selection methods provide rankings for the feature based on the relationship to target feature. They don't provide the explicit subset of features. Wrapper based feature selection methods use predictive models to score the subset of features and provide the best performing feature with respect to the predictive model. So it is very specific to the predictive model and the subset of feature selected using one model may not perform well for other algorithms. We are going to explore knearest neighbours and decision tree models for predicting mice classes. As we already identified the classification models ,we decide to go with the wrapper based feature selection method to find the subset of best performing feature for the two classifiers separately. Subset of features might be totally different for the two classifiers. For this problem we apply Hill climbing technique which is one of the wrapper based feature selection methods. Full set of data is used for the feature selection process. Models are scored on the hold out set. Each model is trained with the new subset of feature and scored using hold out set. The subset of features which result in maximum accuracy are the selected features.

The subset of 39 features is selected for KNN with the accuracy of 0.972.

The subset of 20 features is selected for decision tree with the accuracy of 0.872.

14 features are common for both the models.

Common features selected for both models:

'AKT_N', 'Tau_N', 'pERK_N', 'pCFOS_N', 'nNOS_N','MTOR_N','P38_N','P70S6_N','pNUMB_N',

'pMTOR_N', 'pPKCG_N','ADARB1_N','pNR1_N','pNR1_N'

Features selected only for KNN:'JNK_N','P3525_N','CDK5_N','BDNF_N','MEK_N', 'pNR2A_N', 'NUMB_N', 'IL1B_N', 'EGR1_N', 'SHH_N', 'NR1_N','GluR3_N', 'pPKCAB_N','pCAMKII_N','APP_N', 'Ubiquitin_N','pJNK_N', 'pRSK_N', 'BRAF_N', 'GSK3B_N','ITSN1_N', 'pAKT_N', 'PKCA_N' **Features selected only for Decision Tree:**
'RAPTOR_N', 'RRP1_N','BCL2_N','CAMKII_N','S6_N','BAX_N','NR2A_N', 'pCASP9_N', 'AcetylH3K9_N','pNR2B_N', 'pELK_N'

Features - knearest neighbours		**Features - Decision tree**
ADARB1_N	P3525_N	AKT_N
AKT_N	P38_N	APP_N
APP_N	P70S6_N	BCL2_N
BDNF_N	pAKT_N	DYRK1A_N
BRAF_N	pCAMKII_N	ERBB4_N
CDK5_N	pCFOS_N	MTOR_N
DYRK1A_N	pERK_N	nNOS_N

EGR1_N	pJNK_N	NR2A_N
GluR3_N	PKCA_N	NUMB_N
GSK3B_N	pMTOR_N	P3525_N
H3AcK18_N	pNR1_N	P38_N
IL1B_N	pNR2A_N	P70S6_N
ITSN1_N	pNUMB_N	pCFOS_N
JNK_N	pPKCAB_N	pERK_N
MEK_N	pPKCG_N	pNUMB_N
MTOR_N	pRSK_N	pPKCG_N
nNOS_N	SHH_N	RAPTOR_N
NR1_N	SOD1_N	S6_N
NUMB_N	Tau_N	SOD1_N
	Ubiquitin_N	Tau_N

Model Building And Hyperparameter Tuning

Next step is to build our KNeighbours model and Decision tree model using the subset of features obtained from Feature selection process. Before we build the model we need to configure the hyperparameters that are optimal for our model. The process of identifying the optimal parameter values which will give the best performance for the model is called Hyper parameter tuning. We perform hyper tuning using 5-fold cross validation strategy.

We prepare two datasets one with descriptive features. This dataset has the descriptive features selected from feature selection process. These features are scaled which will help reduce the impact of outliers and target dataset has our target feature. This is integer encoded from 0 to 7

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

Data and target are split into test and train . Hyper tuning is done using cross validation on the training set. Cross validation is used to make use of all the training data to avoid overfitting as our training set only has 753 observations.

We notice that there is a class imbalance especially for the class “t-CS-S” and mild imbalance for the other 5 classes c-CS-s,c-SC-s,t-SC-m &t-SC-s.

To make sure that each fold gets samples from each class we use stratified 5 -fold cross validation which will enforce the class distribution in each split of the data to match the distribution in the complete training dataset. In 5-fold cross-validation, the entire dataset is partitioned into 5 equalsized chunks. The first four chunks are used for training and the 5-th chunk is used for testing. Next, all the chunks other than the 4-th chunk are used for training and the 4-th chunk is used for testing, and so on. In the last iteration, all the chunks other than the 1-st chunk are used for training and the 1-st chunk is used for testing. The final step is to take the average of these 5 test accuracies and report it as the overall cross-validation accuracy.Grid search is used to search through the provided combinations of hyperparameters. grid search starts training a model of K = 1 and p=1 and calculates its accuracy score. Then it moves to train models of (K = 2, p = 1), (K = 3, p = 1), (K = 1, p = 2), ..., and (K = 3, p = 2) and obtain their score values. Based on the accuracy scores, the grid search will rank the models and determine the set of hyperparameter values that give the highest accuracy score.

KNeighbours Model:

The most important parameters for KNearest neighbours

- Number of neighbours(n_neighbors)
- P: Power parameter for the Minkowski metric.

Values considered for Number of neighbours (n_neighbors): 1,2,3,4,5,6,7,8,9,10

Values considered for p: 1,2,3,5. We have 39 features, so we will not choose the p values higher than 5 to make sure distance is not based on only dominant features and all the features have a say.

Results from Hyperparameter tuning:

- The hyper parameters with the highest accuracy score for KNN are k = 1 & p=3 with an accuracy of 0.99 on the training set.
- Our final best KNN model is sklearn.neighbors.KNeighborsClassifier(1, weights = ‘distance’, minkowski, 3).
- This model is already trained using training data while performing hyperparameter tuning.

Decision tree Model:

The most important parameters for Decision tree model are:

- max_depth: maximum depth of the tree(controls overfitting)
- min_samples_leaf: Minimum number of samples required in the terminal node
- min_samples_split: Minimum number of samples required in a node to consider for splitting.
- criterion: Algorithm which looks at the node impurity with respect to target feature.
- max_features: The number of features to consider while searching for best split.

We consider the below values for each parameter:

- criterion: gini and entropy
- max_features: we have 20 features selected from Hillclimbing technique. as per the best practices choosing 30-40% of 27 features, so tuning with 6,7,8.
- max_depth: Tuning with values from 1 to 12 as higher values can lead to overfitting. min_samples_split:Tuning with values between 2 and 7 as higher values can lead to overfitting.
- min_samples_leaf:Tuning with values 1,2,3 as higher values can lead to overfitting.

Results from Hyperparameter tuning:

The hyper parameters with the highest accuracy score for Decision tree are as below with an accuracy of 0.792 on the training set.The accuracy of 0.792 suggests that decision tree is not able to train well using the available training data.

- max_depth=12
- max_features=8
- min_samples_leaf=1
- min_samples_split=2
- So our final best Decision tree model is DecisionTreeClassifier(criterion='entropy',max_depth=12, max_features=8,min_samples_leaf=1, min_samples_split=2)
- This model is already trained using training data while performing hyperparameter tuning.

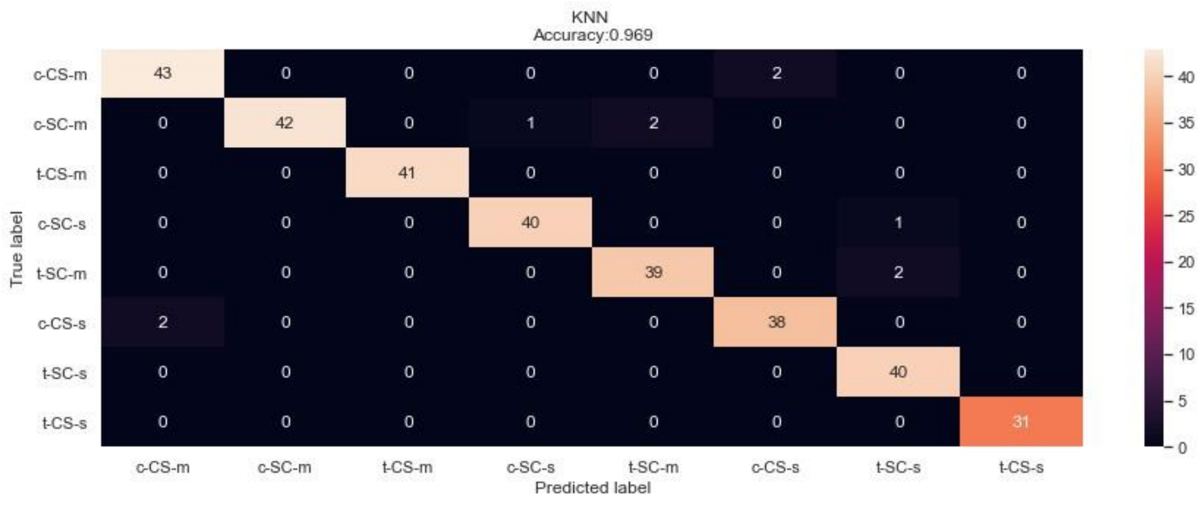
Model evaluation

Now that we have our KNN model and decision tree model built with tuned hyper parameters and trained using cross validation on training data , next step is to evaluate the model on the test set.

This test set is held out and not used while hyper tuning parameters. This is also stratified to make sure it has samples from all classes. So the bias is low and should give a good indication of the performance of our model. Accuracy score is used as a measure of performance of our model. It will be interesting to look at the f1-score also as we have mild class imbalance.

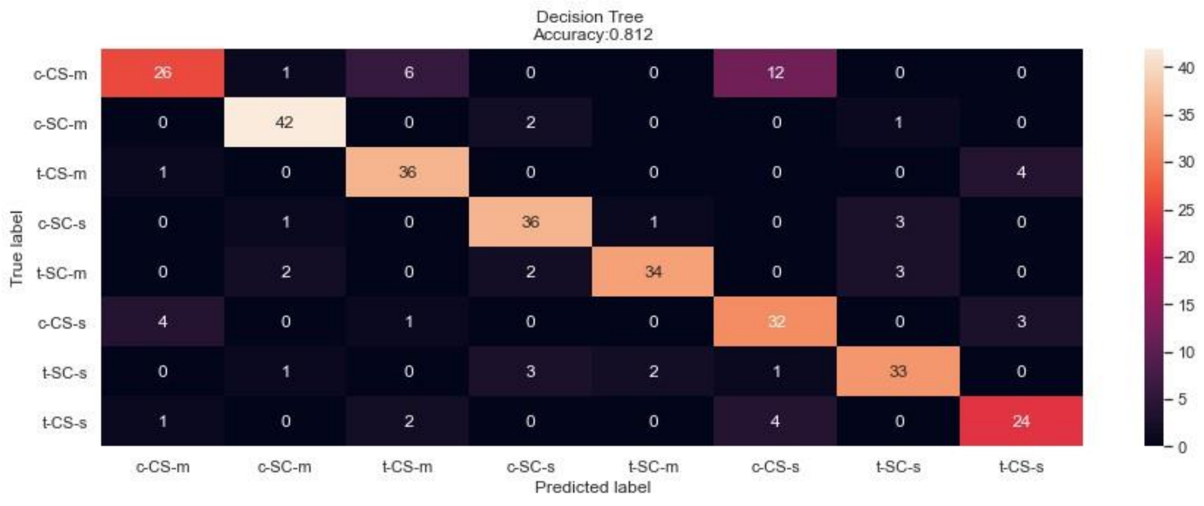
KNN Model performance:

- Accuracy score when our best KNN model is used to predict the test data : 0.969.
- Of the 324 observations from the test set 10 observations are predicted incorrectly.so the Classification error rate is 0.03. We notice that all the 10 observations are misclassified either on Genotype or Treatment. There is no misclassification based on Behaviour.
    - ✦ 2 observations are predicted as c-CS-S while the actual class is c-CS-m
    - ✦ 2 observations are predicted as t-SC-m while the actual class is c-SC-m
    - ✦ 1 observation is predicted as c-SC-s while the actual class is c-SC-m
    - ✦ 1 observation is predicted as t-SC-s while the actual class is c-SC-s
    - ✦ 2 observation is predicted as t-SC-s while the actual class is t-SC-m
    - ✦ 2 observation is predicted as c-CS-m while the actual class is c-SC-s



Decision tree Model performance:

- Accuracy score when our best Decision tree is used to predict the test data: 0.812. This is consistent with the accuracy score on training data as well.
- The classification error rate is 0.188.
- Total of 61 observations out of 324 observations from the test set are incorrectly predicted. We notice that there are many incorrect predictions for class c-CS-m as below. While many incorrect predictions are due to genotype and treatment there are few incorrect predictions for Behaviour also.
    - ✦ 12 observations are predicted as c-CS-S
    - ✦ 6 observations are predicted as t-CS-m
    - ✦ 1 observation predicted as c-SC-m



Results

Below table compares the performance of the two models on different measures.

Measure	KNN	Decision tree
Accuracy Score	0.969	0.812
Classification Error rate	0.03	0.188
Average f1-score	0.97	0.81
Average recall	0.97	0.81

From the above table it is evident that KNN outperforms decision tree for this problem.

Decision tree has a high classification error rate and large number of misclassifications are as a result of not being able to separate Genotype and treatment. Perhaps this is a result of not having protein features with strong relationship to Genotype and treatment. We have noticed this while training the Decision tree model also that accuracy was low and that the model was not able to train very well.

However, KNN overcomes this issue by making use of additional subset of features which are able to predict the mice classes.

Recommendation is to model this data using KNNeighbours classifier.

Discussion

Even though KNN model built has a very good accuracy score it uses large number of features to predict target feature.

Our KNN model is using 39 features to predict the 8 mice classes. As this number is also high, we further investigated how these 39 predictors are related the target class. We performed Anova statistical test on descriptive features and Genotype, behaviour and treatment. This test revealed that while many features have a strong relationship to behaviour none of the predictors have a strong relationship to Genotype and treatment. This is the reason why many predictors are required to predict the mice class.

There is a limitation with the feature selection process as the selection is done based on the default KNN classifier. This process could benefit from first applying filter based feature selection method which score individual features based on the relationship to target. we could remove some features before we pass it to the wrapper based feature selection method.

Conclusion

While Decision Tree suffers from lack of features with strong relation to Genotype and Behaviour, KNN is still able to predict the 8 mice classes with a very good accuracy score of 0.969 on test data with the only downside of using large number of features.

Recommendation from the above analysis is to model this data using KNN as it performed very well on the test set which is not used while training/hyperparameter tuning.

The suggestion to overcome the only downside of 39 features is to try different feature selection methods and evaluate the models.

References

- *UCI Machine Learning Repository: Mice Protein Expression Data Set*, http://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression
- "Genetics of Down Syndrome." *Wikipedia*, Wikimedia Foundation, 25 Mar. 2020, http://en.wikipedia.org/wiki/Genetics_of_Down_syndrome
- Boschetti, Alberto, and Luca Massaron. Python Data Science Essentials. Packt Publishing, 2018.
- Cielen, Davy, et al. Introducing Data Science: Big Data, Machine Learning, and More, Using Python Tools. Manning, 2016.
- Nisarachmatika. "Nisarachmatika/Machine-Learning-Mice-Protein-Expression-." *GitHub*, 21 Jan. 2018,http://github.com/nisarachmatika/Machine-Learning-mice-proteinexpression-
- "API Overview." API Overview - Matplotlib 3.2.1 Documentation, https://matplotlib.org/3.2.1/api/index.html
- "API Reference." API Reference - Pandas 1.0.3 Documentation, https://pandas.pydata.org/pandas-docs/stable/reference/index.html
- "API Reference." *API Reference - Seaborn 0.10.1 Documentation*, https://seaborn.pydata.org/api.html
- Higuera, Clara, et al. "Self-Organizing Feature Maps Identify Proteins Critical to Learning in a Mouse Model of Down Syndrome." PLOS ONE, Public Library of Science, https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0129126#sec011