Student ID: s3758150

Student Name: Bijo B Thomas

Date of Report: 10/06/2020

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.

Mice Protein Expressions

Table of Content:

Abstract:

Down syndrome is a condition in which a person is born with an extra copy of their 21st chromosome a.k.a trisomy21. This causes physical and mental development delays and other disabilities.
This is a condition which is still prevalent in today's modern world and whose complete cure has still not been completely found. Studies and experiments are still on to find the correct cure for this gene mutation and several combinations of drugs have been used.
Mice have been playing a crucial role in such experiments and identifying the genes and dosage experiments. In this experiment, there are some control mice and some trisomic mice.According to the behaviour some have been stimulated to learn and some have not. Now some have been injected with the chemical memantine and the effects of memantine on trisomic mice is studied on its ability to learn.
In this assignment, we have used 2 methods of supervised learning techniques namely KNN and Decision Tree classifiers to identify the type or class of mice based on a combination of protein levels. The analysis is based on expression levels of 77 types of proteins and the mice are classified into 8 classes.

Problem Statement:

The aim of the project is to identify the subsets of protein that are discriminant between the classes.The mice is discriminated into 8 classes based on the Behaviour, genotype and treatment they are subjected to.

Introduction:

The main idea is to identify the group of proteins that will help identify the different classes of mice. These classes are basically a combination of 3 different factors namely behaviour, genotype and treatment they are subjected to.

According to the behaviour some mice are stimulated to learn i.e context-shock and others haven't i.e shock-context. The 2 different types of genotypes of the mice are controlled(normal characteristics) and trisomic(down syndrome).To study the effect of the drug in improving the learning ability in trisomic mice , some of the mice have been treated with memantine drug and some aren't.

So the different types of mice are:
c-CS-s : Controlled mice with context-shock behaviour (so stimulated to learn), injected with saline.
c-SC-s : Controlled mice with shock-context behaviour (so not stimulated to learn), injected with saline.
c-CS- m: Controlled mice with context-shock behaviour (so stimulated to learn), injected with memantine.
c-SC-m : Controlled mice with shock-context behaviour (so not stimulated to learn), injected with memantine.
t-CS-s : Trisomy mice with context-shock behaviour (so stimulated to learn), injected with saline.
t-SC-m : Trisomy mice with shock-context behaviour (so not stimulated to learn), injected with memantine.
t-CS-m : Trisomy mice with context-shock behaviour (so stimulated to learn), injected with memantine.
t-SC -s : Trisomy mice with shock-context behaviour (so not stimulated to learn), injected with saline.

The dataset contains 1080 measurements of 77 protein expression for different mice. These sample measurements are independent for each mice.

Methodology

The methodology adopted for identification of different classes of ice according to the protein level expression are as follows:

1. Uploading and Initial Analysis

	MouseID	DYRK1A_N	ITSN1_N	BDNF_N	NR1_N	NR2A_N	pAKT_N	pBRAF_N	pCAMKII_N	pCREB_N	...	pCFOS_N	SYP_N	H3AcK18_N	EGR1_N
0	309_1	0.503644	0.747193	0.430175	2.816329	5.990152	0.218830	0.177565	2.373744	0.232224	...	0.188336	0.427099	0.114783	0.131790
1	309_2	0.514617	0.689064	0.411770	2.789514	5.685038	0.211636	0.172817	2.292150	0.226972	...	0.184315	0.441581	0.111974	0.135103
2	309_3	0.509183	0.730247	0.418309	2.687201	5.622059	0.209011	0.175722	2.283337	0.230247	...	0.196219	0.435777	0.111883	0.133362
3	309_4	0.442107	0.617076	0.358626	2.466947	4.979503	0.222886	0.176463	2.152301	0.207004	...	0.111262	0.391691	0.130405	0.147444
4	309_5	0.434940	0.617430	0.358802	2.365785	4.718679	0.213106	0.173627	2.134014	0.192158	...	0.110694	0.434154	0.118481	0.140314

```
for columns in df.columns:
    if(df[columns].dtypes == object):
        print(df[columns].value_counts()," \n")
```

```
Control     570
Ts65Dn      510
Name: Genotype, dtype: int64

Memantine    570
Saline       510
Name: Treatment, dtype: int64

S/C    555
C/S    525
Name: Behavior, dtype: int64

c-SC-m    150
c-CS-m    150
c-CS-s    135
c-SC-s    135
t-SC-m    135
t-SC-s    135
t-CS-m    135
t-CS-s    105
Name: class, dtype: int64
```

Here we see the initial data and its columns. We also check the categorical values for the irregular values and white spaces which we see are not there.

2. Removing the redundant features and missing values

   We have a column of mouse_id which are unique for each mouse and hence do not contribute to any analysis and therefore we simply remove this feature from out dataset for analysis.
   Similarly the columns of genotype, behaviour and treatment acts as redundant features as the column class is just a combination of these features, and hence we remove these 3 features from out dataset as well.
   While analysing the data we see that there are a lot of missing values as we can see below.There are rows which has more than 50% of missing data and hence these observations wont contribute to correct analysis and we simply delete these rows from our data. Next the rest of the missing

values we replace these missing values with the median of the corresponding features as the medians are not affected by the outliers in the data.

```
GluR4_N                 0
IL1B_N                  0
P3525_N                 0
pCASP9_N                0
PSD95_N                 0
SNCA_N                  0
Ubiquitin_N             0
pGSK3B_Tyr216_N         0
SHH_N                   0
BAD_N                 213
BCL2_N                285
pS6_N                   0
pCFOS_N                75
SYP_N                   0
H3AcK18_N             180
EGR1_N                210
H3MeK4_N              270
CaNA_N                  0
Genotype                0
Treatment               0
Behavior                0
class                   0
Length: 81, dtype: int64
```

```
df.isnull().sum()
```

```
DYRK1A_N      3
ITSN1_N       3
BDNF_N        3
NR1_N         3
NR2A_N        3
pAKT_N        3
pBRAF_N       3
pCAMKII_N     3
pCREB_N       3
pELK_N        3
pERK_N        3
pJNK_N        3
PKCA_N        3
pMEK_N        3
pNR1_N        3
pNR2A_N       3
pNR2B_N       3
pPKCAB_N      3
pRSK_N        3
AKT_N         3
BRAF_N        3
CAMKII_N      3
CREB_N        3
ELK_N        18
ERK_N         3
GSK3B_N       3
```

3. Checking and Treating Outliers

For numerical features, we check for outliers as there are high chances of having them and they greatly influence the prediction of the models to an extent. Hence its important to treat these outliers or remove them. When we check for all the features that have outliers we see that 23 features have outliers and there are chances that removing the outliers may result in lesser data to train out model , hence we treat them. Here we are treating these outliers through capping wherein we check for outliers and replace them with the upper and lower bounds respectively.

```
count = 0
for col in df.select_dtypes(['int','float']):
    q1 = np.percentile(df[col],25,interpolation = 'midpoint')
    q2 = np.percentile(df[col],75,interpolation = 'midpoint')
    iqr = q3 -q1
    if(df[col].min() <(q1 - 1.5*iqr) or (df[col].max() >(q3 + 1.5*iqr))):
        count = count +1
print(count," features has outliers")
```

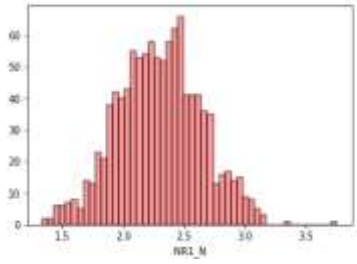
```
23  features has outliers
```

```
for col in df.select_dtypes(['int','float']):
    q1 = np.percentile(df[col],25,interpolation = 'midpoint')
    q2 = np.percentile(df[col],75,interpolation = 'midpoint')
    iqr = q3 -q1
    if(df[col].min() < (q1 - 1.5*iqr)):
        df[col] = np.where(df[col] < (q1 - 1.5*iqr),(q1 - 1.5*iqr),df[col])
    elif (df[col].max() > (q3 + 1.5*iqr)):
        df[col] = np.where(df[col] > (q3 + 1.5*iqr),(q3 + 1.5*iqr),df[col])
```

4. Data Exploration

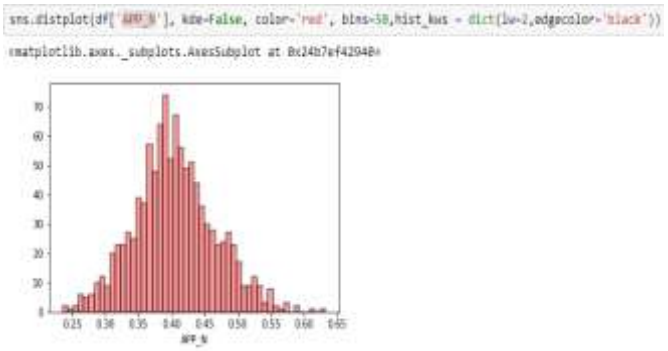
- Exploring the distribution of protein ‘NR1_N’

```
sns.distplot(df['NR1_N'], kde=False, color='red', bins=50,hist_kws = dict(lw=2,edgecolor='black'))
<matplotlib.axes._subplots.AxesSubplot at 0x24b7e50b6d8>
```



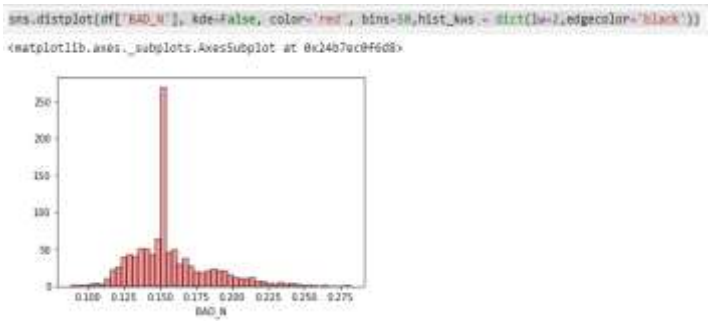
From the graph above we see that this protein various in different mice over a wide range but the majority of the mice has the protein level in the range 2.0 to 2.50.

- Exploring the distribution of ‘APP_N’



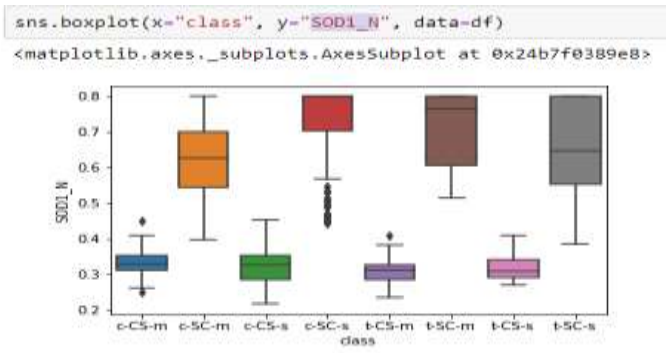
We see that this category of protein is also widely spread between the levels from 0.25 to 0.65 but the peak is in between the range 0.35 to 0.45.

- Exploring the distribution of ‘BAD_N’



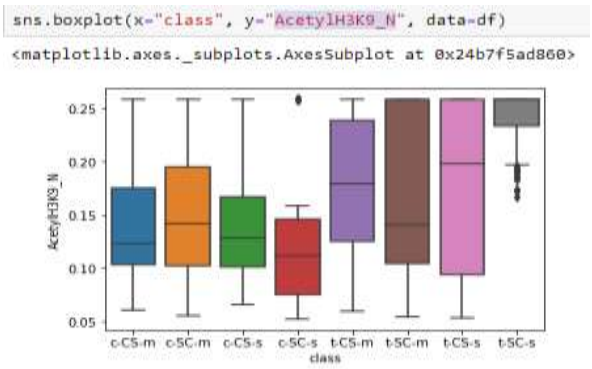
In contrast to the previous 2 graphs above, we see that in case of BAD_N levels, the protein levels is not much uniformly distributed and more than 250 mice show 0.150 level of this protein

- Exploring the relationship of behaviour and the protein ‘SOD1_N’



From the above visuals its clear that irrespective of the genotype or treatment, the mice those are stimulated to learn have lower level of the SOD1_N protein as compared to noticeable higher level of the protein in mice that are not stimulated to learn.

- Exploring the relationship between genotype and the protein ‘AcetylH3K9_N’



Here from the graph we see that irrespective of the treatment and behaviour, the trisomic mice have a higher level of ‘AcetylH3K9_N’ type of protein as compared to the controlled mice.

5. Preparing the data for modelling

```
target = df['class']
data = df.drop(columns=['Genotype','Treatment','Behavior','class'])

columnNames = data.columns
data.shape
```

```
(1077, 77)
```

```
# Scaling the data
from sklearn import preprocessing

data = pd.DataFrame(preprocessing.MinMaxScaler().fit_transform(data))
data.columns = columnNames

data.head()
```

We separate the data into the feature variables and the target variable.
Also we scale the feature variables because every column may have different magnitudes of values and these difference in values may affect the training and modelling. Hence we eliminate these difference in magnitudes through scaling so that one feature doesn't over power the other due to changes in numeric values.

6. Training and building the Algorithm

Aim of our project is to classify the mice based on the different levels of proteins and their levels, hence we use KNN and Decision Tree classifiers.

- KNN Classifier
  KNN classifier as the name suggests, classifies the new data point based on its distance from the neighbours.
  The accuracy and precision of the models can be varied by varying the parameters of the model. The process of varying the parameters to obtain the best possible accuracy of the model is called parameter tuning.
  In our project we have mainly used 2 different parameters for the tuning purpose namely distance metric 'p' which signifies the type of distance used i.e p=1 used Manhattan distance and p=2 uses Euclidean distance metric. Next we have used the 'n_neighbour' parameter which is the no. of neighbour considered while calculating the distance.
  Here we have built 4 KNN model by varying the different parameters and obtained the maximum accuracy score.

- Decision Tree Classifier
  Decision Tree classifier classifies the data in a flow chart manner wherein a feature is selected as the root node and this root node is further broken down using a split with the next best feature which forms the branch node. This is repeated till we have a leaf node that is classified into a category. Again the accuracy of the model varies with varying the parameters of the classifier. Here we have used 2 different parameters for tuning purpose namely 1.Criterion which can be 'entropy' or 'gini' and 2.max_depth which is the max depth of the tree, the default value is none which means nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
  We built 4 DT classifiers by varying the parameters and obtained the maximum accuracy score that was possible for the data.

  From the 2 models, we get the performance as shown below for different parameters:

KNN Classifier

p value	No. Of Neighbours	Accuracy Score
2	5	0.97
2	3	0.981
1	5	0.96
1	3	0.984

Decision Tree Classifier

Criterion	max_depth	Accuracy Score
gini	none	0.808
entropy	none	0.787
entropy	11	0.805
gini	8	0.799

Feature Selection

Once we have obtained the accuracy score for these models using all the features, we then do the feature selection for the model that gave the highest accuracy score. In this project we use a method called **Hill climbing technique** for feature selection which gives the best features with which we may get the highest accuracy.

From the above 2 tables we can see that the KNN model gives the higher accuracy score as compared to DT classifier.

Results

Ideally, looking at the accuracy scores of the classifiers, the KNN model which shows the higher accuracy of around 0.98 should be selected for prediction. But such high level of accuracy score also depicts overfitting. There are chances that the model may show such high accuracy on the training data but for a new real world data the accuracy drops below 50 for such over-fitted model.
Hence we select Decision Tree classifier as our model and we implement feature selection on the DT with parameters criterion='gini' and max_depth=8.

We get the result as below for the new dataset with selected features only:

Selected 20 Features are:

```
selected_data2.columns
```

```
Index(['Bcatenin_N', 'NUMB_N', 'GluR3_N', 'GluR4_N', 'P3525_N', 'pERK_N',
       'ARC_N', 'BDNF_N', 'pPKCG_N', 'pGSK3B_N', 'AKT_N', 'CaNA_N', 'pMTOR_N',
       'nNOS_N', 'P70S6_N', 'Tau_N', 'pPKCAB_N', 'H3AcK18_N', 'BAX_N',
       'BRAF_N'],
      dtype='object')
```

Accuracy Score:

```
X_train, X_test, y_train, y_test = train_test_split(selected_data2,target, test_size=0.3, random_state=0)

DT_clf =DecisionTreeClassifier(criterion='gini',max_depth = 8)
DT_fit = DT_clf.fit(X_train, y_train)
pred = DT_fit.predict(X_test)
print("Accuracy Score with selected features is",accuracy_score(y_test,pred))
```

```
Accuracy Score with selected features is 0.845679012345679
```

Confusion Matrix:

```
from sklearn.metrics import classification_report
cr = classification_report(y_test,pred)
print("Classification Report \n",cr)
```

```
Classification Report
              precision    recall  f1-score   support

        c-CS-m       0.80      0.86      0.83        37
        c-CS-s       0.75      0.79      0.77        42
        c-SC-m       0.89      0.91      0.90        43
        c-SC-s       0.93      0.91      0.92        47
        t-CS-m       0.77      0.75      0.76        44
        t-CS-s       0.77      0.69      0.73        29
        t-SC-m       0.85      0.93      0.89        43
        t-SC-s       1.00      0.87      0.93        39

   avg / total       0.85      0.85      0.85       324
```

Classification Report:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,pred)
print("Confusion Matrix \n",cm)
```

```
Confusion Matrix
 [[32  0  0  0  5  0  0  0]
 [ 2 33  0  0  2  4  1  0]
 [ 0  0 39  2  0  0  2  0]
 [ 0  0  1 43  0  0  3  0]
 [ 4  6  0  0 33  1  0  0]
 [ 2  4  0  0  3 20  0  0]
 [ 0  0  2  1  0  0 40  0]
 [ 0  1  2  0  0  1  1 34]]
```

Discussion

We have implemented 2 different classifiers namely KNN and Decision Tree Classifiers on the Mice Protein Expression data set in order to obtain subsets of protein that are discriminant between classes and to predict the class of mice based on the protein sets.
Among the 2 models, even though KNN showed a very high accuracy score of around 0.98, it also shows a scenario of overfitting which may give rise to incorrect or less accurate prediction rates for future unseen datasets. Hence we select the decision Tree model with has relatively lower but good enough accuracy rate of around 0.845 with only 20 selected features. Hence by selecting DT classifier with parameters criterion= ‘gini’ and max_depth=8, we were able to derive the 20 most important features that are required to classify the class of the mice based on protein expression levels.

Conclusion

To conclude we were able to classify the class of the mice based on the protein expression levels using the Decision Tree algorithm which had a accuracy score of 0.845 and by using just the 20 features that were obtained through Hill climbing feature selection technique.
The KNN model was also implemented , but it wasn’t chosen even after showing a very high accuracy score of 0.98 as such high score often depicts overfitting in the model which does work well with future unseen data and implementing such models posts high risks n terms of performance.

References

- Ren, Dr.Yongli 2020, 'Practical data Science:Classification', lecture notes, COSC2670, RMIT University, viewed 10 June 2020, < https://rmit.instructure.com/courses/67430/files/11273015?module_item_id=2268982 >
- Elite Data Science, 2020, *Overfitting in Machine Learning blog*, viewed 10 June 2020, < https://elitedatascience.com/overfitting-in-machine-learning >.
- Clara Higuera, Department of Software Engineering and Artificial Intelligence, Faculty of Informatics and the Department of Biochemistry and Molecular Biology, Faculty of Chemistry, University Complutense, Madrid, Spain.
  Email: clarahiguera '@' ucm.es
- Katheleen J. Gardiner, creator and owner of the protein expression data, is currently with the Linda Crnic Institute for Down Syndrome, Department of Pediatrics, Department of Biochemistry and Molecular Genetics, Human Medical Genetics and Genomics, and Neuroscience Programs, University of Colorado, School of Medicine, Aurora, Colorado, USA.
  Email: katheleen.gardiner '@' ucdenver.edu
- Krzysztof J. Cios is currently with the Department of Computer Science, Virginia Commonwealth University, Richmond, Virginia, USA, and IITiS Polish Academy of Sciences, Poland.
  Email: kcios '@' vcu.edu