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*.

Compare K Nearest Neighbor and Decision Tree Model in Feature Selection with Mice Protein Expression Data Set

Wenkai Li, Student ID: s3815738, Contact details: s3815738@student.rmit.edu.au

Date of report: 02/06/2020

Contents

Abstract

Overexpression of extra genes causes learning deficits in Down syndrome (Higuera, C., Gardiner, K. and Cios, K., 2015). This experimental data set is collected by measuring protein expression of mice with different genotype, treatment, behavior and is classified into 8 classes. It contains expression levels of 77 mice proteins modifications and labels. KNN and decision tree are supervised classification algorithms. In this study, these two machine learning models are used and compared to identify subsets of proteins that are discriminants between the classes. Results suggest that two models selected similar numbers of proteins but KNN generates a higher accuracy score.

Introduction

Down syndrome is a genetic disorder caused by extra copy of genes on chromosome 21 (Hsa21). It is also known as trisomy 21 (*Down Syndrome,* Wikipedia). Overexpression of these genes will affect brain development and function to cause learning or memory deficits (Higuera, C., Gardiner, K. and Cios, K., 2015). The experiment used two groups of mice, normal mice for control group and Ts65Dn mice for experimental group. Ts65Dn mice is a mouse model used to study Down syndrome as it is similar to human trisomy 21(*Mouse Models of Down Syndrome*, Wikipedia). Each of these two groups then were put into context fear conditioning (CFC). CFC involves placing mice in a new environment, providing an aversive stimulus, then removing it (Peter C, Nathan R. Rustay, and Kaitlin E. Browman., 2009). Later put these mice back to the same environment. If mice showed response to fear like freezing, then we say that these mice learnt successfully from the environment (Peter C, Nathan R. Rustay, and Kaitlin E. Browman., 2009). By contrast, if these mice showed no response, then we say the learning failed. A Control group of mice were put in non-CFC environment. In this experiment, normal mice learnt successfully while untreated Ts65Dn mice learnt failed. Then a subgroup of Ts65Dn mice were injected with memantine. Memantine is a medication used to treat Alzheimer`s disease, which works by blocking NMDA receptors (*Memantine,* Wikipedia). These group of Ts65Dn mice learnt successfully after the injection while the control group of Ts65Dn mice failed to learn with saline injection. Context-shock (CS) and shock-context (SC) are used for describing stimulated to learn or not stimulated to learn. The mice are divided into eight classes. Class c-CS-s means control mice, stimulated to learn, injected with saline, includes 9 mice. Class c-CS-m means control mice, stimulated to learn, injected with memantine, includes 10 mice. Class c-SC-s means control mice, not stimulated to learn, injected with saline, includes 9 mice. Class c-SC-m means control mice, not stimulated to learn, injected with memantine, includes 10 mice. Class t-CS-s means trisomy mice, stimulated to learn, injected with saline, includes 7 mice. Class t-CS-m means trisomy mice, stimulated to learn, injected with memantine, includes 9 mice. Class t-SC-s means trisomy mice, not stimulated to learn, injected with saline, includes 9 mice. Class t-SC-m means trisomy mice, not stimulated to learn, injected with memantine, includes 9 mice (Higuera, C., Gardiner, K. and Cios, K., 2015).

There are 38 control mice and 34 Down syndrome mice. Each mouse has a mouseID. Each protein per mouse has 15 measurements. The measurements are expression levels of proteins modifications that produced signals in the nuclear fraction of cortex. The dataset contains 1080 rows of measurements per protein and columns for 77 different proteins. Four columns describe features include genotype for

control or trisomy, behavior for context-shock or shock-context, treatment for memantine or saline injection and the 8 classes. Totally, the data set is a 1080 * 82 data frame.

The research goal is to identify subsets of proteins that can discriminant between the 8 classes and compare these two classification models.

Methodology

Retrieving and Preparing the Data

Data Retrieving

Load the csv file and preview the first 5 rows of the dataset. Check data types of each column. Protein columns are float64 and other description columns are object. There are 1080 rows and 82 columns, which was the same with the original csv file.

Data Check

Values in protein columns are continues numeric values. Values in description columns are categorical values. A self-defined function checkValue () is used to check unique values in description columns and converted them to categorical values with no order. There are no typos nor impossible values in these columns.

Missing values and outliers

A self-defined function checkNa () was used to print total number of missing values in each column and printed the index of rows. In some columns there are high percentage of missing values. These columns include BAD has 213 missing values, BCL2 has 285 missing values, H3AcK18 has 180 missing values, EGR1 has 210 missing values and H3MeK4 has 270 missing values.

Boxplot was used to visualize each protein column to check outliers. Tukey`s method was used to detect outliers (*Outliers*, Wikipedia). Q1 represents 0.25 quantile. Q3 represents 0.75 quantile. IQR means the range of Q1 and Q3, i.e. Q3 – Q1. Outliers defined as values lower than Q1 – 1.5*IQR or higher than Q3 + 1.5*IQR. Then outliers higher than upper bound were replaced by upper bound and outliers lower than lower bound were replaced by lower bound (Anil D). After replacement, there is no outliers regarding to calculation of Tukey`s method.

Because each class might have different distributions, missing values were replaced by mean values of the same classes. This is a method mentioned in one of the relevant papers (Higuera C, Gardiner KJ, Cios KJ ,2015).

In this research, I replaced missing values with mean values from same class for the whole data set. Then I split data set into training subset and testing subset. This might introduce data leakage or other adverse effect to quality of testing data.

Normalization

The range of numeric columns are different. Some range from 0 to 1, others range from 0 to 3. Columns with higher range might influence more on the result of classifications (Higuera C, Gardiner KJ, Cios KJ ,2015). All columns were normalized to range 0 to 1 using MinMaxScaler().

Final Check

After all these steps implemented, shape, missing value, and preview the of the data frame were checked again. All issues are fixed, and the data frame is ready for data exploration and modelling.

Data Exploration

In this chapter, I explored each column by checking descriptive statistics and data visualization. I examined hypothesis relations of column pairs by plotting graphs.

Explore each column

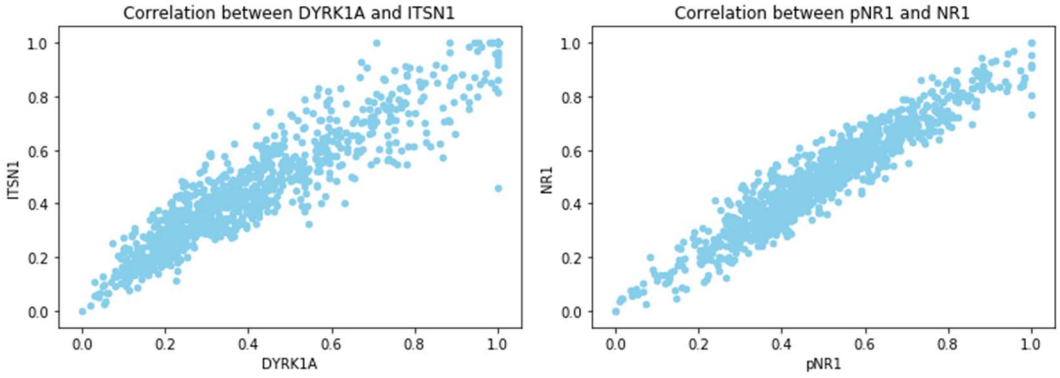
I checked descriptive statistics include count, mean, standard deviation, min, max and important quantiles of continues numeric columns. Histograms and density graphs are used to check distributions of each protein.

Some columns are normal distributed while some columns are skewed. Because there were high percentage of missing values in some columns and I simply replaced them with mean values of same classes, distributions of these columns might change. Further research might need to be conducted for exploring a more appropriate method of missing value replacing.

Explore the relationship between pairs of attributes

Hypothesis 1: DYRK1A and ITSN1 have high correlation.

DYRK1A and ITSN1 are both chromosome 21 (Hsa21) proteins. Overexpression of ITSN1 and DYRK1A leads to learning and memory deficits. (Higuera C, Gardiner KJ, Cios KJ ,2015). As can be seen from the scatter plot, they are showing a high positive correlation.

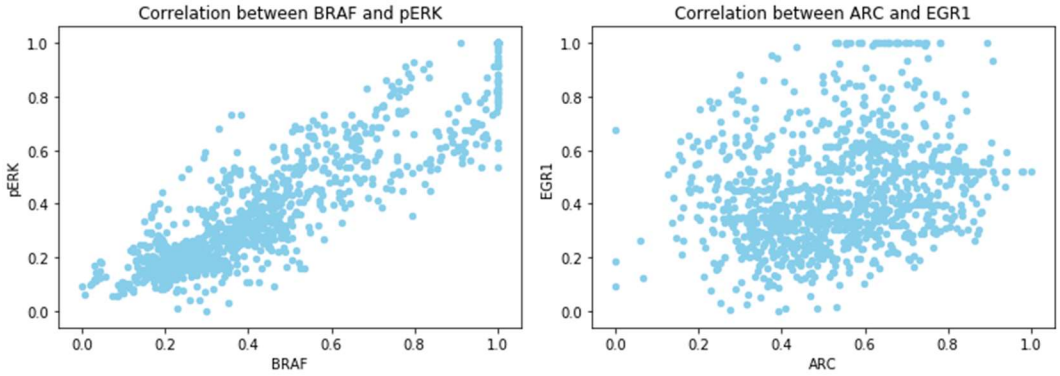


Hypothesis 2: pNR1 and NR1 have high correlation.

NR1 is N-methyl-D-aspartate receptor (NMDAR) subunit and pNR1 is phosphorylated form of NR1 (Higuera C, Gardiner KJ, Cios KJ ,2015). As can be seen, they are showing a high positive correlation.

Hypothesis 3: BRAF and pERK have high correlation.

BRAF and pERK are two components of the Mitogen-Activated Protein Kinase (MAPK) signaling pathway (Higuera C, Gardiner KJ, Cios KJ ,2015). As can be seen, they are showing a high positive correlation.

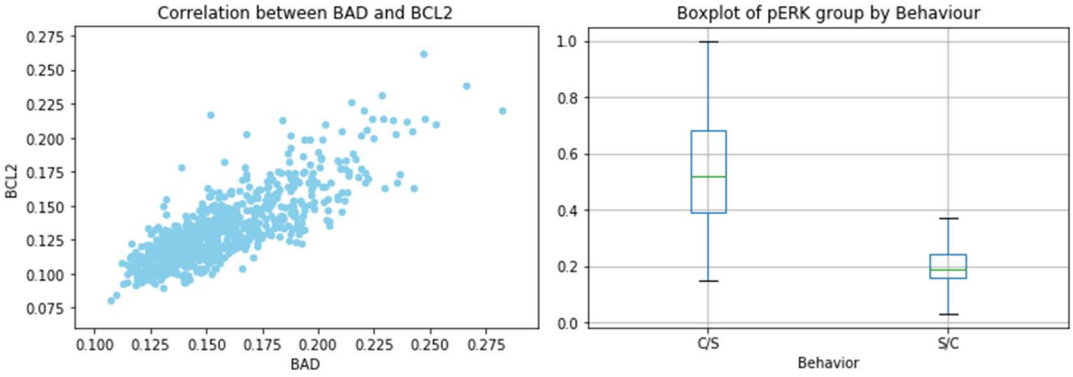


Hypothesis 4: ARC and EGR1 have high correlation.

ARC and EGR1 are both immediate early gene proteins (IEG) (Higuera C, Gardiner KJ, Cios KJ ,2015). However, they are not showing a high correlation.

Hypothesis 5: BAD and BCL2 have negative correlation.

Apoptotic is a form of programmed cell death that occurs in multicellular organisms (*Apoptotic*, Wikipedia). BAD and BCL2 are both apoptosis related. BAD is proapoptotic while BCL2 is antiapoptotic (Higuera C, Gardiner KJ, Cios KJ ,2015). However, as can be seen in the plot, they are showing a high positive correlation.

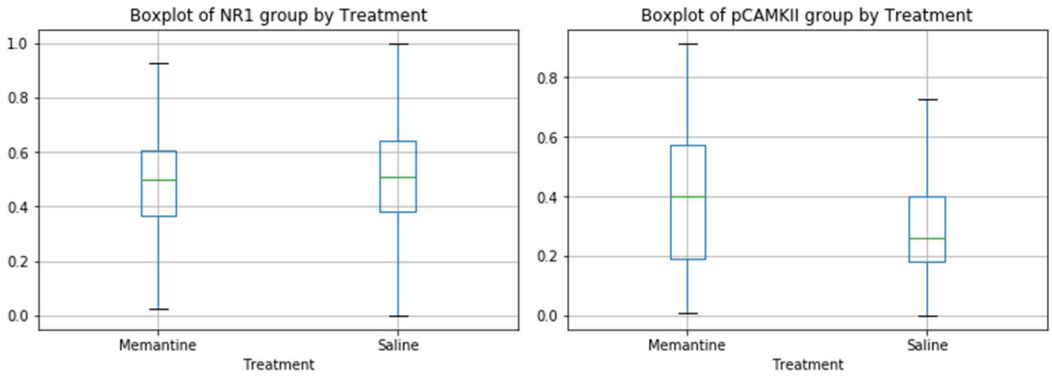


Hypothesis 6: pERK is a good discriminant between Behaviors.

pERK plays an important role in learning and memory in both general and CFC (Higuera C, Gardiner KJ, Cios KJ ,2015). As can be seen, measurement for pERK has a large difference between context-shock and shock-context. Values for C/S are much higher than S/C.

Hypothesis 7: NR1 is a good discriminant between the Treatments.

Memantine is an antagonist of the N-methyl-D-aspartate receptor (NMDAR). NMDAR subunit NR1, pNR1 plus pNUMB and pCAMKII are related to signaling from the NMDAR (Higuera C, Gardiner KJ, Cios KJ ,2015). However, from the boxplot, we cannot see a large different between mean values of memantine and saline.

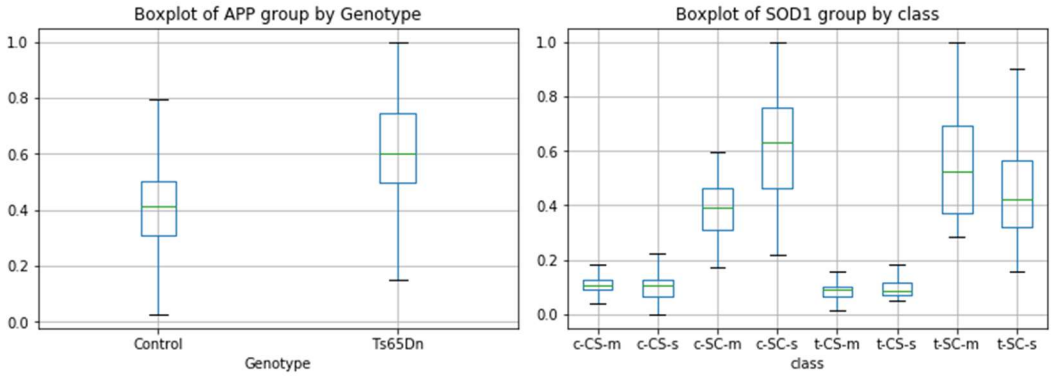


Hypothesis 8: pCAMKII is a good discriminant between the Treatments.

pCAMKII is also related to signaling from the NMDAR (Higuera C, Gardiner KJ, Cios KJ ,2015). As can be seen, mean value of saline is lower than that of memantine.

Hypothesis 9: APP is a good discriminant between the Genotypes.

APP is a Hsa21 protein. Ts65Dn mice have extra copy of genes on Hsa21 (Higuera C, Gardiner KJ, Cios KJ ,2015). As can be seen, mean value of Ts65Dn is much higher than that of Control group.



Hypothesis 10: SOD1 plays an important role in discriminating the 8 classes.

SOD1 represents to superoxide dismutase. It is an Hsa21 protein and contributes to elevated levels of reactive oxygen species seen in Down Syndrome. Overexpression of SOD1 in Ts65Dn mice leads to learning and memory deficits (Higuera C, Gardiner KJ, Cios KJ ,2015). As Higuera C, Gardiner KJ and Cios KJ reported in the relevant paper, SOD1 level increases further in rescued learning and differed in the corresponding set of control mice in normal learning, c-CS-s vs. c-SC-s.

As can be seen, mean values of different classes are quite different, especially in classes c-SC-m, c-SC-s, t-SC-m, t-SC-s.

Data modeling

As we have all the classes labeled, I chose supervised learning algorithms to build the classifier. In this research, I used and compared K Nearest Neighbor (KNN) and Decision Tree models.

Generating Train and Test Set

Before data modelling, data has to be prepared for model training and validating. I defined a subset "data" to save protein columns and a subset "target" to save classes labels. I used train_test_split()

imported from sklearn.model_selection to separate data and target into train sets and test sets. This function split data and target into 4 subsets, two for training the model and two for testing the model. They can be named as X_train, X_test, y_train and y_test. Parameter test_size and random_state can be modified. Testing set proportion can be changed through tuning the parameter test_size. test_size = 0.2 means split 20% of the data frame into testing subset and 80% of the data frame into training subset. If this parameter is blank, the default test size 0.25 will be set. In this research, I used default test size 0.25. Set random_state quals to an integer is setting a seed to random number generator. It can be used for regenerating the splitting result. In this research, I set random_state = 1.

KNN model

KNN Model building and validating

I used KNeighborsClassifier imported from sklearn to build the KNN model.

In Python, function KNeighborsClassifier () is used to build a KNN classifier. Then use fit = clf.fit(x_train,y_train) to train the classifier. After training the model, labels of test subset x_test can be predicted though y_pre = fit.predict(x_test). Evaluation metrics include confusion matrix, precision, recall rate and f1-score can be printed though classification_report(y_test,y_pre).

Then change parameters for the classifier. Repeat the steps in data modelling and compare the evaluation scores to optimize the model.

KNN Parameter tuning

Parameter “n_neighbors” is for setting the number of neighbors K. Larger value of K reduces the effects of noise but makes the classification boundaries less distinct (Yongli R, Lecture Notes). In this research, I tried n_neighbors from 3 to 21 and found that n_neighbors = 3 generates the highest score. A more strategic way of deciding n_neighbors can be deployed in further research.

Parameter “weights” is to set whether the classifier calculate a weight on distances. Set weights = “uniform” means distance from a query point to labeled neighborhoods are weighted equally. Set weights = “distance” means the classifier will assign closer neighbors of a query point a larger weight than further neighbors. In this research, weights = “distance” outperformed weights = “uniform”. We can get less features selected with same accuracy score by setting weights = “distance” than by setting weights = “uniform”.

Different distance metrics can be set by tuning parameter “metric”. The default distance metric is Minkowski metric. Parameter “p” is the power parameter for Minkowski metric. P = 1 is equivalent to Manhattan distance and p = 2 is equivalent to Euclidean distance. The influence of some features might dominate the result if we set a higher p value. Then other features influence might be ignored. To make all features contributes to the training, we can set a lower p value (Yongli R, Lecture Notes). The mice data set is high dimension with 77 features, a smaller p value might have a better result. In this research, I set p = 1 and got a better result than with higher p values.

Hill climbing for feature selection

A light version of hill climbing is used to select most important features. The model I used is the best performing model I trained from last step.

First, columns should be shuffled so they can be selected randomly. Select first column and present it to the classifier defined in last step. Then we can do classification and generate a score. Then select the

second column and present these two columns to the classifier and get another score. We compare these two scores. If the second score is higher than the previous one, we save these two columns and keep the second score as highest score. Otherwise, we remove the second column and keep the previous score. We keep repeating these steps: select one more column; do the classification; compare new score with highest score; if new score is higher, keep new column and assign the new score as highest score. Otherwise remove the new column and do not update highest score. The process will end if adding new column would not improve the highest score or the highest score hits 1.0.

KNN Results

By using the tuned KNN classifier and hill climbing technic, 20 proteins were selected with precision and recall rate both equal to 1.0. The 20 proteins in alphabet order are AcetylH3K9, BAD, BCL2, CREB, EGR1, ERBB4, GluR4, IL1B, JNK, MEK, MTOR, NR1, NR2B, P3525, P38, S6, Tau, pCAMKII, pCASP9, pGSK3B, pP70S6. The final parameter I set are n_neighbors = 3, weights = 'distance', p = 1.

Decision Tree model

I used DecisionTreeClassifier imported from sklearn to build the Decision Tree model.

Decision Tree Model building and validating

In Python, function DecisionTreeClassifier () is used to build a decision tree classifier. Following steps are the same as in KNN model building. Use fit = clf.fit(x_train,y_train) to train the classifier. After training the model, labels of test subset x_test can be predicted though y_pre = fit.predict(x_test). Evaluation metrics include confusion matrix, precision, recall rate and f1-score can be printed though classification_report(y_test,y_pre).

Decision Tree Parameter tuning

Two functions can be choosing to measure the quality of split in sklearn decision tree classifier. Parameter “criterion” can be set to choose different functions, “gini” or “entropy”. The default criterion is “gini”. Raileanu, L. and Stoffel, K. compared these two criteria and found that there is no much difference between their results, so it is not possible to decide which one performs better (Raileanu, L. and Stoffel, K., 2004.). In terms of efficiency, there is no need to compute logarithmic in gini index, so the implementation would be faster than entropy (Sebastian R, 2020). In this research, I tried both gini and entropy. The one with criterion = “gini” had a higher score.

One main challenge of decision tree is overfitting. Setting constraints on tree size though parameter tuning is a method to control overfitting. We need to set different parameters to the classifier and compare the testing results. Parameters can also be used for tuning a decision tree including max_depth, min_samples_split, min_samples_leaf, max_features.

max_depth represents the maximum depth of the tree. Default value is “None” which means the tree will develop until all leaves are pure. In this research, default max_depth value generated the best score.

Min_samples_split represents the minimum number of samples can to split in a node. The default value is 2 which means if there are more than 2 samples in a node, this node can be further split. For another example, we set min_samples_split =10. Then if there are 5 samples in a node, this node would not be split, and this node would become a leaf. Float number can be assigned to represent a percentage of total samples. In this research, default min_samples_split value generated the best score.

min_samples_leaf represents the minimum number of samples in a leaf node. The default value is 1. In this research, default min_samples_leaf value generated the best score.

max_features represents for how many features are considered. The default value is to consider all the features in the data set. It can also be set with “auto”, “sqrt” for considering square root of total feature numbers and “log2” for considering log2(n_features). Lecture slide mentioned that the square root of the total number works better than higher values, but that can vary for different cases. In this research, I tried max_features = “sqrt” but did not get a better result. So, I used the default value.

The best performing decision tree model I trained was with all parameters set to default.

Hill climbing for feature selection

I used the same hill climbing steps in KNN to select most important features. The model I used is the best performing model I trained from last step.

Results of Decision Tree

By using the tuned Decision Tree classifier and hill climbing technic, 29 proteins were selected with precision equal to 0.83 and recall rate equal to 0.82.

The 29 proteins in alphabet order are ADARB1, BAD, BAX, BCL2, CAMKII, CREB, DSCR1, EGR1, ERBB4, ERK, GluR3, GluR4, H3AcK18, IL1B, JNK, NR1, NUMB, P38, PKCA, S6, SHH, SOD1, pBRAF, pCAMKII, pCASP9, pGSK3B_Tyr216, pMTOR, pP70S6, pPKCAB. The final parameters I set are default.

The results of decision tree are not stable. It changes when running different times.

Discussion

In this study, two classification models are used to analyze which proteins are discriminant between the classes. These two models have their advantages and disadvantages.

They are both nonparametric method and have no assumptions about data distribution. In decision tree model, nodes and splits can be visualized. So, important features can be identified fast. Decision tree is not influenced by outliers and missing values, so it requires less data cleaning. However, KNN is sensitive to outliers and scales differences. KNN is normally used for numeric variables while Decision Tree can be used for both numeric and categorical variables (Yongli R, Lecture Notes). However, when using decision tree with continues numeric variables, it divides continues variables into bins. As a result, it might cause information loose when it categorizes variables. Variables in this data set are continues, which might be a reason why KNN performs better. Another disadvantage of decision tree is overfitting. Constraints should be set through parameter tuning.

In this study, for KNN classifier through hill climbing, 20 proteins were selected as the most important discriminants between the classes. By using subset of 20 selected features, precision rate and recall rate were both equal to 1.0. The accuracy score was 1.0. For Decision Tree classifier through hill climbing, 29 proteins were selected as the most important discriminants between the classes. By using subset of 29 selected features, precision rate was 0.83 and recall rate was 0.82. The accuracy score was 0.82.

There are 14 proteins selected by both KNN and decision tree. They are BAD, BCL2, CREB, EGR1, ERBB4, GluR4, IL1B, JNK, NR1, P38, S6, pCAMKII, pCASP9, pP70S6. Other proteins selected have different

influence on different classifiers. The mechanism of how these proteins discriminate classes and why they play an important role in classification need further research with domain knowledge.

Conclusion

The numbers of features selected by KNN is less than features selected by decision tree. KNN has higher precision, recall rate and accuracy score. KNN is recommended for this study regarding to project goal and accuracy. Decision tree visualization can be used to illustrate which proteins influence more than others by checking nodes and splits.

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 2 June 2020].

Wikipedia.org. 2020. *Down Syndrome*. [online] Available at: <https://en.wikipedia.org/wiki/Down_syndrome> [Accessed 2 June 2020].

Wikipedia.org. 2020. *Memantine*. [online] Available at: <https://en.wikipedia.org/wiki/Memantine> [Accessed 2 June 2020].

Wikipedia.org. 2020. *Mouse Models of Down Syndrome*. [online] Available at: <https://en.wikipedia.org/wiki/Mouse_models_of_Down_syndrome> [Accessed 2 June 2020].

Peter Curzon, Nathan R. Rustay, and Kaitlin E. Browman., 2009. *Methods of Behavior Analysis in Neuroscience*. 2nd ed. *Chapter 2Cued and Contextual Fear Conditioning for Rodents.* CRC Press. <https://www.ncbi.nlm.nih.gov/books/NBK5223/>

Wikipedia.org. 2020. *Outlier*. [online] Available at: <https://en.wikipedia.org/wiki/Outlier#Tukey's_fences> [Accessed 2 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.

Wikipedia.org. 2020. *Apoptosis*. [online] Available at: <https://en.wikipedia.org/wiki/Apoptosis> [Accessed 2 June 2020].

Scikit-learn.org. 2020. *Sklearn.Tree.Decisiontreeclassifier — Scikit-Learn 0.23.1 Documentation*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?highlight=decision%20tree#sklearn.tree.DecisionTreeClassifier> [Accessed 2 June 2020].

Raileanu, L. and Stoffel, K., 2004. Theoretical Comparison between the Gini Index and Information Gain Criteria. *Annals of Mathematics and Artificial Intelligence*, 41(1), pp.77-93.

Raschka, S., 2020. *Rasbt/Python-Machine-Learning-Book*. [online] GitHub. Available at: <https://github.com/rasbt/python-machine-learning-book/blob/master/faq/decision-tree-binary.md> [Accessed 2 June 2020].

Anil Dolgun, *Lecture Notes, Data Preprocessing,* MATH2349

Yongli Ren, *Lecture Notes, Practical Data Science with Python,* COSC2670