COSC 2670 - Practical Data Science with Python
Data Modelling and Presentation

Report by: Jeyakaran Karnan - s3773303
Date : 10th June, 2020

Contents

Honor Code:

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.***

Chapter 1

Abstract:

The data set for the mice protein expression was created to study the effect of learning with Down Syndrome (DS) between normal and trisomic mice or mice. It is suspected that the extra copy of a normal chromosome in DS is the trigger that affects the natural pathways and natural stimulus responses, causing deficits in learning and memory. The signals that are generated in the cortex of the mice due to the injection of various measurements of proteins in them. There are totally 38 control mice and 34 trisomic mice. 15 Experiments were conducted on each sample of mouse. They will produce 8 different sets of mice based on genotype(**Control ot Trisomic**), behaviour(**Stimulated to learn and not**) and treatment(**treated with saline and memantine**). The dataset is processed through a different kind of Data Science Algorithm(**Classification or Clustering**). Analysing which type of algorithm helps us to predict the best score for the dataset stands out as the main aim of this report.

Chapter 2

Introduction:

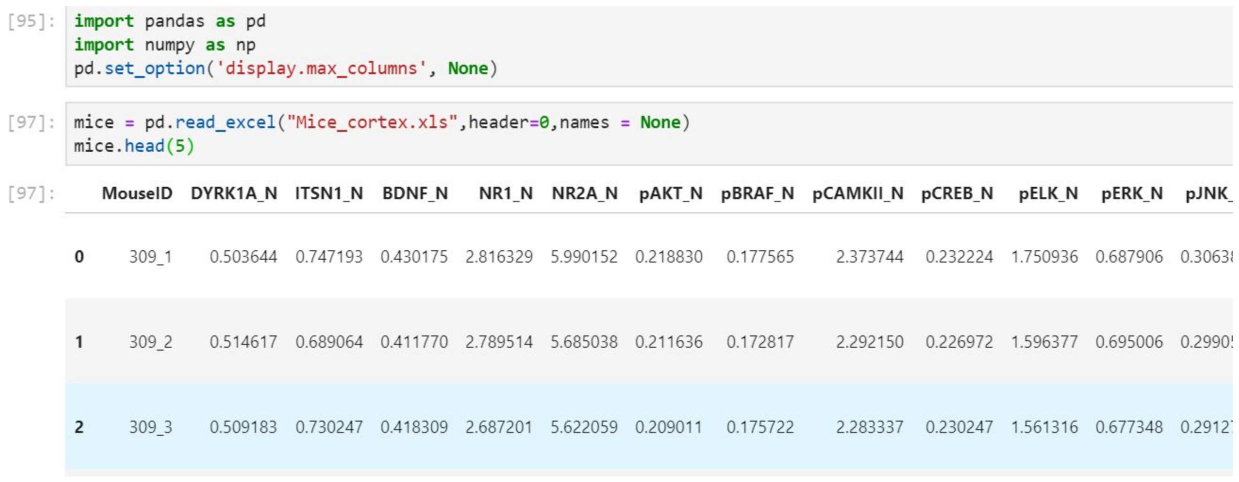
The dataset used in this process is taken from UCI repository. The dimensions of the dataset is **1080 * 82**. **570** instances for control mice and **510** instances for trisomic mice. The measurement of protein in every mice will really help us to assess the behaviour of it. Assessing the proteins in mice is really important as it may help to reflect the same in humans. Lets see how the Data Science concept gives hand is doing so.

Chapter 3

Methodology:

Data Cleaning and Preparation:

The first and foremost step in the Data Science process is Data Cleaning and Preparation. The dataset has no inconsistent values, no insane values or such. The datatype of every column has been checked at the initial stage. The dataset consists of Double data type and four other categorical columns. While working out the modelling, all other categorical columns can be dropped because it is obvious that they won't be needed in the modelling procedure.



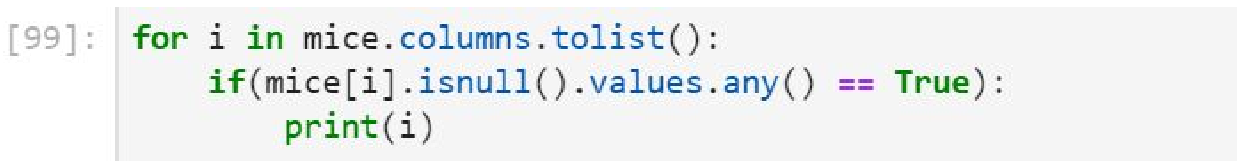
```
[95]: import pandas as pd
      import numpy as np
      pd.set_option('display.max_columns', None)
```

```
[97]: mice = pd.read_excel("Mice_cortex.xls",header=0,names = None)
      mice.head(5)
```

[97]:

	MouseID	DYRK1A_N	ITSN1_N	BDNF_N	NR1_N	NR2A_N	pAKT_N	pBRAF_N	pCAMKII_N	pCREB_N	pELK_N	pERK_N	pJNK_
0	309_1	0.503644	0.747193	0.430175	2.816329	5.990152	0.218830	0.177565	2.373744	0.232224	1.750936	0.687906	0.3063
1	309_2	0.514617	0.689064	0.411770	2.789514	5.685038	0.211636	0.172817	2.292150	0.226972	1.596377	0.695006	0.2990
2	309_3	0.509183	0.730247	0.418309	2.687201	5.622059	0.209011	0.175722	2.283337	0.230247	1.561316	0.677348	0.2912

There are many **NaN** values in each and every column. We can impute the null values using the **mean** values of every column. As a first step, let's find out what columns have **null** values by running the below code.



```
[99]: for i in mice.columns.tolist():
          if(mice[i].isnull().values.any() == True):
              print(i)
```

Data Exploration

As a next step, we move onto Data Exploration. We will be dealing with one variable and two variable plots to discuss the relationship between columns in two variable plots and the column's nature in one variable plots. Let's start with exploring one variable plots,

Fig 1 : Histogram on Protein ITSN1_N

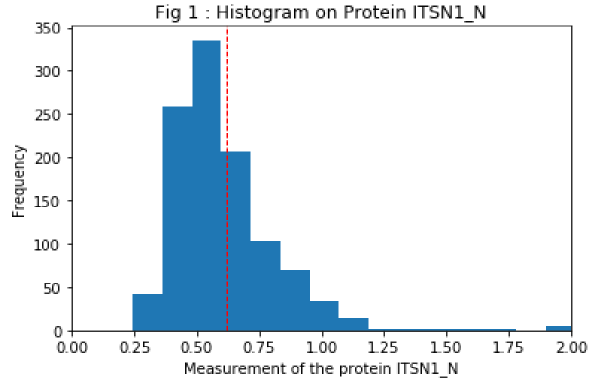


Fig 2 : Histogram on Protein DYRK1A_N

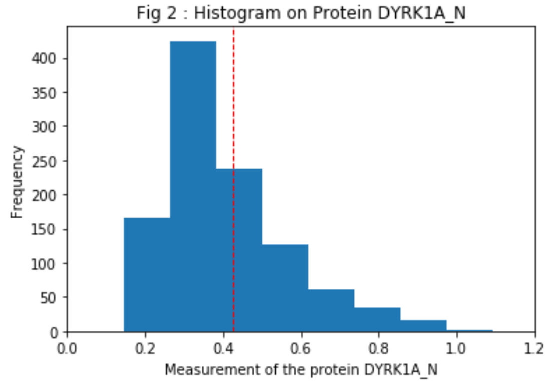


Fig 3 : Histogram on Protein BDNF_N

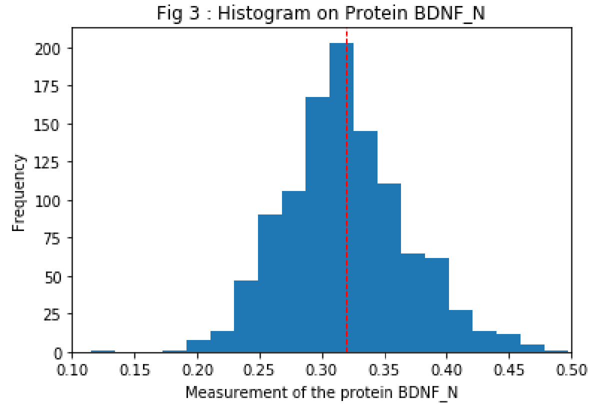


Fig 4 : Histogram on Protein NR1_N

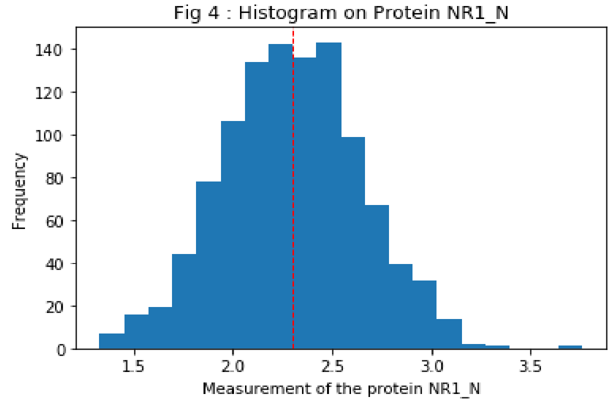


Fig 5 : Histogram on Protein NR2A_N

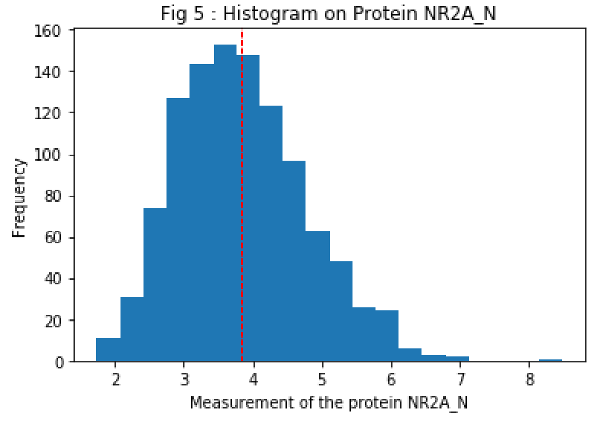
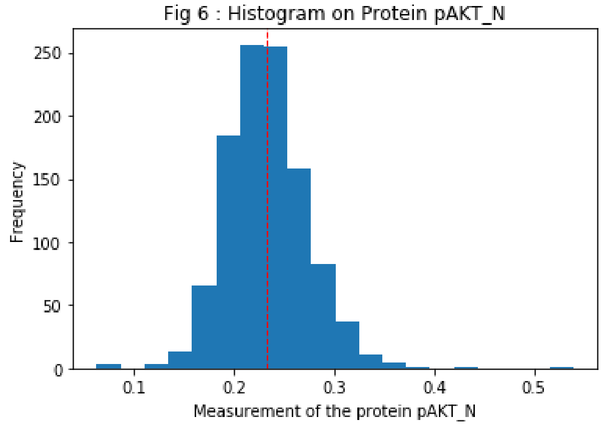


Fig 6 : Histogram on Protein pAKT_N



The amount of protein used in mice is visualised here.The mean value is computed for every protein to find out and the line is drawn between the graph to have an overall view of how average the protein is used is every sample(mouse). As far as, the distribution of NR1_N is so dense. Hereby, we can conclude that the protein **NR1_N** is a widely used amount of protein here.

Now let’s move on with two variable plots.

Let's make a hypothesis and check whether the hypothesis satisfies the results. Let's assume that every learning capacity of the control mice is **stronger** than the trisomic mice or mice with down syndrome. We can see from the graph that it proves the hypothesis as the pink color is dense in all the plots.

Figure 12: Histogram of PKCA_N segregated by Class

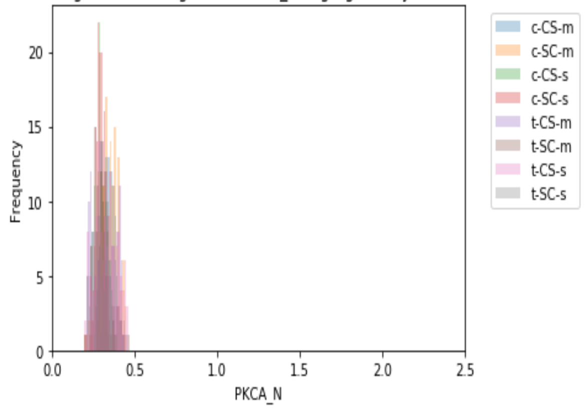


Figure 11: Histogram of pJNK_N segregated by Class

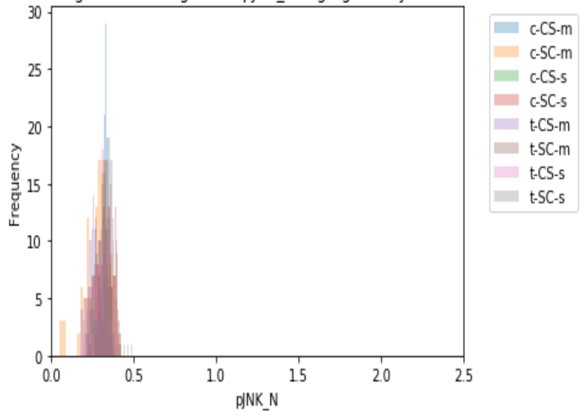


Figure 13: Histogram of pMEK_N segregated by Class

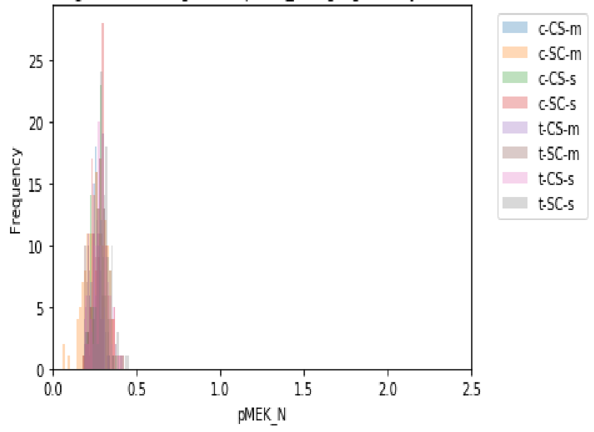
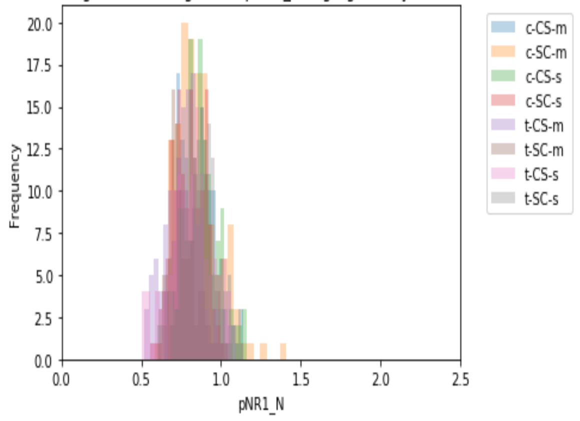


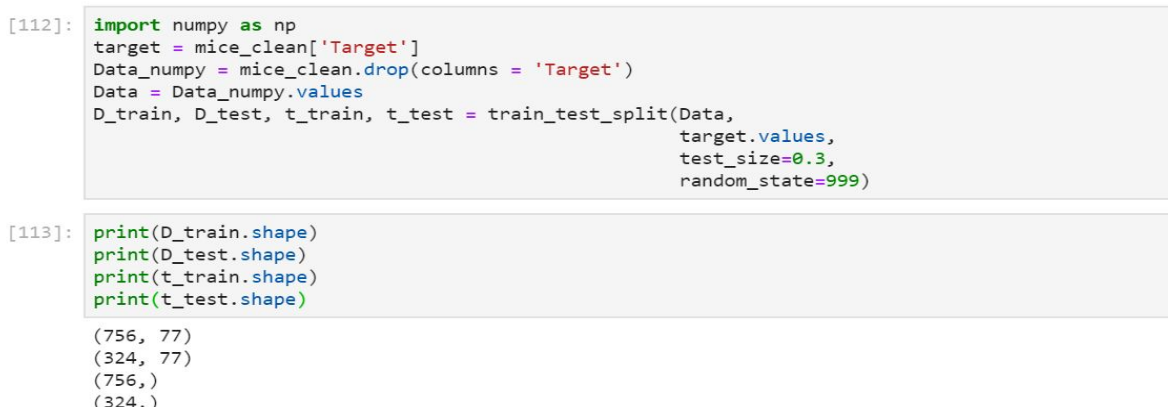
Figure 14: Histogram of pNR1_N segregated by Class



Chapter 4

Modelling

The whole dataset is split into **70% of the training set and 30% of the test set.** The best features are extracted from the training set and those features are fitted into the algorithm and check the accuracy by matching the best score with the test set.

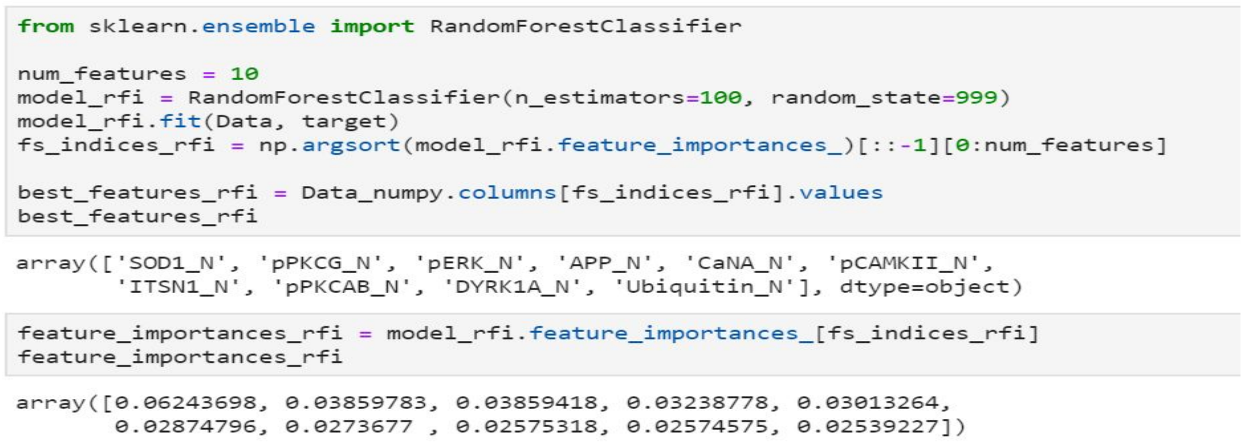


```
[112]: import numpy as np
       target = mice_clean['Target']
       Data_numpy = mice_clean.drop(columns = 'Target')
       Data = Data_numpy.values
       D_train, D_test, t_train, t_test = train_test_split(Data,
                                                           target.values,
                                                           test_size=0.3,
                                                           random_state=999)
```

```
[113]: print(D_train.shape)
       print(D_test.shape)
       print(t_train.shape)
       print(t_test.shape)
```

```
(756, 77)
(324, 77)
(756,)
(324,)
```

Lets first find out the best features out of the training set by running it through the Random forest classifier model. We can see the best features in the below figure,



```
from sklearn.ensemble import RandomForestClassifier

num_features = 10
model_rfi = RandomForestClassifier(n_estimators=100, random_state=999)
model_rfi.fit(Data, target)
fs_indices_rfi = np.argsort(model_rfi.feature_importances_)[::-1][0:num_features]

best_features_rfi = Data_numpy.columns[fs_indices_rfi].values
best_features_rfi
```

```
array(['SOD1_N', 'pPKCG_N', 'pERK_N', 'APP_N', 'CaNA_N', 'pCAMKII_N',
       'ITSN1_N', 'pPKCAB_N', 'DYRK1A_N', 'Ubiquitin_N'], dtype=object)
```

```
feature_importances_rfi = model_rfi.feature_importances_[fs_indices_rfi]
feature_importances_rfi
```

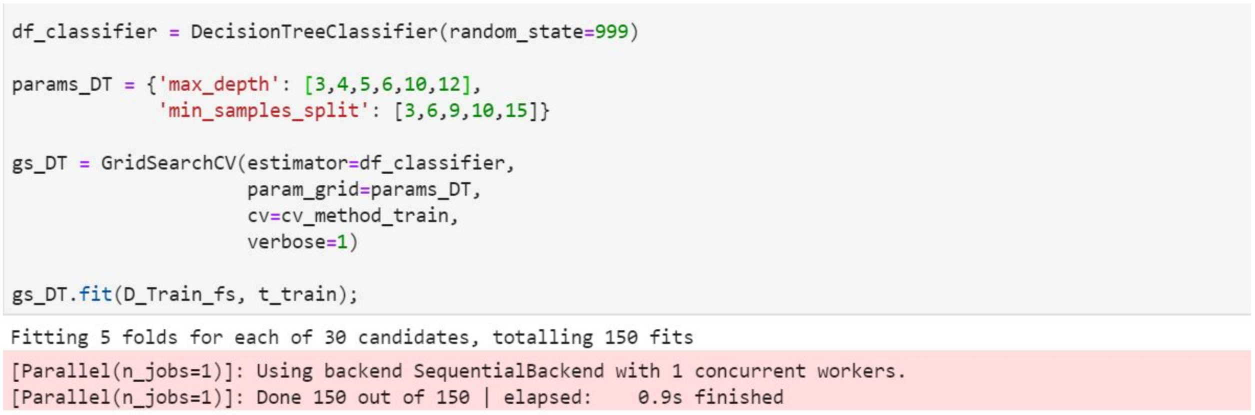
```
array([0.06243698, 0.03859783, 0.03859418, 0.03238778, 0.03013264,
       0.02874796, 0.0273677 , 0.02575318, 0.02574575, 0.02539227])
```

From the best features selected through the Random Forest Classifier the data is fitted into the model and the model is evaluated by using Cross Validation and P value test.

Decision Tree Algorithm:

We are going to apply one of the best classification methods which is **Decision Tree Classifier**. Generally, the Decision Tree will separate all the features in the form of tree. We will tell the model with varying depths and maximum samples split. So, we will be getting the best depth and best score to frame the model with.

The parameters in the **Decision Tree** are **maximum depth** and **minimum split**. The dataset we chose has the columns with all continuous values. We don't have any categorical values so the higher the depth we keep, the more accuracy we get. We can model the Tree in the following way,



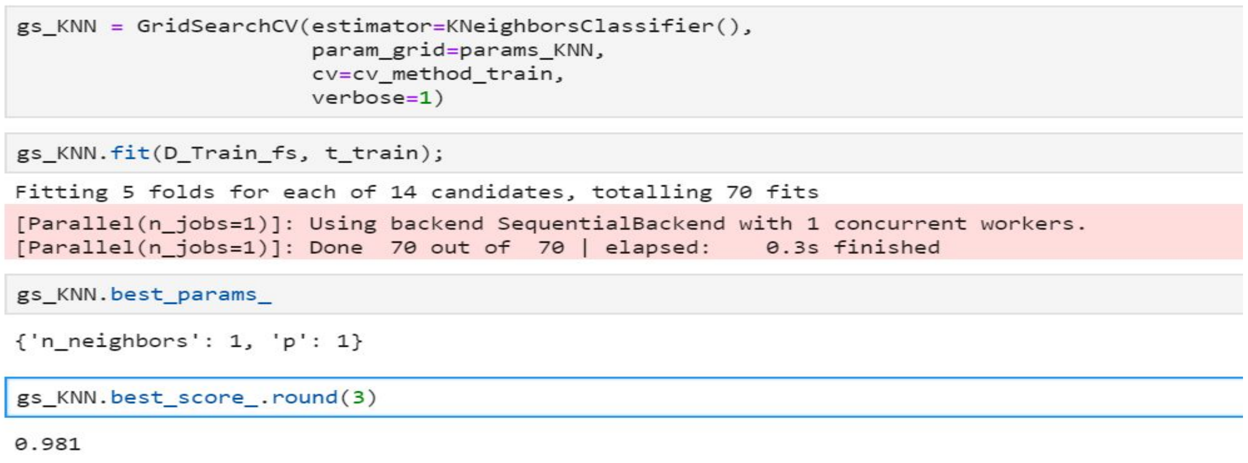
```
df_classifier = DecisionTreeClassifier(random_state=999)

params_DT = {'max_depth': [3,4,5,6,10,12],
             'min_samples_split': [3,6,9,10,15]}

gs_DT = GridSearchCV(estimator=df_classifier,
                     param_grid=params_DT,
                     cv=cv_method_train,
                     verbose=1)

gs_DT.fit(D_Train_fs, t_train);
```

```
Fitting 5 folds for each of 30 candidates, totalling 150 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 150 out of 150 | elapsed:    0.9s finished
```

The best score we derived from the decision tree is **83% accuracy.**

K-Neighbours Classifier:

K-Nearest Neighbour is one of the simplest methods that the training set is placed between the set of neighbours so that the nearest neighbour will become the predicted value. The parameters of the model with neighbours ranging from **[1, 5, 10, 15, 20, 25, 30]** and p-value as **[1, 2]**. Because of the continuous values we can't keep more neighbours in the model. Anyhow we get the best **neighbours as 1** and best **p-value as 1**.


```
gs_KNN = GridSearchCV(estimator=KNeighborsClassifier(),
                      param_grid=params_KNN,
                      cv=cv_method_train,
                      verbose=1)
```

```
gs_KNN.fit(D_Train_fs, t_train);
```

```
Fitting 5 folds for each of 14 candidates, totalling 70 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  70 out of  70 | elapsed:    0.3s finished
```

```
gs_KNN.best_params_
```

```
{'n_neighbors': 1, 'p': 1}
```

```
gs_KNN.best_score_.round(3)
```

```
0.981
```

KNN showers us with the amazing accuracy score of **98%**.

Cross Validation:

Now the best score is matched with the test set and check the matching percentage with the test set. The **KNN** model shows higher accuracy than the **Decision Tree** model. The T-Test comparison also shows us the negligible value. So we can conclude that the **KNN model** is the best classification model to predict the class of the mice.

Chapter 5

Discussion:

Before my study, I predicted that the trisomy mice could've been easily separated from the control mice, and that the drug therapy would then have a significant impact on the proteins. Interestingly, however, the mice that have been taught to know may be very well distinguished from those that have not been, however it is more hard to ascertain the genotype and procedure. So multiple variables are needed.

Chapter 6

Conclusion:

The dataset has gone through one of the two best algorithms of Data science to get the accuracy score of the algorithm for this dataset. Still the accuracy would be better if the dataset had a couple of categorical variables. But still the KNN stands out as the best algorithm for this dataset with an accuracy score of **98%**. Meanwhile, all classification models implemented and compared in the classification experiments have efficiently classified protein samples into the given eight classes with very high accuracy.

Chapter 7

Reference:

The dataset is taken from the UCI Repository.
https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression#