Student ID: s3799691

Student Name: Gaurav Diwan

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*.

Master of Data Science, RMIT University

CLASSIFICATION OF MICE BASED ON PROTEIN EXPRESSION

Author:
Gaurav Diwan
S3799691@student.rmit.edu.au

Date: June 10, 2020

Table of Contents

An abstract/ executive summary

The aim of this report is to identify subsets of proteins that are discriminant between the classes leading to the success and failure of mice learning (8 classes based on genotype (control, c, and trisomy, t), stimulation to learn (Context-Shock, CS, and Shock-Context, SC) and treatment (saline, s, and memantine, m)). Expression levels of 77 proteins were analysed on the mice's having different characteristics to study genetic Down Syndrome which causes mental disability. Overall, the results indicate that the proteins profile of SOD1_N, pPKCG_N, pERK_N are most affected with the change in the behavior (CS or SC) and protein levels of Tau_N and H3AcK18_N changes drastically when exposed to the mice with different Genotype (Control and Trisomy). Also Control mice learn successfully while the trisomic mice fail, unless they are first treated with the drug, which rescues their learning ability. Furthermore, for our study we have built 2 supervised machine learning classification model – KNN and DT to predict different classes of mice based on the values of proteins that lead to associative learnings in mice.

Introduction

Down syndrome is a genetic disorder caused when abnormal cell division results in an extra full or partial copy of chromosome 21. This extra genetic material causes the developmental changes and physical features of Down syndrome. It's the most common genetic chromoso mal disorder and cause of learning disabilities in children. In our study, we will learn how diff erent Hsa21 genes that are critical to learning and memory reacts when exposed with differ ent drugs.
For practical evaluation of drug effects, multiple mouse models of DS, each carrying an extra copy of a subset of Hsa21 orthologous genes, have been created. Moreover, protein profiles have been measured for 77 proteins that are responsible for associative learning in both cas es where they are injected with Memantine drug and injected with no drug (saline).

The dataset used here is sourced from https://archive.ics.uci.edu/ml/datasets/Mice+Protein +Expression consists of the expression levels of 77 proteins/protein modifications that prod uced detectable signals in the nuclear fraction of cortex. There were 38 control mice and 34 trisomic mice, 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 mea surements, and for trisomic mice, there are 34x15, or 510 measurements. The dataset conta ins a total of 1080 measurements per protein. Each measurement can be considered as an i ndependent sample/mouse.

The eight classes of mice are described based on features such as genotype, behavior and tr eatment. According to genotype, mice can be control or trisomic. According to behavior, so me 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 t risomic mice, some mice have been injected with the drug and others have not.

The data set we are working with is of multivariate characteristics that contains 82 features in total with a total of 1080 observations. Information regarding each feature is given below

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

  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)

Methodology

Data Preparation

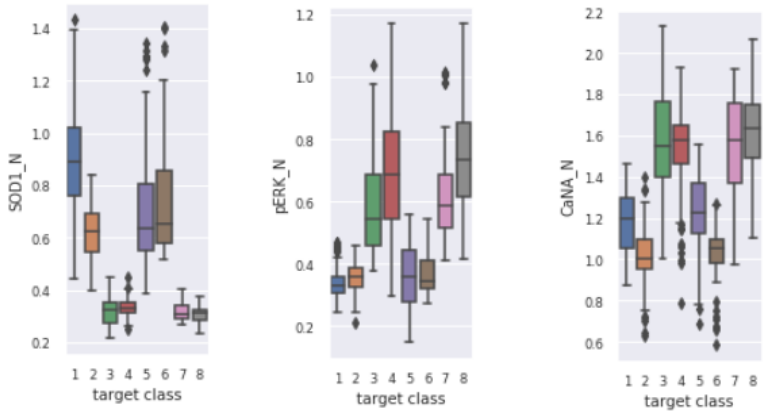
By loading the dataset into Jupyter environment using the pandas package read_excel(), we check for the general characteristics of data (dimension of data, column name and type, summary statistics for different variable types i.e. categorical and numerical). During our formal inspection we came across number of abnormalities in our dataset that were taken care in a sequence shared below

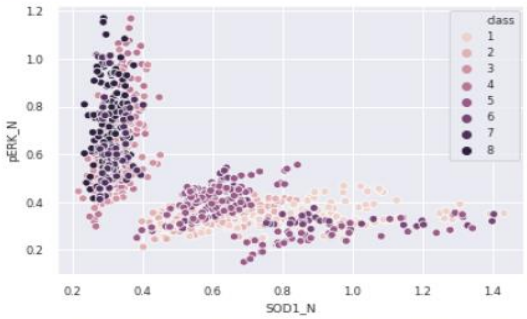
1. We drop the columns 'Mouse-ID', 'Genotype', 'Treatment' and 'Behavior' as they can cause data leakage during predictive modelling.
2. In total 49 features contain missing data. In case of observations containing more than 60% missing values we excluded that observations (3 observations in our dataset were having missing data for more than 43 features). Rest of the missing data we imputed with the mean of that feature value of the class the observation belong to.
3. Large number of features were rightly skewed. Using the quantile method, we dropped top 1% values so they follow the normal distribution (this was after exploring the distribution of each feature in our Data Exploration step) and lower 0.5% values for features that were left skewed.

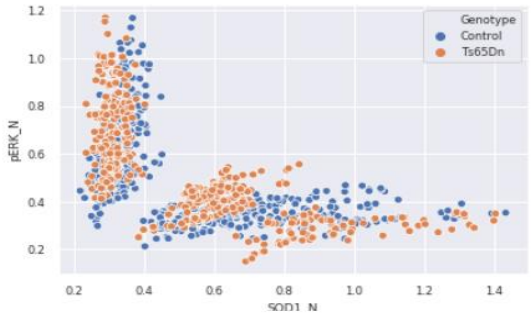
Data Exploration

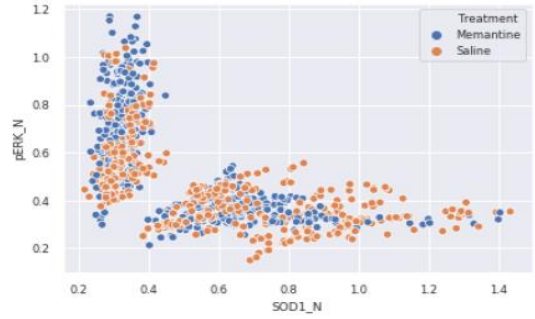
Exploring 77 features was a very time-consuming process. Only a handful of features were able to give the information about our target ‘class’ based on the different protein profiles. For the conciseness of the report we are sharing only those explorations through which we were able to get insights. Different proteins were able to identify different class labels separately.
Proteins SOD1_N , pERK_N and CaNA_N were able to distinguish classes based on Behavior i.e. mice which are stimulate to learn from the one that are not stimulated to learn.

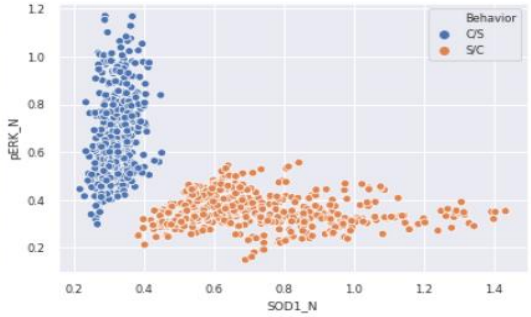
```
# integer encoding target feature
target.replace({'c-CS-m':4, 'c-SC-m':2, 'c-CS-s':3, 'c-SC-s':1, 't-CS-m':8, 't-SC-m':6,'t-CS-s':7, 't-SC-s':5},inplace = True)
```



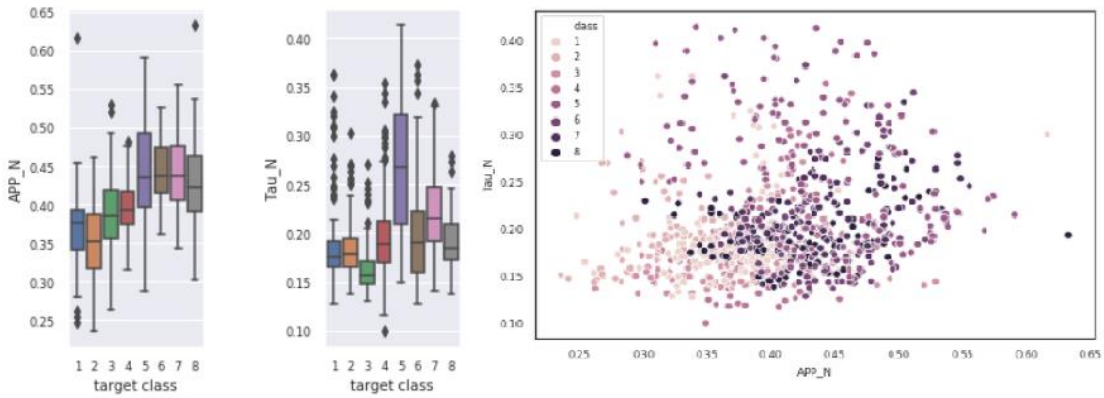








Proteins APP_N and Tau_N were able to separate majority of Control mice from Trisomic mice but with a small classification error due to overlapping.Report assignment 2 major.docx
All those that have low values for the both 'APP_N' and 'Tau_N' proteins belong to Control mices and those with high values belong to Trisomic.



Protein BRAF_N, IL1B_N, pERK_N were somewhat correlated with the 'Treatment' but none of them were able to separately classify on the basis of Treatment. They do provide the information about the range of values the particular class belong to when combined with the behavior and genotype.

```
raw_data.groupby(['Behavior','Genotype','Treatment'])['BRAF_N'].mean()
```

```
Behavior  Genotype  Treatment
C/S       Control   Memantine    0.425222
                    Saline       0.369856
          Ts65Dn    Memantine    0.509542
                    Saline       0.395663
S/C       Control   Memantine    0.262376
                    Saline       0.275757
          Ts65Dn    Memantine    0.306578
                    Saline       0.288171
Name: BRAF_N, dtype: float64
```

Apart from these proteins there were many more proteins that were providing information to predict the different labels of our class but not much significant information.

Data Modelling

The sole purpose of our model is to classify the 8 different classes(derived from genotype, treatment and behavior) of mice based on different protein values. Since the problem belongs to supervised machine learning, we are building 2 classification predictive models separately using KNN and DT from the supervised machine learning algorithm family. For the model building we will be including only features that contribute in increasing accuracy making use of Hill climbing feature selection technique.

Before building a model we performed some transformation steps on the dataset that are required by the model.

1. Partitioned the Independent (Descriptive) and Dependent(Target) features
2. Integer encoded the target feature. No label encoding required for descriptive features as all the features are already numerical.

3. Scaled descriptive feature between 0 and 1 using MinMax Scaler(from sklearn.preprocessing module) – will convert the type to ndarray.
4. Converting both the descriptive and target feature to ndarray as sklearn model works on array.

Hill Climbing Feature Selection Technique

Before creating a model we wanted to know what number of features and which features can actually help in predicting the 8 classes. Not only it will help in reducing the complexity of our model by taking care of curse of dimensionality issue but also will reduce the computational time for training the model as the model will be trained only on useful features that can help in prediction.
To perform feature selection we have used Wrapper Feature selection method Hill Climbing. It gives the number of feature after evaluating the accuracy on the estimator we will be using to create model. We have applied Hill climbing FS method on the base estimator of KNN and DT to find the features that contributes to the final accuracy score. From result KNN gives best result with 30 features (actually gives highest accuracy score with 37 feature but the difference in the accuracy score with 30 features and 37 features is less by 0.5% which can be traded off for the dimensionality reduction) and DT gives best result with 18 features.

Cross-Validation

Hold out sampling has been used for model training and evaluation. Model has been trained on the 70% training set and the performance of the model has been evaluated on remaining 30% test set with Stratified 5-fold cross validation. The random state has been fixed for results reproducibility.
**Note: Train-test split has been performed separately for both the models as different number of descriptive features are selected on basis of Hill climbing FS results.

Classification Model

- K Nearest Neighbour (KNN) classifier

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

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
```

KNN Model with default parameters (n_neighbors=5, p=2,weights='uniform') gives the accuracy of 79.4% on test set. The results are obtained from a 5 fold CV with 3 repetitions.

- Decision Tree (DT)

```
clf = DecisionTreeClassifier(random_state= 999)
clf.fit(X_train, y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=999,
            splitter='best')
```

DT model with default parameter gives the accuracy of 64.36% on test set (again cross validated in 5 fold CV with 3 repetitions).

Both the models are resulting in high overfitting. We will look for optimal values of parameters which can increase the accuracy. In next stage we hyper-tuned our models on multiple values of parameters and see if we are able to increase the accuracy of our predictive model.

Hyperparameter Tuning

To search the set of "hyperparameters" which produces the highest accuracy score for each model we performed repeated cross-validation in a Grid Search framework.

- KNN Hyperparameter Tuning

  We search for the optimal values of 2 parameters. Only a limited search of parameters were conducted since KNN algorithm is very computationally expensive.
  k = nearest neighbours -> to choose such value of 'k' for which we get a minimum of both the errors(train error and test error) and avoid overfitting as well as underfitting. Since it is multiclassification we have preferred to go with lower values of K
  p = distance metric -> both the Manhattan and Euclidean distance metric were tested.
  weights -> 'distance'. Favour will be given to the points that are nearby.

  Best model with optimal parameters

```
## Building the new model with the best parameters
final_model_knn = KNeighborsClassifier(n_neighbors=5, p=1, weights='distance')
final_model_knn.fit(X_train,y_train)
```

The model built with best parameters gave an accuracy score of 85.3% on cross validated test set. There was almost 6% improvement in the accuracy with the optimal parameters.

- DT Hyperparameter Tuning

  For DT we conducted a search on the optimal values of parameters
  Criterion for splitting -> 'Gini' or 'Entropy'
  Max_depth -> More the value more chances of overfitting.
  Min_sample_split -> should be selected such that model avoid overfitting and underfitting

  Best model with optimal parameters

```
clf = DecisionTreeClassifier(criterion='entropy', max_depth= 8 ,min_samples_split=2, random_state= 999)
clf.fit(X_train, y_train)
```

  There isn't any increase in the accuracy with the optimal parameters. The optimal model gave an accuracy score of 63.6% which reflects on the fact that decision tree model is not suited for continuous numerical variables as it looses information when it categorizes variables in different categories.

Model Evaluation

Model accuracy scores were evaluated on the test (unseen) data and were cross-validated with 5 fold CV with 3 repetitions to avoid overfitting.
Accurcy score was chosen as a metric to evaluate our model on since it provide the average score for all the classes correctly classified.

Cross- validated results

```
from sklearn.model_selection import cross_val_score

cv_results_full = cross_val_score(estimator=gs_KNN.best_estimator_,
                                  X=X_test,
                                  y=y_test,
                                  cv=cv_method,
                                  scoring='accuracy')
```

```
cv_results_full
```

```
array([0.9       , 0.84210526, 0.84210526, 0.83636364, 0.77358491,
       0.9       , 0.80701754, 0.92982456, 0.89090909, 0.77358491,
       0.81666667, 0.85964912, 0.78947368, 0.92727273, 0.90566038])
```

```
cv_results_full.mean().round(3)
```

```
0.853
```

```
from sklearn.model_selection import cross_val_score

cv_results_full = cross_val_score(estimator=gs_DT.best_estimator_,
                                  X=X_test,
                                  y=y_test,
                                  cv=cv_method,
                                  scoring='accuracy')
```

```
cv_results_full
```

```
array([0.65      , 0.68421053, 0.63157895, 0.58181818, 0.58490566,
       0.65      , 0.66666667, 0.61403509, 0.63636364, 0.66037736,
       0.65      , 0.66666667, 0.64912281, 0.6       , 0.62264151])
```

```
cv_results_full.mean().round(3)
```

```
0.637
```

Model Comparison

For performance assessment of classifiers, we use repeated cross-validation. However, we need statistical tests in order to determine if any difference between the performance of any two classification methods is statistically significant; or if it is within the sample variation and the difference is statistically insignificant. Therefore for both the model be on the same level ground we build decision tree with top 30 features as were in case of KNN and compared their accuracy on the result of cross validations using paired t-test.

Comparison with 30 features

Model	Accuracy	Classification Error
KNN	0.853	0.147
Decision Tree	0.676	0.324

```
from scipy import stats

print(stats.ttest_rel(cv_results_full_DT, cv_results_full_KNN).pvalue.round(3))
```

```
0.0
```

Since the p-value is less than 0.5 therefore difference is statistically significant and the performances of these two classifiers are not comparable. Thus, KNN model is best model among the two.

Results

KNN model outperforms DT model in classifying different classes with a much higher accuracy score. Not all the proteins profiles that deals with the associate learning in our dataset gives much information about the different classes. Only a handful number of proteins impact the learning abilities (SOD1_N impact the learning behavior most) in mice.

Confusion Matrix of KNN – correctly classifying all the 8 classes with an average accuracy of 85.3%

```
from sklearn import metrics
metrics.confusion_matrix(y_test,y_pred)
```

```
array([[42,  2,  0,  0,  0,  0,  0,  0],
       [ 0, 36,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 28,  1,  0,  0,  2,  0],
       [ 0,  0,  0, 45,  0,  0,  1,  2],
       [ 0,  0,  0,  0, 35,  0,  0,  0],
       [ 0,  2,  0,  0,  0, 26,  0,  0],
       [ 0,  0,  0,  0,  0,  0, 24,  2],
       [ 0,  0,  0,  0,  0,  0,  0, 34]])
```

Conclusion

We can also conclude from our analysis that the proteins were easily able to differentiate classes in terms of different behavior the mice were presented with. Control mice when context-shock were able to learn with or without treatment of drug (memantine), however tri-somic mice only learns when they were injected with memantine. From our study we can also conclude how ineffective Decision model becomes when exposed to continuous

numerical values. Recommended model for predictive analysis of this dataset is K Nearest Neighbour classifier which has the least error classification rate among the two built for our study.
We can further extend our study to build more powerful classification model that have low error classification score but that's beyond the scope for this report.

References

- Archive.ics.uci.edu. 2020. *UCI Machine Learning Repository: Mice Protein Expression Data Set*. [online] Available at: <https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression> [Accessed 1 June 2020].
- Higuera, C., Gardiner, K. and Cios, K., 2015. Self-Organizing Feature Maps Identify Proteins Critical to Learning in a Mouse Model of Down Syndrome. *PLOS ONE*, 10(6), p.e0129126. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0129126#sec011
- Ahmed, M., Dhanasekaran, A., Block, A., Tong, S., Costa, A., Stasko, M. and Gardiner, K., 2015. Protein Dynamics Associated with Failed and Rescued Learning in the Ts65Dn Mouse Model of Down Syndrome. *PLOS ONE*, 10(3), p.e0119491. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0119491