PRACTICAL DATA SCIENCE

ASSIGNMENT – 2

Made By:

Aakash Indoria

S3361518

Email: s3631518@student.rmit.edu.au

Date: 21/05/2020

CONTENTS:

DECLARATION:

Student ID: s3631518

Student Name: Aakash Indoria

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 honour code by typing "Yes": ***Yes*.**

EXECUTIVE SUMMARY:

This report has been created for the purpose of the second assignment for the course of Practical Data Science, semester 2020. For the assignment each student had to use a dataset, prepare and analyse it and reach to some conclusion afterwards. This report would be exploring the third dataset, which is related to the strains of protein that are commonly found in the brain of mice, affected by a mental disorder called Down's Syndrome, and normal mice, alike. This report has been made after the data was prepared and explored in the 'Jupyter' notebook using different python scripts, taught us by the RMIT tutors and lecturers for the course of PDS. It has four major parts; First one being **Introduction;** having a brief explanation about the dataset, that I chose for this assignment. Second is, **Methodology**, that explains the process of preparation of the data, here I've used two AI models, one using decision tree classifier and another using; K Nearest Neighbour (KNN). After that comes **Results**; that sheds light upon the exploration that has been done on the data, different graphs were used in this process, each having their own explanation. There after comes the **Discussion and Conclusion** part, that encompasses the results and conclusions that can be safely drawn out from the data, all the conclusions that I reached to have been backed up by respective arguments. The last part is **References**, having bibliography for this report.

INTRODUCTION:

ABOUT THE DATA:

There were three data sets given to us for the second assignment for practical data science. We were supposed to choose any one of them. I chose the third data set that was about different strains of common proteins found in the brain of mice who are affected by Downs Syndrome (a genetic mental disorder that happens when there is an extra copy of the 21st chromosome), v/s the counterparts that were normal. This mental disorder can lead to growth and intellectual delays and / or defects in the patient. I chose this dataset because, this condition is common in humans, mice and some other mammals like, pigs, etc. Also, I wanted to study and explore more about the proteins that are secreted in brain of the subjects that have this disorder.

This dataset had about 1080 rows and 82 columns, out of which, 77 columns belonged to the different strains of proteins that are found in the brain of mice. One belonged to the 'Genotype', meaning whether the subject had Down's Syndrome, specified as, 'Ts65Dn' or not, specified as 'Control'. Another column was for; 'Treatment', specified as 'Mamentine', a drug given to the mice or they were given a placebo, 'Saline'. The second last column is for behaviour portrayed by the mice, it was either 'C/S' (context-shock), or S/C (shock-context).

GOAL:

As per the first rule of data science, there must be a goal before the preparation and exploration of the data should start. For this assignment, I made my goal to take out the top 5 strains of proteins that directly affected the 'Class' column of the data set. Simply put, the top 5 proteins that are directly responsible in determining whether a subject has Down's Syndrome or not. Once found, these would help in determining this disorder in an earlier stage in infants, helping them and their loved one's cope with this disorder in a better way.

METHODOLOGY:

DATA PREPARATION:

The dataset given to us was in 'xls' format. I decided not to change it to 'csv' format, as I wanted to avoid any possible data loss while file conversion. So, to load the data into my 'Jupyter workbook', I downloaded the dataset from the link given in the assignment specifications and loaded the 'xls' file using the 'pd.read_excel' function of pandas library. My script looked like this;

```
rats = pd.read_excel('Data_Cortex_Nuclear.xls', encoding ='utf-8', header=0)
```

The first task was to 'clean' and 'cure' the dataset given to us, which was done by carrying out specific tasks on the given data. The section below would address each sub task in detail.

1. Outliers:

Outliner values are the ones that don't belong in the dataset. Treating these values carry utmost importance, as, if left untreated, they can hinder the result and analysis of the data in a great manner.

In the given dataset, I've made sure to keep the data error free before the data exploration is begun. For this I used the following scripts to cure the data in a respective way.

- ➔ Striping all the extra white spaces from the objects that are of non-numeric type:

```
rats = rats.apply(lambda x: x.str.strip() if x.dtype == "object" else x)
```

- ➔ Converting all the data, that is non-numeric to uppercase. To make the scripting easier.

```
rats = rats.apply(lambda x: x.str.upper() if x.dtype == "object" else x)
```

2. NaN / Null Values:

There were two types of column types in the dataset given to us. One as numeric type (float64), all the protein values were given in this format. And the next were 'Object' type, that included columns like 'Class', 'Genotype', 'Behaviour', etc.

I checked the object type columns and found no missing values there. But, in order to fill the 'float64' columns I made a separate dataset containing only the numeric values and used the 'fillna' function of pandas. The script is given below;

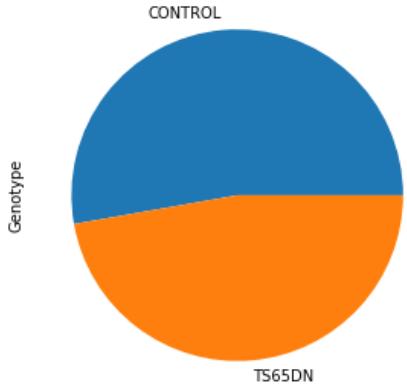
```
rats = rats.fillna(rats.mean()
```

DATA EXPLORATION

Graphs are a handy and efficient way to do the analysis of some data and show the respective results in one go. Pandas' '.plot()' function provides many graphical libraries that can be implemented to represent the type of data one has. There were two types of graphs that were asked to be made from the dataset that we choose, which included 10 single column graphs and 10 double/ multiple column graphs. Keeping in mind the structural limitation of this report I'd only include two major graphs, each, of single and multiple columns. Although, my python file could be referred to, to see the rest of the graphs.

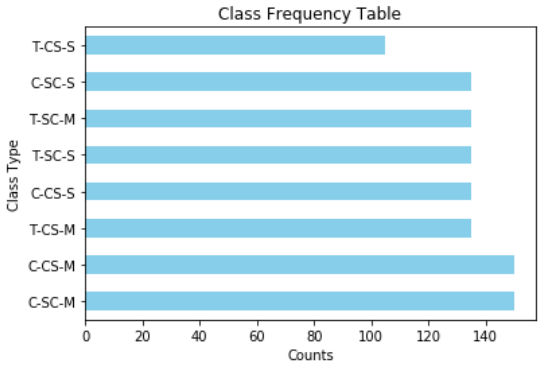
1. Task 2.1: Single Column Exploration:

- For this task we were supposed to create a graph to explore a specific column. To create my first graph I chose the column, 'Genotype'. The pie chart is given below:



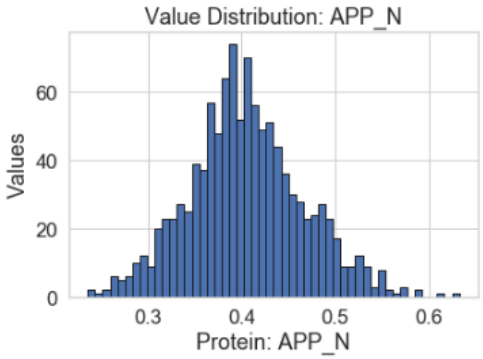
Description: As seen in the above graph the percentage of mice having ‘Control’ or normal brain type is clearly more than the ones with ‘TS65DN’ or Downs Syndrome. By looking at this data one can say that the data given to us is not an exact 50% of both Genotypes.

- Another graph that I made for the single column is for the column ‘Class’, as this column is the combination of all the last three columns of the dataset. The idea was to see how many mice fall in each of the 8 different categories given in the class. The graph is given below;



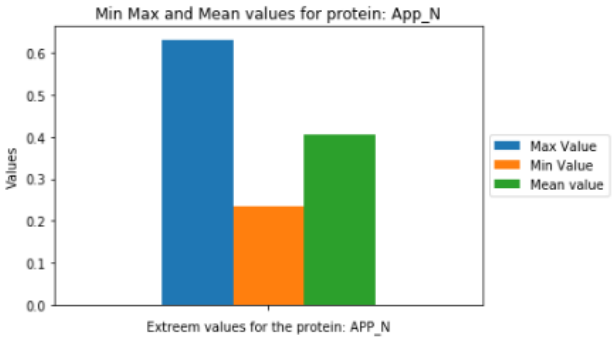
Description: The bar-graph clearly shows that about 5 of 8 values fall under the same 135 count. The first being only about 110, while the last two classes being the highest at about 150.

- The below given graph is for the protein; “APP_N”, a protein that affects the class column, the most, in a positive way.

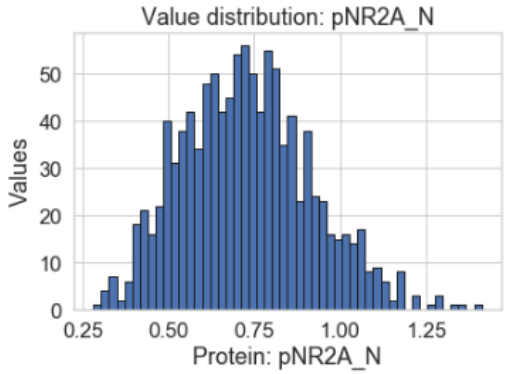


Description: The above graph explains the frequencies for the column that affects the column of class, directly, in a positive way.

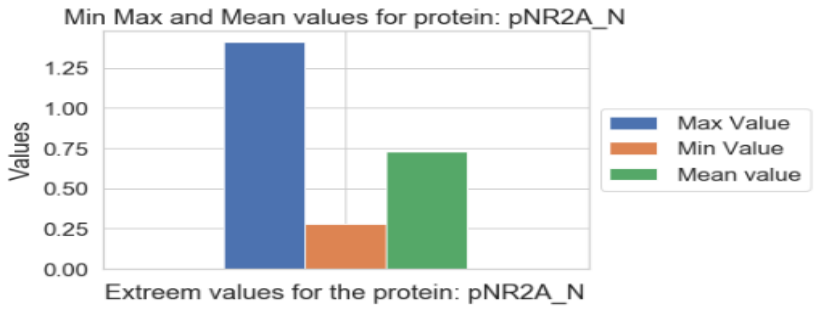
- The below mentioned graph shows the value distribution for max, min and mean of the above protein.



- Now let's look at the graph of a protein that effects the class column in a negative way.

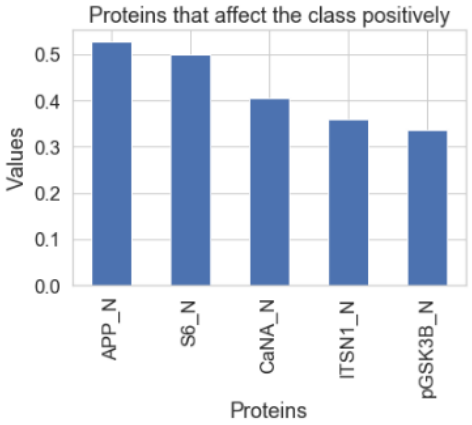


- Let's plot another graph that shows the extreme values of the above-mentioned columns, with max, min and their mean.



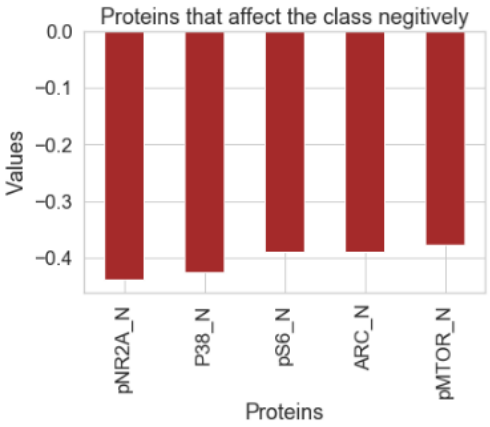
2. Task 2.1: double/multiple Column Exploration:

- This section would show some of the graphs that were derived out of double or multiple columns:

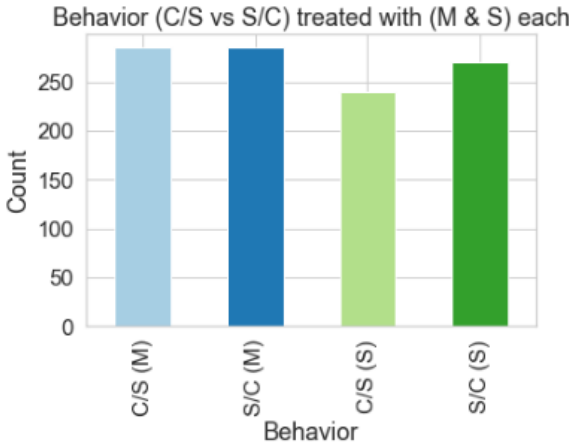


Description: The above graph portrays the different values of the top 5 columns that positively affect the class column. The derivation of this graph was possible by putting all the protein columns (from no. 2 to 78) in a separate dataset with the class column. Then the function '.corr' was used with respect to the 'pearson' method. Then the result was converted to a series, after which the series was sorted. The above graph has been made after taking the first five values.

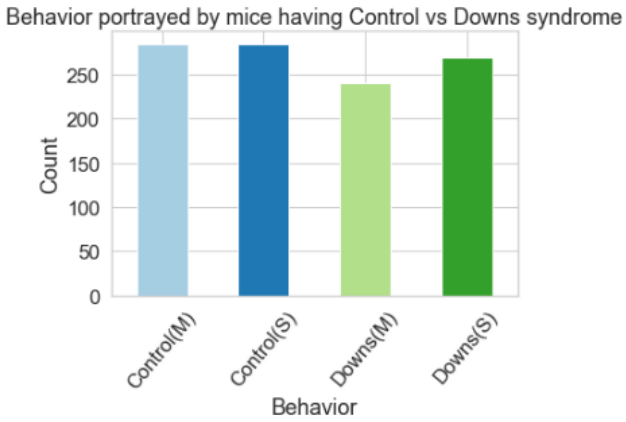
- Now let’s plot the graph of the top 5 negative columns that affect the class column directly. Which



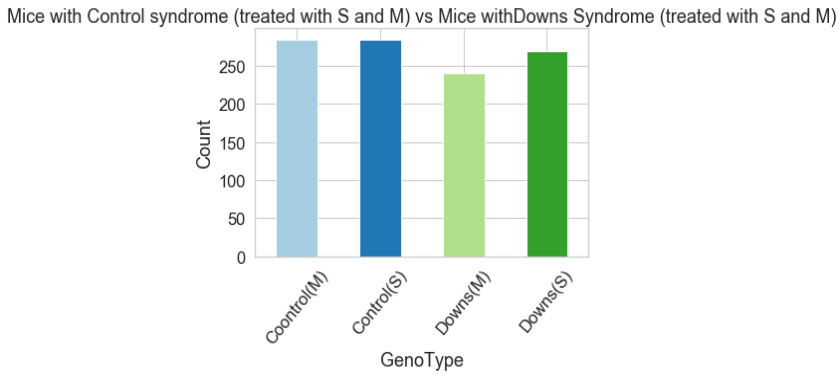
- The below graph shows the behaviour treated by either ‘Mamentine’ or ‘Saline’. The two types of behaviour C/S (counter shock) or S/C (shock counter).



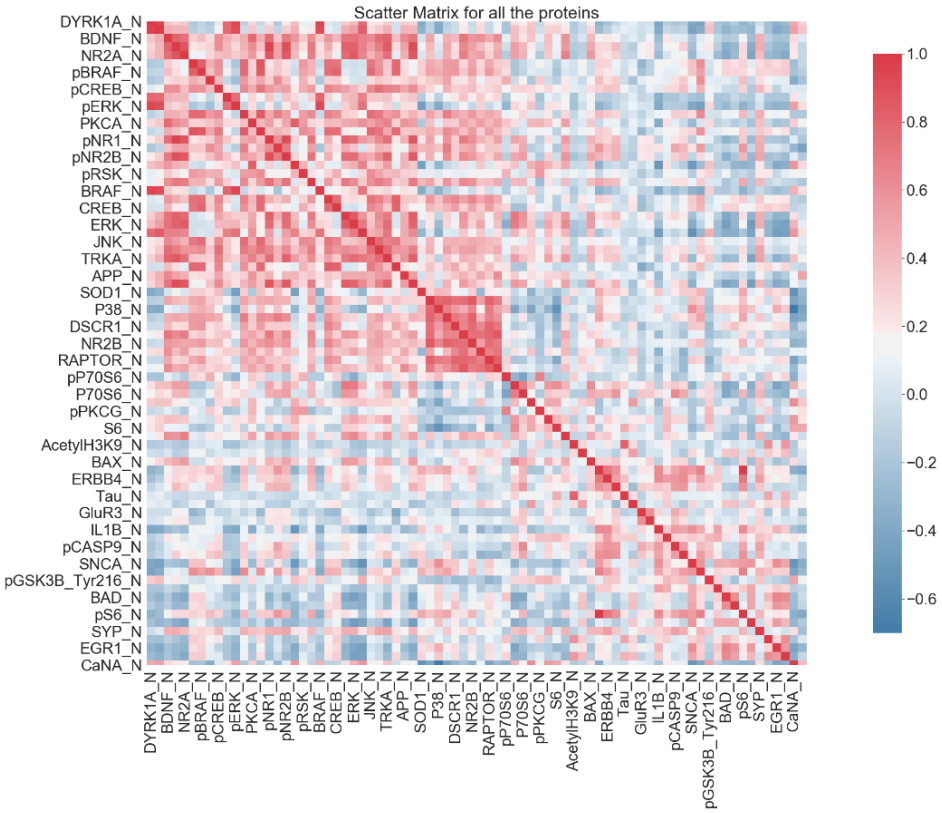
- The below graph shows the behaviour of mice having control and downs syndrome.



- The next graph shows the mice with Control syndrome (treated with Saline and Mamentine) vs Mice with Downs Syndrome (treated with Saline and Mamentine).



- The last multiple column graph is a heat map of all the proteins with respect to the class column. One can clearly see that the upper left column has lager values that are almost near the value 1. Also, we can see a small red patch, right in the middle of the graph.



MODEL CREATION AND TRAINING:

For the purpose of our assignment we had two create two different models and train them accordingly. The purpose of these models was to address the main question that we created to set our goal of this project. Hence, I created two different models that best suited my data sets, first one being Decision Tree, and the second one being KNN, K-Nearest Neighbours.

Before we jump into the creation, training and testing of different models. Let's understand that why these models carry an utmost importance in the world of data science. As, humans have their limitations and cannot do anything that needs extensive attention for a prolonged amount of time. Also, humans are prone to different errors that can be easily avoided by the machines.

Hence, in data science, one can create and train certain models that would be able to perform functions like prediction of certain types of data. This could be anything from weather forecast to the understanding and forecasting of share market stocks.

For me, the creation of these models was done to forecast the column of class, that is comprised of a subject's genotype, behaviour and the type of drug used. This forecast was supposed to be done on the basis of different values of strains of proteins that are secreted in the brain of a mouse and human alike.

As I was working with the mouse protein dataset. I took the first half, X, of my data to be all the columns with protein values, (columns 2 - 77), and for the second half, Y, of the dataset, I chose the class column to be the target column, as it is a mixture and combination of the last three columns.

Given below is the process that I used to create, train, test and finally, tune my Decision-Tree model.

RESULTS:

FIRST MODEL: DECISION TREE:

For the creation of my first model I choose the decision tree as it best suited the type of my dataset. First, the below query was used to create different datasets for the purpose of training and testing the tree.

-> *xTrain, xTest, yTrain, yTest = train_test_split(x, y, test_size=0.4, random_state=1)*

Here, I've used the 'train_test_split' function of sklearn, a library created in python for creation of models that could be used in deep learning, neural networks, etc. Also, I've taken the 'test_size' as 0.4, meaning, I'm keeping 60% of the data to train my model and then the rest 40% data would be used to test the trained model.

After this I've made the model using 'DecisionTreeClassifier' and ran it on the default parameters. Only setting the criterion as 'gini'. Please refer to my script below;

-> *modelOne = DecisionTreeClassifier(criterion='gini', random_state=1)*

modelOne.fit(xTrain,yTrain)

forcast = modelOne.predict(xTest)

accuracy_score(yTest, forcast)

By running the above script, I got an accuracy of 84%. But when I created the same tree using 'entropy' as my criterion. I got an accuracy of 81%. Just like this, I tested and tuned each parameter of my decision tree by using many scripts. Which can be found in my python notebook.

The below given table shows the different values of the parameters found after tuning each of them using the cross-validation folds.

Parameter	Value achieved after tuning:
criterion	Gini
Max_depth	10
Min_samples_split	2
Min_samples_leaf	1
Max_features	None
Max_leaf_nodes	None
Random_state	2

But, after all the parameters were tuned, I got the final script that is given below, please note that all the parameters that are included in the decision tree are given below;

-> *testMod = DecisionTreeClassifier(criterion='gini', max_depth=10, min_samples_split=2, min_samples_leaf=1, max_features=None, max_leaf_nodes=None, random_state=2)*

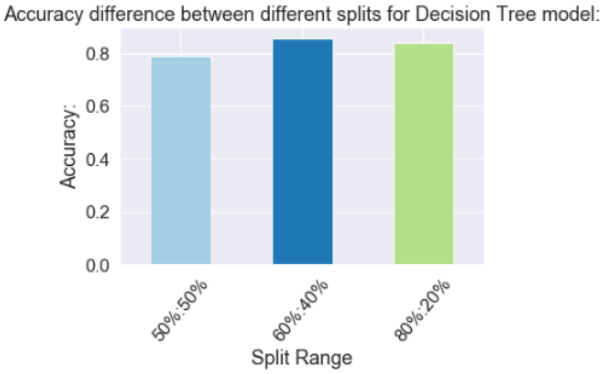
testMod.fit(xTrain,yTrain)

forcast = testMod.predict(xTest)

accuracy_score(yTest, forcast)

By running the above script, I was able to achieve an accuracy score of 85%. Apart from this, I've also included a confusion matrix and a classification report based on my model for Decision Tree Classifier.

-> Before finalising this split ratio, I tried t split to make this model with three different split ratios: 50:50, 60:40 and 80:20, respectively. The graph comparison of the achieved accuracy is given below:



SECOND MODEL: KNN (K NEAREST NEIGHBOUR)

K-Nearest Neighbour, often known as KNN. Is another model that is used in the type of classification algorithms. It would help in drawing out certain conclusion on the given data after its training is complete.

I chose it as my second model as I had taken decision tree as the first one. And in the world of classification algorithms these two models are very famous.

Below given is the process that I went through to create my KNN model. But first let's look at the KNN model made with all the default values:

-> *knn = KNeighborsClassifier()*

knn.fit(xTrain, yTrain)

forcast = knn.predict(xTest)

accuracy_score(yTest, forcast)

One thing to note is that I kept the xTrain and yTrain datasets same as I did for the Decision Tree Classifier. And took the train and test ratio as 60% to 40%.

After running the above script, I got an accuracy of 90%.

Please see the table below for different values of parameters used in the tuning of KNN model.

Parameters	Values
n_neighbors	2
p_scores	2
Algorithm	auto
Weights	uniform

So, after calculating and tuning all the above parameters I was able to formulate the following script and got the accuracy as 93%:

-> *knn = KNeighborsClassifier(algorithm='auto', n_neighbors=2, p=2, n_jobs=4, weights='uniform')*

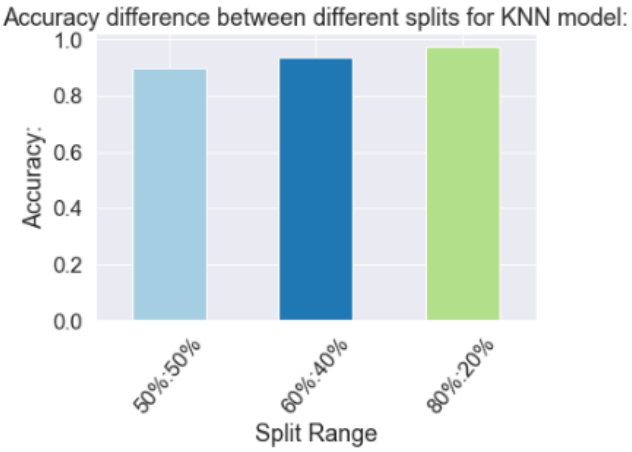
knn.fit(xTrain, yTrain)

forcast = knn.predict(xTest)

accuracy_score(yTest, forcast)

Also, a respective confusion matrix and a classification report has been generated for my KNN model and put in my python workbook.

-> Before finalising this split ratio, I tried t split to make this model with three different split ratios: 50:50, 60:40 and 80:20, respectively. The graph comparison of the achieved accuracy is given below:



DISCUSSION:

This part would shed some light on the discussion of the report. As asked in the assignment specifications, I was able to cure, prepare, analyse and study the data set that I chose, while using different techniques and processes like generating graphs, creating two different models, using Decision tree classifier and K Nearest Neighbours.

CONCLUSION:

I was able to draw out a conclusion that, for the data set that I took for my project, Mice Protein Dataset, the KNN model gives a better accuracy than a decision tree classifier.

Again, this is just my opinion that has been drawn out of the results that I got from following the different processes of the data science on the data set that I chose.

Given below is a graph that shows the difference in the accuracy achieved by two different models. KNN and Decision Tree Classifier.

REFERENCES:

1. Mithrakumar, M., 2020. *How To Tune A Decision Tree?*. [online] Medium. Available at: <https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680> [Accessed 9 June 2020].
2. Gupta, P., 2020. *Decision Trees In Machine Learning*. [online] Medium. Available at: <https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052> [Accessed 9 June 2020].
3. Medium. 2020. *Decision Trees In Machine Learning*. [online] Available at: <https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052> [Accessed 9 June 2020].
4. Albhai, E., 2020. *Building A K-Nearest-Neighbors (K-NN) Model With Scikit-Learn*. [online] Medium. Available at: <https://towardsdatascience.com/building-a-k-nearest-neighbors-k-nn-model-with-scikit-learn-51209555453a> [Accessed 9 June 2020].
5. Paruchuri, V., 2020. *Tutorial: K Nearest Neighbors (KNN) In Python - Dataquest*. [online] Dataquest. Available at: <https://www.dataquest.io/blog/k-nearest-neighbors-in-python/> [Accessed 9 June 2020].
6. RMIT Sem1, 2020, Practical Data Science, Lectures and Tutorials.

End.