EPQ Artefact Report

Which Machine Learning Algorithm is Most Effective and Accurate at Diagnosing Thyroid Disease: A Naive Bayes Algorithm or a Deep Neural Network?

Saatvik Kambhampati
(2021)

Contents

Section Heading

Abstract - Machine Learning (ML) is a subfield of Artificial Intelligence focused on using mathematics and statistics to emulate the learning of humans'. ML can be achieved through the use of many different algorithms and methods. This Extended Project Qualification (EPQ) Artefact will investigate whether a Gaussian Naïve Bayes (GNB) algorithm or a Deep Neural Network (DNN) algorithm provides better predictions in diagnosing thyroid disease. The two algorithms will attempt to predict whether any given patient is "hypothyroid", "hyperthyroid", or "negative" (ie, has neither). Both algorithms have been programmed in Python. The GNB algorithm has been programmed using NumPy and Pandas, in addition to the Python standard libraries. The DNN algorithm has been programmed here using Tensorflow, as well as NumPy Pandas, and standard libraries.

I. Introduction

Machine learning (a branch of Artificial Intelligence) is the process of developing algorithms that use statistical equations, concepts, and data to emulate the learning of humans' in order to categorise data *("Machine Learning", Anon., 2020), (Tamir, 2020)*.

Some real-world applications of machine learning include: speech recognition (eg: virtual assistants such as Siri or Google Assistant), computer vision, and automated stock trading *("Machine Learning", Anon., 2020)*

Machine learning is also used in various other fields, such as marketing, healthcare, retail, and many others *(Tamir, 2020)*.

I.1. ML Algorithms - A computer algorithm is a precise step-by-step series of rules that leads to a solution to the original problem *(Weber, 2020)*. An ML algorithm is similar, but here, certain parameters are adjusted as the algorithm "learns" leading to predictions of increasing accuracy.

In this project, ML algorithms predict a diagnosis (called a "prediction") for patients (called "instances") *("Input-2017", Anon., 2017)*. The diagnoses predicted could be "hypothyroid" (indicating an underactive thyroid gland), "hyperthyroid" (indicating an overactive gland), or, "negative" (indicating a gland with a normal level of thyroid hormone production). Within each dataset, a given patient (or

instance) will have an actual or correct diagnosis (or "label") given by a medical expert. The predictions by the ML algorithm are matched against the actual labels given by the experts. A label (sometimes called "class") is therefore a set that categorises all patients (or instances) in the dataset.

At its core, machine learning has three core components *(Tamir, 2020)*:

1) **The decision process:** This is the stage at which the algorithm will carry out certain computations and then make a prediction for the current patient/instance.

2) **An error function:** A function that answers the following questions:
   - Was the prediction correct compared to the actual label/diagnosis?
   - If the prediction was wrong, the function will then carry out the quantification of the error.

3) **The update/optimisation process:** Using the above quantification, the algorithm will update certain parameters within itself so that it yields more accurate results for the next instance of the training data set.

This project will compare the predictions of two ML algorithms, the Gaussian Naive Bayes (GNB) and the Deep Neural Networks (DNN) algorithms with the labels given by experts. A comparison will also then be made between the two algorithms.

I.2. Why a Medical Dataset? - Complex decisions need to be made in diagnosing patients and a lot of data (including age, sex, certain hormone levels, etc) may need to be used in this process. Hence medical datasets are heterogenous (meaning that they can hold many types of data) often with complex relationships within the dataset. Thyroid disease datasets have many of these characteristics making them ideal candidates for developing effective and accurate predictive models based on machine learning algorithms. Hence such a dataset was used for this project. The thyroid dataset used in this project is open-source and available at: https://archive.ics.uci.edu/ml/datasets/Thyroid+Disease

I.3. The Thyroid Gland and Thyroid Diseases - The thyroid gland is part of the endocrine system and is situated in the front of the neck. It produces two hormones: thyroxine (T4) and triiodothyronine (T3) *(NHS 1, n.d.)*. Both of these hormones are important for regulation of metabolism in the human body. The production of hormones by the thyroid gland can sometimes be affected by disease. This could lead to under or over production of T3 and T4. Hypothyroidism is when the gland is not producing enough T3 and T4 *(NHS 2, n.d.)*. On the other

hand, hyperthyroidism is when the gland produces too much T3 and T4 (*NHS 1, n.d.*).

I.4. Datasets - A dataset is a structured collection of data that holds common information categories (or "attributes") *("Input-2017", Anon., 2017)* between all patients or instances. For example, a dataset might have 1,000 instances and have sex, height, weight, and foot size as attributes for each instance. For each instance, an observation of attributes is recorded. A data set example might look like in Table 1 below where each row is an instance, and each column is an attribute:

Table 1

Sex	Height	Weight
male	100	60
female	90	60

Datasets may be training or testing datasets. A training set is a section of the dataset that the algorithm will use to "learn". Any algorithm learns using a training dataset ("training set"). ML algorithms will make a prediction for all instances in the training set and compare these predictions with the correct (expert-given) label for each instance. If the algorithm classified the instance incorrectly, it would adjust the parameters to account for this, to enable it to be more accurate with its later predictions.

Once the machine learning algorithm has learnt from the training dataset, it is tested using a testing dataset. It is then ready to be used.

II. Methodology

II.1. Data Description - When data is downloaded, it comes with a file in which a general description of the data, the attributes, and their types is given (see table 2). The download also comes with two datasets, a training dataset and a testing dataset. The training and testing datasets in the database used in this project had 5,600 and 1,944 instances (people) respectively.

Table 2 shows the 28 attributes that were originally in each dataset when downloaded. Of the 28, 7 are continuous and 21 are boolean. A continuous attribute can hold any real number value (e.g., 5 or 16.57347), but a boolean attribute can only be one of two values (e.g., True or False). In the table below, the column 'Variable Type' specifies what type of value a particular variable is and what type of mathematical, relational, or logical operations can be applied to it (in this case, could be 'continuous' or 'boolean'). *("data type", Anon., 2016)*.

Table 2

Feature/Attribute	**Variable Type**
Age	Continuous
Sex	Boolean
Thyroxine	Boolean
Query on Thyroxine	Boolean
Antithyroid Medication	Boolean
Sick	Boolean
Pregnant	Boolean
Thyroid Surgery	Boolean
I131 Treatment	Boolean
Query on Hypothyroid	Boolean
Query on Hyperthyroid	Boolean
Lithium	Boolean
Goitre	Boolean
Tumour	Boolean
Hypopituitary	Boolean
Psych	Boolean
TSH Measured	Boolean
TSH (Level)	Continuous
T3 Measured	Boolean
T3 (Level)	Continuous
TT4 Measured	Boolean
TT4 (Level)	Continuous
T4U Measured	Boolean
T4U (Level)	Continuous

FTI Measured	Boolean
FTI (Level)	Continuous
TBG Measured	Boolean
TBG (Level)	Continuous

In order to make the training more efficient, certain attributes were removed from the downloaded dataset for the purpose of this project. For example the boolean attributes other than sex have been removed as they had little or no effect on the predictions of either algorithm.

II.2. Data Cleaning - Data must be "cleaned" (or "pre-processed") before the algorithms can be used. This involves removing or modifying data that is missing, incomplete, or irrelevant *(Gimenez, 2020)*. Data cleaning is an essential step to any machine learning or data science project because, if data is incorrect, the algorithms that use the data become unreliable (even if they may seem "correct").

There was at least one attribute missing in 1,707 of 5,600 instances of the training dataset and 552 of 1,944 instances of the testing dataset. These instances were removed during data cleaning leaving the training dataset with 3,893 instances of data and the testing dataset with 1,392. A Python script was created to execute this cleaning.

It is easier for the algorithms to identify and predict labels for each instance correctly if the total number of labels are fewer. The downloaded dataset had nine labels in total. For this project, the original labels were re-grouped into three new labels as shown in Table 3.

Table 3

Labels on dataset as it was when downloaded	**Labels**
Hypothyroid, Primary Hypothyroid, Compensated Hypothyroid, Secondary Hypothyroid	**Hypothyroid**
Negative	**Negative**
Hyperthyroid, T3 Toxic,Goitre, Secondary Toxic	**Hyperthyroid**

II.3. The Machine Learning Algorithms - Two algorithms will be investigated, a Naive Bayes algorithm using a Gaussian Normal Distribution (referred to as "GNB" algorithm) and a Deep Neural Network (referred to as "DNN" algorithm).

II.3.1. Gaussian Naïve Bayes (GNB) - A GNB algorithm uses conditional probability to determine the probability of each possible label. Given a set of observations of an instance, the algorithm predicts the probability of the instance belonging to a particular label. The label which has the highest probability is then the label under which that instance is classified. This is also known as the 'maximum likelihood' approach.

The Naïve Bayes uses a Gaussian function to determine the conditional probability of a continuous variable against a discrete one, using a normal distribution. Let us suppose that a training set is split into n labels. The set of labels, L, can be expressed as:

$$L = \{l_1, l_2, l_3, ..., l_n\} \quad (1)$$

The dataset also contains a continuous attribute denoted as the random variable X. The set is split into n labels [eg., ~4,000 instances can be split into groups of three (as there are three labels)]. The mean, μ_k, and variance, σ_k^2, of X is calculated for each label l_k separately.

Then, an observation, v, for the attribute X is collected. We use the following equation to calculate $p(X = v \mid l_k)$ (i.e., the probability that $X = v$, given label l_k, and where $p(...)$ denotes a Gaussian probability function *(Brownlee, 2020)*:

$$p(x = v \mid l_n) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}} \quad (2)$$

For example, in this experiment, the attribute X is age. Taking the first instance of the training data, the observation v of X is 41. So, we can say:

$$\begin{gathered} p(Age = 41 \mid Hypothyroid) \\ p(Age = 41 \mid Hyperthyroid) \\ p(Age = 41 \mid Negative) \end{gathered} \quad (3)$$

Equation 2 will produce a conditional probability of an observation of a particular continuous attribute given that a specific label is true.

After this, the evidence, E, must be determined. The evidence is calculated as follows:

$$E = (p(l_1) * p(x \mid l_1) * p(x_1 \mid l_1) * ... \cdot * p(x_k \mid l_1))$$

$$+\ldots+\ (p(l_n)\ *\ p(x\mid l_n)\ *\ p(x_1\mid l_n)\ *\ldots*\ p(x_k\mid l_n)) \tag{4}$$

Next, $posterior(l_n)$ is calculated with the following equation:

$$posterior(l_n) = \frac{p(l_n)*p(x\mid l_n)*p(x_1\mid l_n)*\ldots*p(x_k\mid l_n)}{evidence} \tag{5}$$

This is done for all labels.
Finally, to determine the prediction, we find the greatest value out of

$$\{\ posterior(l_1),\ posterior(l_2),\ \ldots,\ posterior(l_n)\ \} \tag{6}$$

Thus for example, if,

$$\begin{gathered} posterior(Hypothyroid)\ >\ posterior(Hyperthyroid) \\ AND \\ posterior(Hypothyroid)\ >\ posterior(Negative) \end{gathered} \tag{7}$$

then, the algorithm will predict that instance to have the label “hypothyroid” as the probability of this being hypothyroidism is greater than the probability of it either being hyperthyroidism or being neither (“negative” label).

II.3.2. Deep Neural Network (DNN) - An artificial neural network (ANN) takes a certain number of inputs and passes them through a series of synapses (each with a weight) which feed into neurons, also called nodes. (A node is just a point in the network where computation occurs, *(Nicholson, 2020)*). These nodes will then use other synapses to feed into other nodes until the final probability is sent to the output layer, as shown in Figure 1. The output layer produces probabilities for each label based on the inputs.

There are various definitions for the term “deep neural network”. One is that an artificial “neural network with *some level* of complexity” and “at least two [hidden] layers” *(Johnson, 2020)* is classed as “deep”. However, another is that a deep neural network consists of more than three layers (including input and output layers) *(“Machine Learning”, Anon., 2020)*. This latter definition is endorsed by the International Business Machines Corporation and will be used here.

ANNs are composed of various node layers, which are: The input layer, one (or more) hidden layers), and an output layer. Each node in one layer is connected to every single node in the next layer.

Following the production of probabilities, the algorithm will determine the highest probability out of all the outputs, and predict the instance with the label that has the highest probability of being true.

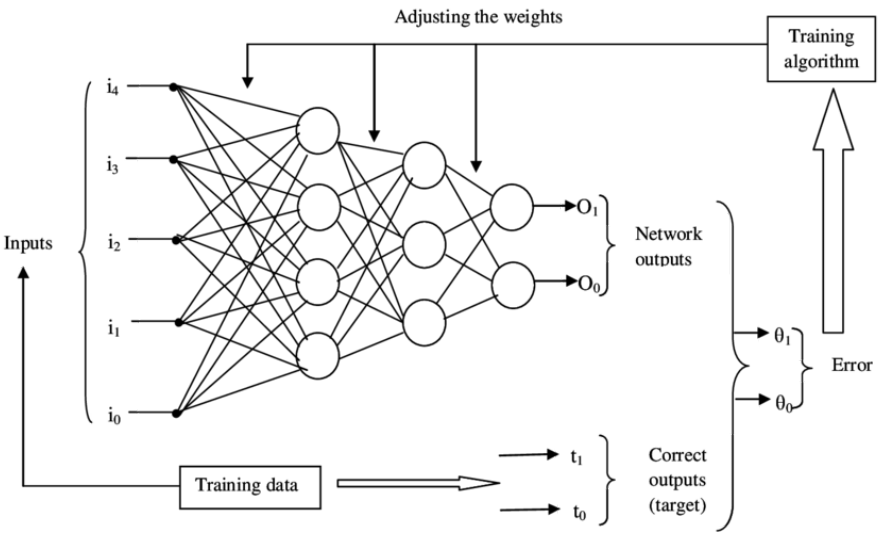


Figure 1

Figure 1 is a topology of a neural network. The layers of nodes between the input and output layers are known as “hidden layers”. They carry out all of the main calculations for determining the label probabilities.

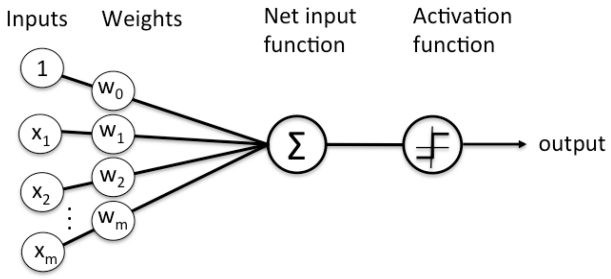


Figure 2 (Nicholson, 2020)
This figure shows how the outputs from the last layer act as the inputs to the next layer when multiplied by their product and the sum is found.

As shown in figure 2, each node produces an output by combining “input from the data with a set of weights” (from the synapses) “that either amplifies or dampens that input” *(Nicholson, 2020)*. These weighted inputs are then passed through an “activation function” to produce an output for the node *(Nicholson, 2020)*.

Activation functions are functions that can be used to get the output of a node. Activation functions can also be called "Transfer Functions". These functions can be categorised into two types *(Sharma, 2017)*:

1. **Linear activation functions**. These functions are a straight line when graphed and will follow the $y = mx + c$ general formula (where m and c is the gradient and the y-intercept of the line, respectively).

2. **Non-linear activation functions.** These are functions that do not produce a straight line when graphed. Non-linear activation functions can be further split into two categories: Differentiable and Monotonic functions. Differentiable functions are ones that can be differentiated and where the derivatives are finite. Monotonic functions are either entirely "non-increasing" or "non-decreasing" functions.

The sigmoid function is a differentiable, continuous, non-linear activation function. It is often used in ANNs when the output values are probabilities. Both the Sigmoid Function and probabilities only exist between $0 \leq S(x) < 1$, (as shown in figure 3, and equation (9)). Hence, Sigmoid is the appropriate choice of activation function for this project, particularly as probabilities also only exist within this range. *(Sharma, 2017)*.

$$S(x) = \frac{1}{1+e^{-x}} \tag{8}$$

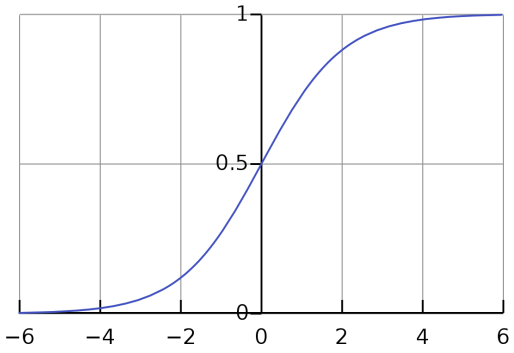


Figure 3
Figure 3 is a graph of the Sigmoid Function.

The DNN "learns", by comparing the predicted output with the actual label, and changing the weights of each synapse to yield better results next time.

III. Results

The effectiveness of an ML algorithm can be measured through the accuracy of its prediction of labels. But, although accuracy is an important performance metric, it

is not thought to be the most revealing metric in a clinical context. What matters more is the number of true positives, true negatives, false positives and false negatives that exist within the predictions (see Table 4). This, for a clinical dataset, is reflected in the sensitivity and specificity rates, often referred to as the 'gold standard' *(Swift, Heale, and Twycross, 2019)*.

Table 4

Term	**Definition**
True Positive (TP)	If the patient actually has the disease and the algorithm predicts that they have the disease.
True Negative (TN)	If the patient does not actually have the disease, the algorithm predicts that they do not.
False Positive (FP)	If the patient does not actually have the disease, but the algorithm predicts that they do.
False Negative (FN)	If the patient actually has the disease, but the algorithm predicts that they do not.

(Swift, Heale, and Twycross, 2019)

The results from the algorithm are arranged in a confusion matrix, as shown in table 5. A confusion matrix is a table that "describes the performance of a classification model (or "classifier") on a set of test data for which the [true/actual] values are known." *(Markham, 2014)*

Table 5

	Positives (Predicted)	**Negatives (Predicted)**
Positives (Actual)	Number of "true positive" cases	Number of "false negative" cases
Negatives (Actual)	Number of "false positive" cases	Number of "true negative" cases

From the confusion matrix, specificity and sensitivity can be obtained as follows:

$$Specificity = \frac{T_n}{T_n + F_p} \quad (10),$$

where T_n is the number of true negatives, and F_p is the number of false positives.

$$Sensitivity = \frac{T_p}{T_p + F_n} \quad (11),$$

where T_p is the number of true positives, and F_n is the number of false negatives.

Thus sensitivity is "the ability to correctly identify patients with a disease". In this case, patients with hypothyroidism or hyperthyroidism. Specificity is "the ability … to correctly identify people without the disease". In this case, patients who are negative for hypothyroidism and hyperthyroidism. *(Swift, Heale, and Twycross, 2019)*.

III.1. GNB - The GNB algorithm was programmed in Python (the code can be found in Appendix D). Once the GNB algorithm had been fully programmed and trained, it was tested and the following results were obtained:

1. **94.32% of cases** were correctly predicted
2. **5.68%** of cases were incorrectly predicted.

These results were then represented as a Confusion Matrix (using the Pandas library). Table 6 gives the Confusion Matrix for this algorithm with raw numbers (ie, not percentages). It is a confusion matrix of the predicted hypothyroid, hyperthyroid, and negative cases compared with the actual diagnoses of hypothyroid, hyperthyroid, and negative cases.

		Predicted			
Actual		**Hyperthyroid**	**Hypothyroid**	**Negative**	**All**
	Hyper	4	0	12	16
	Hypo	0	11	48	59
	Negative	8	11	1298	1317
	All	12	22	1358	1392

Table 6: Confusion Matrix for GNB Algorithm

From the table, it can be seen that out of all actual hypothyroid cases, 18.64% (11/59) were predicted correctly. Out of all actual hyperthyroid cases, 25% were

predicted correctly. Out of all actual negative cases, 98.56% were predicted correctly.

Table 5 is a confusion matrix of predicted positives and negatives compared with actual positives and negatives.

Predicted →	**Positives**	**Negatives**
Positives	15 (TP)	60 (FN)
Negatives	19 (FP)	1298 (TN)

Table 7

TP - True positive, TN - True negative, FN - False negative, FP - False positive

Using equation 10 and table 7, the specificity of the GNB algorithm can be found to be $0.9855732726,$ *or* 98.55732726%.

Using equation 11 and table 7, the sensitivity of the GNB algorithm can be found to be 0.2, or 20%.

III.2. DNN - The DNN algorithm was programmed in Python using the Tensorflow library (the code can be found in Appendix E). Once the DNN algorithm had been fully programmed, and trained, it was tested and the following results were obtained:

1. **94.47%** of cases were predicted correctly.
2. **5.53%** of cases were predicted incorrectly.

		Predicted			
Actual		**Hyper**	**Hypo**	**Negative**	**All**
	Hyper	0	0	16	16
	Hypo	0	15	44	59
	Negative	6	11	1300	1317
	All	6	26	1360	1392

Table 8: Confusion Matrix for DNN Algorithm

Key: 0 - Hyperthyroid positive,1 - Hypothyroid positive, 2 - Negative for both

From table 8, it can be seen that out of all hypothyroid cases, the DNN predicted 15 out of 59 cases correctly (25.4% accuracy rate for hypothyroidism). However, it predicted none of the actual hyperthyroid cases correctly (and instead mistook them for negative cases). Finally, it predicted ~98.71% of negative cases correctly.

Table 9 is a confusion matrix of predicted positives and negatives compared with actual positives and negatives.

Table 9

Predicted →	**Positives**	**Negatives**
Positives	15 (TP)	60 (FN)
Negatives	17 (FP)	1300 (TN)

Using equation 10 and table 9, the specificity of the DNN algorithm can be found to be 0.9870918755, or 98.70918755%..

Using equation 11 and table 9, the sensitivity of the DNN algorithm can be found to be $0.2, \ or \ 20\%$.

The results for both the DNN and GNB are caused by, in part, bias and imbalance in the dataset, due to the complexities of clinical data (as mentioned in Section I.2). This will be discussed more in-depth in V.1.

IV. Evaluation and Conclusion

From the results above, it is seen that the DNN algorithm has a marginally higher (by ~0.15%) overall accuracy compared to the GNB Algorithm.

However, neither algorithm had a high true positive rate indicating that neither was really able to predict a reasonable number of positive cases correctly. The reason for this is bias. Bias occurs when the training data sets used have a significantly larger proportion of instances that are labelled with certain classifications (“majority classes”) compared to others (“minority classes”). This leads to the algorithm tending to favour the prediction of majority classes over minority classes. This can result in an increased number of predicted labels to be false positive or false negative, which is not desirable. For example, in the training set here, there were 3,677 instances (out of 3,893) that were predetermined as negative, and only 216 positive (hypothyroid cases + hyperthyroid cases) instances. From this, we can find the *prevalence*. The prevalence is how often positive cases occur in the dataset *(Markham, 2014)*. So, for the training set, we can find this

prevalence to be 5.55%, which is extremely low. This means that, because of the spread of data within the training set, the algorithm has not had a chance to learn to predict positive cases accurately, but has learnt to predict negative cases accurately. Hence it will tend to "want" to predict more negatives than positives, leading to more false negatives.

To overcome this bias, techniques such as SMOTE (Synthetic Minority Over-sampling Technique) are being developed and used. SMOTE works by creating new samples from minority classes, using linear interpolation *(Brownlee, 2015)*. This means that the dataset classes will be balanced, so bias is eliminated.

However, a conclusion cannot be drawn based solely on the overall accuracy of either algorithm. This is because, within a medical context, the specificity and sensitivity represents an algorithm's performance more than most other metrics *(Swift, Heale, and Twycross, 2019)*.

For the reasons mentioned above, both algorithms were much better at predicting negative cases well though, with a high true negative rate. This led to both algorithms having a high specificity. The DNN had a marginally higher specificity by ~0.152%. This means that the DNN was marginally more able to correctly identify patients without either disease. This was expected as the DNN predicted 1300 negative cases correctly, compared to the GNB with 1,298.

As for sensitivity, both the GNB and DNN algorithms had a sensitivity of only 20%, meaning that they both had the same (albeit low) overall ability to diagnose patients with either hypothyroidism or hyperthyroidism. However, it must also be noted that the GNB algorithm was better at diagnosing patients with hyperthyroidism (it correctly diagnosed 25% of hyperthyroid-positive patients, whilst the DNN diagnosed 0%). Whereas the DNN predicted no hyperthyroidism cases correctly, it did manage to predict 6.78% more hypothyroid-positive patients than the GNB algorithm. The reason for this difference in prediction between the two algorithms is possibly because of the basic differences between the two algorithms and could be explored in a further project.

Because the difference in specificity is insignificant, both algorithms are as accurate as each other and are as effective at diagnosing negative patients correctly as each other. When it comes to sensitivity, it can be concluded that although neither is highly sensitive, the DNN algorithm is more accurate and effective at diagnosing hypothyroidism, but the GNB algorithm is more accurate and effective at diagnosing hyperthyroidism.

V. Reference List/Bibliography

1) Anonymous, (2016) “data type”. Available at: https://searchapparchitecture.techtarget.com/definition/data-type (Accessed at: 7 December 2021)

2) Brownlee, J. (2015) “8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset”. Available at: https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/ (Accessed at: 15 December 2021)

3) Brownlee, J., (created 2016, updated 2020) “Naive Bayes for Machine Learning”. Available at: https://machinelearningmastery.com/naive-bayes-for-machine-learning/ (Accessed At: 12 September 2021)

4) Chinese University of Hong Kong (2017), “Input-2017”. Available at: https://www1.se.cuhk.edu.hk/~seem5470/lecture/Input-2017.pdf (Accessed at: 7 December 2021)

5) Markham, K., (2014) “Simple guide to confusion matrix terminology”. Available at: https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/ (Accessed: 21 November 2021)

6) Gimenez, L. (2020) “6 steps for data cleaning and why it matters”. Available at: https://www.geotab.com/blog/data-cleaning/ (Accessed at: 7 December 2021)

7) Anonymous, (2020) “Machine Learning”. Available at: https://www.ibm.com/cloud/learn/machine-learning (Accessed at: 15 December 2021)

8) Johnson, J. (2020) “What’s a Deep Neural Network? Deep Nets Explained”. Available at: https://www.bmc.com/blogs/deep-neural-network/ (Accessed: 29 August 2021)

9) Nicholson, C., (2020) “A Beginner's Guide to Neural Networks and Deep Learning”. Available at: https://wiki.pathmind.com/neural-network (Accessed: 29 August 2021)

10) Referred to as “NHS 1”: Anonymous, (n.d.) “Overactive Thyroid (hyperthyroidism)”. Available at: https://www.nhs.uk/conditions/overactive-thyroid-hyperthyroidism/ (Accessed: 29 August 2021)

11) Referred to as “NHS 2”: Anonymous, (n.d.) “Underactive Thyroid (hypothyroidism)”. Available at: https://www.nhs.uk/conditions/underactive-thyroid-hypothyroidism/ (Accessed: 29 August 2021)

12) Sharma, S., (2017) “Activation Functions in Neural Networks”. Available at: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 (Accessed: 12 September 2021)

13) Swift, A., Heale, R., Twycross, A., 2019. "What are sensitivity and specificity?, vol. 243, pp. 2 - 4

14) Tamir, Michael (2020) “What Is Machine Learning (ML)?”. Available at: https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/ (Accessed at: 10 January 2022)

15) Weber, L. M., (2020) “Explainer: What is an algorithm?”. Available at: https://www.sciencenewsforstudents.org/article/explainer-what-is-an-algorithm (Accessed at: 28 November 2021)

Appendix A: Project Evaluation

What have I learned from completing this project?

I identified my project and learnt how to structure an EPQ (i.e. how to organise and plan it using mind maps, trello, and other graphical forms). I also learnt to actively schedule meetings with supervisors, and make modifications to the project.

Furthermore, I learnt how to understand and review machine learning literature, how to research information and carry out an analysis on both my project and my resources and write a project report. I also learnt many new skills from completing my presentation such as communication skills and how to use LaTeX for mathematical equations. I have learnt how to evaluate my project by identifying strengths and weaknesses so that I can rectify them in the future. I really enjoyed doing an independent project with support from my supervisor.

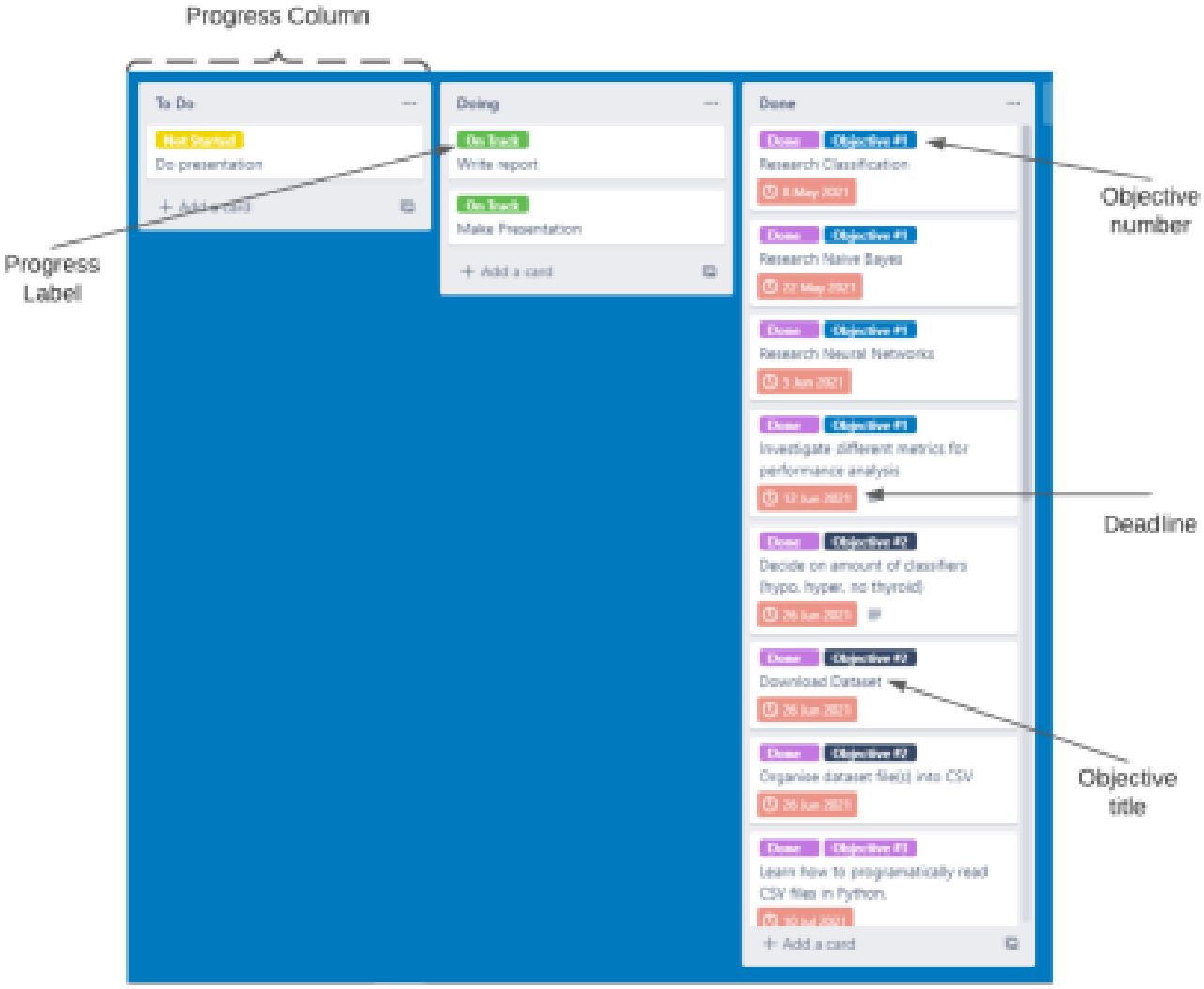
What new knowledge of expertise have I enjoyed or found valuable?

My EPQ project required me to learn many new concepts in Mathematics. These new concepts built on the knowledge that I had acquired on my A level Maths course and helped solidify as well as practically apply ideas and concepts, such as using conditional probability to enable the Naive Based MLA to make predictions. I have also gained new knowledge in the field of data science and machine learning, such as how machine learning works and the process to develop these algorithms.

What are the strengths & weaknesses of my project (including planning & organisation)?

A huge strength of my project was my time management. For my project, I managed and organised my time using an app called Trello. On Trello, a work-board can be made, with various columns. My columns included: “To Do”, “Doing”, and “Done”. Each column can have any number of tasks. I made a list of tasks and to each task, I assigned a title, a deadline date, and a label (either “Not Started”, “On Track”, “Delayed”, or “Done”). As each task progressed, I moved it to the next column, so I could keep track of how much of the overall EPQ I had completed. The label assigned to each task allowed me to keep track of whether I could finish that particular task on time (i.e. on or before the pre-set deadline). If it was delayed, and it seemed like it would not be completed in time, I adjusted the deadline of the task (and the deadlines of other tasks) accordingly.

Below is an image of my Trello board.



Following completion of the project, I realised that for both algorithms, the training parameters were not fully optimised. To better optimise, the DNN MLA could be run through 50,000 epochs instead of the 5000 used here. If this had been done, the results could possibly have been even more accurate.

What skills have I improved?

I have vastly improved my skills in both data science and Python, the programming language which I used to program the ML algorithms. I have developed many new skills in data science, such as data cleaning. Furthermore, I have greatly increased my proficiency in Python, learning how to efficiently use modules such as TensorFlow, NumPy, and Pandas. Each of these modules are very common within the world of data science and will be very useful in the future especially as I am planning to do an undergraduate degree in Computer Science.

I gained many generic skills during this EPQ. I learnt how to organise and arrange meetings with my supervisor during a tight school schedule and use the advice he gave to adapt my project and manage my time. An example of this is when I presented my supervisor with my objectives and deadlines, and he suggested that I create a graphical form of the plan to easily keep track of my progress.

I used this idea to create a Trello board (explained above). Using the Trello board, I learnt how to plan my project and organise my schedule so that I had enough time to finish each task. I also learnt that keeping some slack between deadlines was essential as sometimes certain objectives get delayed due to unavoidable reasons.

My presentation enabled me to improve my skills in Google Slides. Due to the technical nature of my EPQ, I had to display many equations. For this, I had to learn how to use LaTeX, which is a document mark-up language that allows lines of code to be translated as mathematical equations in an image file.
As I practised and modified my presentation, my public speaking & communication skills increased significantly, along with my confidence. I have also increased my skills in writing research reports by learning the format and the formal English that should be used. While writing the report, I developed skills in researching, looking for reliable information, and reading the available literature. I also learnt how to reference using the Harvard Referencing System.

What changes would I make if I undertook such work again?

I would have done further research into Deep Neural Networks and attempted to build one from scratch (and not using TensorFlow) as this would have given me a deeper understanding of the algorithm and allowed me to put the theory of the algorithm to practice. However, the time available this time did not allow for this.

I also would have researched the whole process of building a machine learning algorithm before I started programming the MLAs. This would have enabled me to carry out data cleaning before programming the algorithms. Doing this would have saved me a couple of days of time.

Also, since there was a large amount of bias in the training dataset, if I undertook this project again, I would use techniques to decrease this bias, such as SMOTE, as described in my report and presentation. This would have likely increased the accuracy of both of the algorithms significantly, however time constraints did not allow me to learn and use these techniques this time.

What advice would I give to others undertaking such a project?

My advice would be to finish more of the project before the start of Year 13. Even though I finished the actual artefact before school restarted in September, I could

not finish writing the report as early as I wanted to. This is because I had several other commitments at the start of Year 13 including my UCAS application. Finishing more of my project over summer (such as the report) would have made me feel less overwhelmed.

What difficulties did I experience and how did I overcome them?

One difficulty I experienced was with understanding and then programming the Naive Bayes Algorithm. I could not understand what the training parameter for the algorithm was. I overcame this issue by reading many different articles about how the algorithm works, and simply trying to practically program the algorithm. My research and the practical experience of programming the algorithm helped me understand that the training parameter was the P(Label) and this helped me progress.

Another experience I had was the process of building an ML Algorithm. I had initially thought that data cleaning was done after the algorithms were programmed. After struggling with attempting to figure out how to take inputs from a dataset that wasn't cleaned yet, I eventually settled on carrying out the data cleaning before the programming of the algorithms, so that I would know in advance what the dataset attributes would be.

Are my final outcomes/conclusions originally what I intended or are they different?

My final EPQ research project has progressed and concluded as per my original intention. I am very happy with my final conclusions and the outcome of my project. I have completed what I set out to do, which was to program two ML algorithms (I had no prior knowledge of machine learning when I started) and to find out which was more accurate at diagnosing Thyroid Disease. Although there is some room for improvement with the execution of the programming of the algorithms, valid conclusions were reached and the project was completed to my overall satisfaction.

Appendix B: Source Evaluation

1) Anonymous (2016) “data type”:
    - **Source:** Anonymous, (2016) “data type”. Available at: https://searchapparchitecture.techtarget.com/definition/data-type (Accessed at: 7 December 2021)

    - **Information Gathered:**
      The definitions of a data type and boolean values.

    - **Why it was Useful:**
      It gave concise definitions of the term “data type” and various different data types.

    - **Reliability & Limitations of the Source:**
      The source is a thread on a forum posted by an anonymous user, so credibility cannot be verified and is therefore not very reliable.

    - **How I would Mitigate Those Limitations and Increase its Reliability in the Future:**
      To increase the source’s reliability, I would attempt to find a different source published by a specific user or author, so that this secondary source has credibility.

2) Brownlee, J., (2015) “8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset”:
    - **Source:** Brownlee, J., (2015) “8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset”. Available at: https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/ (Accessed at: 15 December 2021)

    - **Information Gathered:**
      I used to explain a technique to mitigate bias (“SMOTE”).

    - **Why it was Useful:**
      Brownlee gives various ways to “combat imbalance” within datasets in an attempt to decrease bias.

    - **Reliability & Limitations of the Source:**
      This source is reliable as the website is quite popular and the author, Jason Brownlee, has a masters degree and PhD in Artificial Intelligence.

However, a limitation of this source is that the website is owned by one person, Brownlee, so the articles are not peer-reviewed for accuracy and validity.

- **How I would Mitigate Those Limitations and Increase its Reliability in the Future:**
  One way to mitigate this limitation is to use a source from a peer-reviewed journal. However, a problem with this is that many of them would use complex terminology and would be difficult for beginners to understand.

3) Brownlee, J., (2020) “Naive Bayes for Machine Learning”:
   - **Source:** Brownlee, J., (2020) “Naive Bayes for Machine Learning”. Available at: https://machinelearningmastery.com/naive-bayes-for-machine-learning/ (Accessed At: 12 September 2021)

   - **Information Gathered:**
     - Bayes’ Theorem and an explanation of how it is used in machine learning algorithms.
     - Maximum likelihood.
     - The Gaussian Equation.
     - Used to reference the Gaussian equation and the fact that continuous data can be assumed to follow a gaussian/normal distribution.

   - **Why it was Useful:**
     It introduced me to the naive bayes algorithm and the mathematical concepts required to understand it.

   - **Reliability & Limitations of the Source:**
     This source is reliable as the website is quite popular and the author, Jason Brownlee, has a masters degree and PhD in Artificial Intelligence.

     However, a limitation of this source is that the website is owned by one person, Brownlee, so the articles are not peer-reviewed for accuracy and validity.

   - **How I would Mitigate Those Limitations and Increase its Reliability in the Future:**
     One way to mitigate this limitation is to use a source from a peer-reviewed journal. However, a problem with this is that many of

them would use complex terminology and would be difficult for beginners to understand.

4) Anonymous, (2017) “Input-2017”:
    - **Source:** Anonymous, (2017) “Input-2017”. Available at: https://www1.se.cuhk.edu.hk/~seem5470/lecture/Input-2017.pdf (Accessed at: 7 December 2021)

    - **Information Gathered:**
      Two definitions: One for attributes and one for instances.

    - **Why it was Useful:**
      It gives simple formal definitions for these two terms.

    - **Reliability & Limitations of the Source:**
      The source is reliable as it is a department from the Chinese University of Hong Kong.

    - **How I would Mitigate Those Limitations and Increase its Reliability in the Future:**
      Not applicable.

5) Markham, K., (2014) “Simple guide to confusion matrix terminology”:
    - **Source:** Markham, K., (2014) “Simple guide to confusion matrix terminal terminology”. Available at: https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/ (Accessed: 21 November 2021)

    - **Information Gathered:**
        - What confusion matrices are and how to read them.
        - What prevalence is.

    - **Why it was Useful:**
      It allowed me to learn how to represent data in a compact, yet informative way. It also taught me about the term “prevalence” which enabled me to easily describe the quality of the data using a single number.

    - **Reliability & Limitations of the Source:**
      The author of website and article, Kevin Markham, has a degree in Computer Engineering and has previous experience in Data Science, so this source is fairly reliable

One limitation of this source is that the website is owned by one person, Kevin Markham, so the articles are not peer-reviewed for accuracy and validity.

- **How I would Mitigate Those Limitations and Increase its Reliability in the Future:**
  One way to mitigate this limitation is to use a source from a peer-reviewed journal. However, a problem with this is that many of them would use complex terminology and would be difficult for beginners to understand.

6) Gimenez, L. (2020) "6 steps for data cleaning and why it matters":
   - **Source:** Gimenez, L. (2020) "6 steps for data cleaning and why it matters". Available at: https://www.geotab.com/blog/data-cleaning/ (Accessed at: 7 December 2021)

   - **Information Gathered:**
     - The definition of data cleaning.
     - How to carry out the data cleaning process.

   - **Why it was Useful:**
     It taught me the types of data cleaning, why data cleaning is essential, and the steps to clean data.

   - **Reliability & Limitations of the Source:**
     Whilst the credentials of the author, Leo Gimenez, is unknown, the company that published his article is GEOTAB, which is a telematics and Internet of Things provider, so the article is likely very reliable.

     However, a limitation of the source is that it only outlines four key (but short) steps for data cleaning, without in-depthly explaining them or giving examples.

   - **How I would Mitigate Those Limitations and Increase its Reliability in the Future:**
     I would find a new source that more in-depthly explains how to effectively carry out data cleaning.

7) Anonymous, "Machine Learning".
   - **Source:** Anonymous, (2020) "Machine Learning". Available at: https://www.ibm.com/cloud/learn/machine-learning (Accessed at: 15 December 2021)

- **Information Gathered:**
    - What machine learning is.
    - What the difference between an artificial neural network and a deep neural network is.
    - Various applications of machine learning.

- **Why it was Useful:**
  It gave me a formal definition of machine learning, and provided the difference between ANNs and DNNs. It also gave me various applications of ML, and a link to a University of California, Berkeley article that breaks down the ML process into three steps.

- **Reliability & Limitations of the Source:**
  The author of the article is not public, however, it was published by IBM - which is a big technology and machine learning company so the source is very reliable.

- **How I would Mitigate Those Limitations and Increase its Reliability in the Future:**

8) Johnson, J. (2020) “What’s a Deep Neural Network? Deep Nets Explained”:
    - **Source:** Johnson, J. (2020) “What’s a Deep Neural Network? Deep Nets Explained”. Available at: https://www.bmc.com/blogs/deep-neural-network/ (Accessed: 29 August 2021)

    - **Information Gathered:**
        - What DNNs are.
        - How they are different from normal ANNs.

    - **Why it was Useful:**
      Johnson gives a clear and simple definition of a DNN and how it is different from an ANN.

    - **Reliability & Limitations of the Source:**
      Johnson’s credentials are not known and the company that has published the article is a technology company, but is not a very commonly-known one. However, many definitions seem to match his, so this source is reliable.

    - **How I would Mitigate Those Limitations and Increase its Reliability in the Future:**
      Source from more reliable authors in the future.

9) Nicholson, C., (2020) “A Beginner's Guide to Neural Networks and Deep Learning”:
    - **Source:** Nicholson, C., (2020) “A Beginner's Guide to Neural Networks and Deep Learning”. Available at: https://wiki.pathmind.com/neural-network (Accessed: 29 August 2021)

    - **Information Gathered:**
        - What neural networks are.
        - What the various components that make up ANNs/DNNs are.
        - Figure 2.

    - **Why it was Useful:**
      Nicholson first simply defines neural networks as a whole and then in-depthly describes the many features of neural networks, and the complex theory behind it in a beginner-friendly way, whilst also keeping his terminology complex enough so that readers know what to research next.

      It was a very useful source for me when first learning what neural networks are.

    - **Reliability & Limitations of the Source:**
      Nicholson is the CEO of Pathmind, a machine learning company, so this source is reliable.

      One limitation of this source is that the website is not a peer-reviewed journal, so it cannot be checked for accuracy and validity.

    - **How I would Mitigate Those Limitations and Increase its Reliability in the Future:**
      One way to mitigate this limitation is to use a source from a peer-reviewed journal. However, a problem with this is that many of them would use complex terminology and would be difficult for beginners to understand.

10) Referred to as “NHS 1”: Anonymous, (n.d.) “Overactive Thyroid (hyperthyroidism)”:
    - **Source:** Referred to as “NHS 1”: Anonymous, (n.d.) “Overactive Thyroid (hyperthyroidism)”. Available at: https://www.nhs.uk/conditions/overactive-thyroid-hyperthyroidism/ (Accessed: 29 August 2021)

- **Information Gathered:**
    - What hyperthyroidism is.
    - The shape of the thyroid gland.

- **Why it was Useful:**
  Gave basic information about hyperthyroidism that was easy to read.

- **Reliability & Limitations of the Source:**
  The article was written by the National Health Service, which is a government funded health service for the UK, so it is very reliable.

- **How I would Mitigate Those Limitations and Increase its Reliability in the Future:**
  Not applicable.

11) Referred to as “NHS 2”: Anonymous, (n.d.) “Underactive Thyroid (hypothyroidism)”:
    - **Source:** Referred to as “NHS 1”: Anonymous, (n.d.) “Overactive Thyroid (hyperthyroidism)”. Available at: https://www.nhs.uk/conditions/overactive-thyroid-hyperthyroidism/ (Accessed: 29 August 2021)

    - **Information Gathered:**
        - What hypothyroidism is.

    - **Why it was Useful:**
      Gave basic information about hypothyroidism that was easy to read.

    - **Reliability & Limitations of the Source:**
      The article was written by the National Health Service, which is a government funded health service for the UK, so it is very reliable.

    - **How I would Mitigate Those Limitations and Increase its Reliability in the Future:**
      Not applicable.

12) Sharma, S., (2017) “Activation Functions in Neural Networks”:
    - **Source:** Sharma, S., (2017) “Activation Functions in Neural Networks”. Available at: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 (Accessed: 12 September 2021)

    - **Information Gathered:**

- What an activation function does.
- The sigmoid function.
- Why the sigmoid function is the correct activation function when the output of the DNN is a probability.

- **Why it was Useful:**
  It taught me about all the different categories of activation functions, and various different activation functions.

- **Reliability & Limitations of the Source:**
  Sharma, the author, is a software developer and has had experience in deep learning & AI. Furthermore, the website that published their article, Towards Data Science, is a very popular website used by hundreds of thousands of users.

  A limitation of the source is that the article is not published in a peer-reviewed journal and so there may be less checks for validity and accuracy.

- **How I would Mitigate Those Limitations and Increase its Reliability in the Future:**
  One way to mitigate this limitation is to use a source from a peer-reviewed journal. However, a problem with this is that many of them would use complex terminology and would be difficult for beginners to understand.

13) Swift, A., Heale, R., Twycross, A., "What are sensitivity and specificity?":
- **Source:** Swift, A., Heale, R., Twycross, A., "What are sensitivity and specificity?", BMJ, vol. 243, pp. 2 - 4, 2019

- **Information Gathered:**
  - What specificity is and how to calculate it.
  - What sensitivity is and how to calculate it.
  - The definitions of the terms True Positive, True Negatives, False Positives, False Negatives.
  - Why sensitivity and specificity is more important, within a medical context, than accuracy.

- **Why it was Useful:**
  It explains all of the above terms well with useful examples.

- **Reliability & Limitations of the Source:**

One of the authors, Dr Amelia Swift, has a PhD in Nursing and a Master's Degree in Health Sciences. They are also a reader in the School of Nursing in the University of Birmingham.

Furthermore, this source was published in the BMJ, which is a peer-reviewed journal. Therefore, this source is very reliable.

- **How I would Mitigate Those Limitations and Increase its Reliability in the Future:**
  Not applicable.

14) Tamir, Michael (2020) "What Is Machine Learning (ML)?":
- **Source:** Tamir, Michael (2020) "What Is Machine Learning (ML)?". Available at: https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/ (Accessed at: 10 January 2022)

- **Information Gathered:**
  - What machine learning is.
  - Which fields use machine learning.
  - The three core steps of machine learning.

- **Why it was Useful:**
  It breaks up the important, core information into easily readable bullet points, and outlines the three components of machine learning clearly.

- **Reliability & Limitations of the Source:**
  Tamir is the Head of Data Science at Uber ATG and a data science lecturer at the University of California, Berkeley (according to the Berkley website). Therefore, this source is very reliable.

- **How I would Mitigate Those Limitations and Increase its Reliability in the Future:**
  Not applicable.

15) Weber, L. M., (2020) "Explainer: What is an algorithm?":
- **Source:** Weber, L. M., (2020) "Explainer: What is an algorithm?". Available at: https://www.sciencenewsforstudents.org/article/explainer-what-is-an-algorithm (Accessed at: 28 November 2021)

- **Information Gathered:**
  - A formal definition of an algorithm.

- **Why it was Useful:**
    - It gave a formal definition of an algorithm.

- **Reliability & Limitations of the Source:**
  The source was published by a scientific journal, Science for Students, so it is likely reliable. A limitation of the source is that it does not outline the differences between a computer algorithm and a machine learning algorithm. However, that was likely outside of the scope of the article.

- **How I would Mitigate Those Limitations and Increase its Reliability in the Future:**
  I would attempt to find a source that defines both a computer and a machine learning algorithm, and then compares them to find the differences.

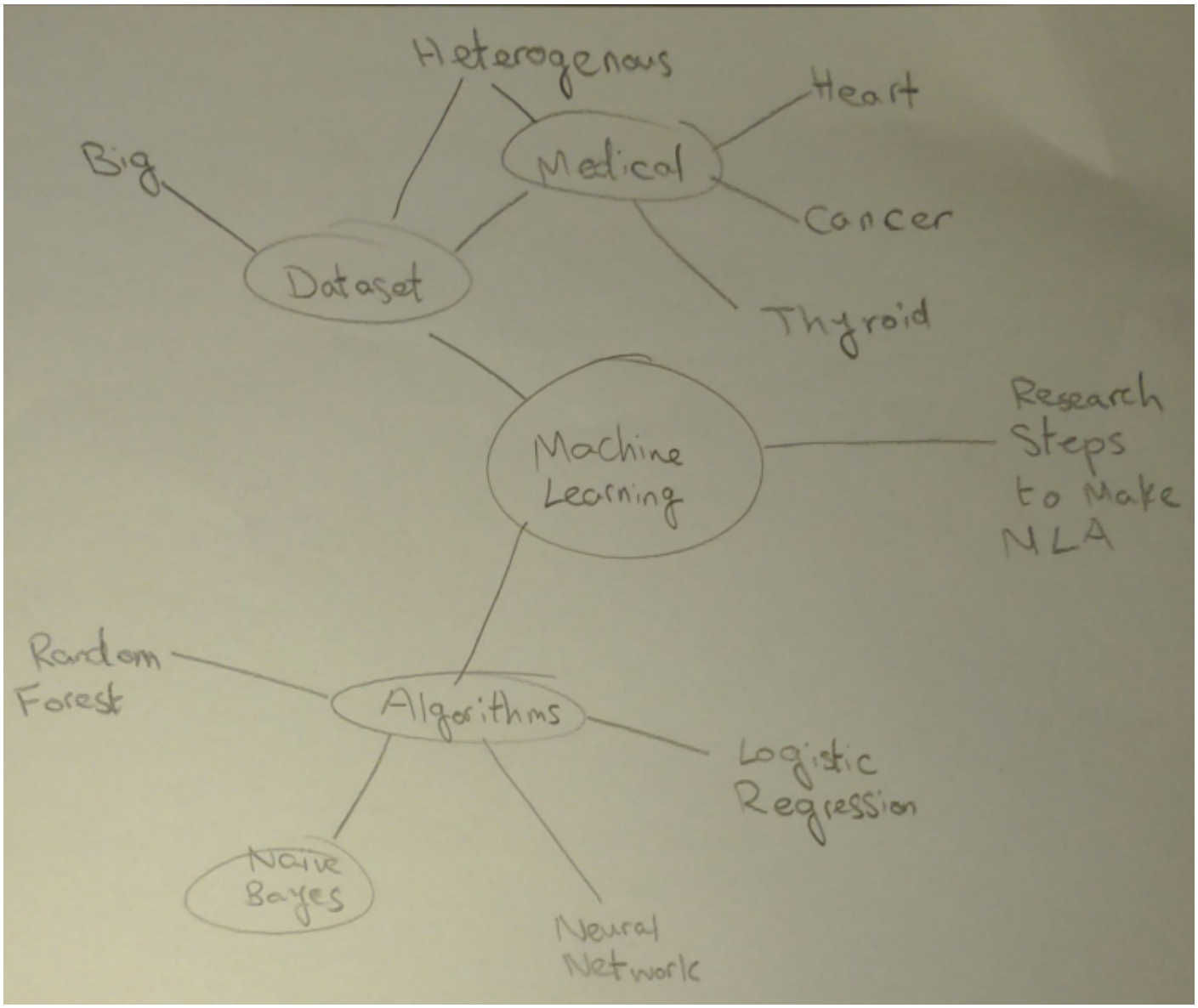
Appendix C: Production Log/Diary

10th February 2021 - Initial Ideas

- Initial question: Which machine learning algorithm is the most effective and accurate at diagnosing a certain medical condition/disease?
- Initial ideas for machine learning algorithms to program and test:
    - Bayesian Decision Making (Naïve Bayes);
    - Logistic Regression;
    - Support Vector Machines (SVMs);
    - Random Forest;
    - Alternating Least Squares;
    - Neural Networks;
    - Decision Tree Learning.
- https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861
    - Explains types of algorithms, such as:
        - ❖ Supervised learning;
        - ❖ Unsupervised learning;
        - ❖ Semi-supervised learning;
        - ❖ Reinforcement learning;
    - Explains various different algorithms (that each use different mathematical concepts).

11th February 2021 - Choosing The Condition/Disease

- Looked at the UCI Machine Learning Repository.
- Chose thyroid disease because:
    - Wanted large, heterogeneous dataset.
    - A very prevalent application of ML is in diagnosing diseases.
    - There is lots of medical literature on thyroid disease which will help me understand the dataset.



20th February - Steps of Machine Learning

- 3 core, simple steps of machine learning:
  1) **The decision process:** This is the stage at which the algorithm will carry out certain computations and then make a prediction for the current instance.

  2) **An error function:** A function that answers the following questions:
     - Was the prediction correct compared to the actual label/diagnosis?

     - If the prediction was wrong, how will this error margin be quantified? The function will then carry out the quantification of the error.

3) **The update/optimisation process:** The algorithm will update certain constants within itself so that it yields more accurate results for the next instance of the training data set.

- Steps in detail:

1) Gathering the data:
   - UCI Machine Learning repository.
   - Thyroid Disease dataset (1987) - 5,285 instances.

2) Data Preparation:
   - Ensure the data is in a table/csv format.
   - Randomise the data.
   - Split the dataset into training and testing sets (UCI dataset comes pre-split in a 3893:1392 ratio.

3) Data Cleaning:
   - Data cleaning is the process of ensuring data is correct, consistent, usable, and not missing.
   - Data can be cleaned by identifying errors, corruptions, or missing data and correcting or deleting them.
   - Data cleaning is essential because if it's not done, then the algorithm is not reliable.

4) Training:
   - The algorithm will use the dataset to increase its prediction/diagnosing accuracy.
   - Steps for training:
     a) The training program "looks" at the first instance of the training set.

     b) The algorithm will predict a label using the values of the attributes from this instance as inputs.

     c) The training program will compare the predicted label with the actual label.

     d) If the prediction is correct, it will do nothing.

     e) If the prediction is incorrect, the training program will adjust the training parameters inside the algorithm.

     f) Then, it will move onto the next instance

g) Steps 2 to 8 are repeated until the whole training set has been used.

5) Testing:
    - The algorithm is run on the testing set.
    - The difference is, that this time, if the prediction is incorrect, the algorithm will not adjust itself to increase its accuracy.
    - Instead, it tallies up all the incorrect and correct predictions.

6) Evaluating Performance:
    - Using the previously mentioned tallies, an accuracy percentage can be created. This is the simplest form of evaluation.
    - A confusion matrix can also be made for more in-depth analysis of the results.

11th March 2021 - Part B Supervisor Proposal

- Met with my supervisor to verify project details.

28th April 2021 - Planning Review

- Finalised objectives, tasks, and deadlines.

- Decided to settle on testing a Naive Bayes Algorithm and a Deep Neural Network because the former is considered a more beginner friendly algorithm, and the latter is among the most popular algorithms.

- Supervisor suggested that I make a graphical form of the plan, such as a Ggantt chart.
    - I decided on both a tTrello board and a Gantt chart.

27th June 2021

- Researched classification.
- Researched Naive Bayes Algorithm:
    - Uses conditional probability and Bbayes' theorem.

    - Need to use the Gaussian equation as the thyroid disease dataset has continuous data, and bayes theorem only works with boolean values.
    - Researched how each algorithm worked.
    - Programmed the wikipedia worked example to practise programming a gaussian naive bayes algorithm.

- Researched Deep Neural Network:
    - Modelled after a brain.
    - Uses a input, hidden, and output nodes.
    - Learnt the intricacies of DNNs, such as the difference between normal artificial neural networks and DNNs, and what activation functions are.

- Learnt how to read CSVs in Python.
- Downloaded dataset.

2nd July 2021

- Changed one of the deadlines. The deadline of “Learn how to do Tensorflow” was changed from 3rd July to 10th July because I need to prepare for my Duke of Edinburgh expedition (5th July to 9th July).

12th July 2021

- Part way through programming the Gaussian Naive Bayes algorithm.
- Lost all progress from 12/07/2021 because of a GitHub error.
- Realised that I can’t use the Gaussian equation for naive bayes because many attributes are boolean values and the gaussian equation uses strictly continuous data.
    - Should have’ve done more research before I started.
    - Can’t use the multinomial equation because that’s for frequency counts.
    - Bernouli uses boolean data only. If I use this, I need to discretize all of my continuous data. This will be done using thresholds. If the variable is above the threshold, it will be given the value of “true”, else “false”.
    - Conclusion: Switch from the Gaussian Naive Bayes to the Bernouli Naive Bayes and discretize continuous data.

14th July 2021

- Began data cleaning process.

- Reduced number of labels in each dataset. Original labels:
    - Hypothyroid
    - Primary Hypothyroid (high TSH)
    - Compensated Hypothyroid (asymptomatic)
    - Secondary Hypothyroid (low TSH)
    - Negative
    - Hyperthyroid
    - T3 Toxic Hyperthyroid
    - Goitre Hyperthyroid
    - Secondary Toxic Hyperthyroid.

- New labels:
    - Hypothyroid
    - Hyperthyroid
    - Negative

15th July 2021

- Data cleaning continues.
- Could discretize data by using the NICE (National Institute for Chealth and Care Excellence (NICE)) thresholds.
- NICE doesn't have thresholds.
- TSH levels from https://www.uclahealth.org/endocrine-centre/normal-thyroid-hormone-levels
    - TSH normal: $0.5 \leq x \leq 5\, mIu/L$
    - FT4 Normal: $0.7 \leq x \leq 1.9\, ng/dL$
    - TT4 Normal: --
    - T4 Normal: $0.8 \leq x \leq 2.2\, \mu g/dL$
    - Couldn't find T4U normal levels (going to remove that attribute).
- Experts disagree on "normal" ranges. Some say that a TSH of 2.5 is too high, but others say it is normal.

17th July 2021

- Because so many experts disagree on the normal ranges, decided to remove many boolean attributes (other than sex).
- This is because many of the boolean attributes just record whether a continuous measurement has been taken or not.
    - This is useless because if the value is false, we can just assume the measurement to either be `0` or a `None` value.
    - If the value is true, then there is already a value for the measurement.
    - So the boolean variable adds no new information.

- The attributes that remain are:
    - Age (continuous real)
    - Sex (boolean)
    - TSH (continuous real integer)
    - FT4 (continuous real integer)
    - TT4/T4 (continuous real integer)
    - T4U (continuous real integer)

- Because the only attributes left are continuous, the gaussian naive bayes algorithm can be used (other than sex).
- Also removed any instance/patient that has at least one missing cell/value (part of data cleaning).

24th July 2021

- Finished naive bayes algorithm.
- Evaluated the algorithm with the testing dataset.

3rd August 2021

- Finished deep neural network.
- Evaluated it on the testing data set.

7th August 2021

- Started the report.
- Wrote abstract, introduction, and methodology (for data cleaning).

18th August 2021

- Started to write about results and findings.
- Skipped methodology about programming the algorithms for now.

19th August 2021

- Finished results section.
- Finished conclusion.
- Finished EPQ evaluation.

29th August 2021

- Finished writing about the GNB algorithm.

- Finished writing about the DNN algorithm.

2nd September 2021

- Made various grammatical changes.
- Added a table for the dataset attributes and labels.

5th - 9th September 2021

- Added paragraph to methodology to describe the process of training.
- Explained the DNN more thoroughly (explained weights and activation functions).

11th - 13th September 2021

- Started reference list/bibliography.
- Various grammatical changes.
- Added paragraphs explaining thyroid disease.
- Added “figure” titles to images and diagrams.

October 2021

- Put a pause on EPQ to focus on subjects and UCAS Application.

6th November 2021

- Explained the concept of “maximum likelihood”.
- Explained “evidence” and how a posterior probability for each label is found for GNB.

10th November 2021

- Explained the concepts of “true/false positives” and “true/false negatives”.
- Added more references to the reference list.

20th November 2021

- Added experiment evaluation (bias) section.
- Added equation numbers.
- Moved some paragraphs around, added some new sections.

4th - 10th December 2021

- Added explain dataset to help better explain how datasets work.
- Wrote about types of missing data for the data cleaning section, and “listwise deletion” (a method to clean the dataset).
- Wrote more in-depthly about the thyroid gland.
- Grammatical changes.
- Gave formal definitions of “instances” and “attributes”.
- Added definition of “variable/data type”.

11th - 13th December 2021

- Changed wording of abstract section.
- Grammatical changes.
- Changed variable names when explaining how the Gaussian Equation works.
- Explained the difference between an ML algorithm and model.
- Explained the difference between a predicted label and the actual label.
- Changed experiment evaluation: edited paragraphs about bias.

14th December 2021

- Removed “project evaluation” section as the supervisor said it is to be done
- in the production log.
- Changed report from two column format to one column as it allows diagrams, images, and equations to take up more space and so are more visible.
- Removed types of missing data section as it adds no overall information to the report.
- Added source code for each algorithm to the appendices.

9th January 2022

- Edited in-text references to follow harvard referencing.
- Added source evaluation appendix.
- Changed first definition of ML in the introduction.
- Changed definition of “deep neural network”.
- More in-depthly described types of activation functions.
- Removed unused references.
- Added three core components of ML.

Appendix D: Python Code for GNB

Main.py:

```
########### Random Info ###########
# Naive Bayes Classifier EPQ
# Saatvik Kambhampati
# 14/07/2021 -
###################################

########### Importing Modules ###########
import csv;
import json;
import numpy as np;
import pandas as pd;
from ClassOutcomes import classOutcome;
#########################################


class main():
  def __init__(self):
    return;

  def getAtts(self, data):
    atts = [
            [], #age
            [], #sex
            [], #tsh
            [], #t3
            [], #tt4
            [], #t4u
            [] #fti
           ];

    ClassOutcomes = [] #hypo, hyper, negative/no.

    for i in range(len(data)):
      atts[0].append(float(data[i][0]));
```

```
    atts[1].append(str(data[i][1]));
    atts[2].append(float(data[i][2]));
    atts[3].append(float(data[i][3]));
    atts[4].append(float(data[i][4]));
    atts[5].append(float(data[i][5]));
    atts[6].append(float(data[i][6]));
    ClassOutcomes.append(str(data[i][7]));
  return [atts, ClassOutcomes];


def readCSV(self, FileName):
  with open('../' + FileName + '.csv', 'r') as file:
    data = [];
    for row in csv.reader(file):
      if("Age" not in row):
        data.append(row);

    return data;



def generalProbabilities(self, outcomes, load):
  hypo.generalProbabilityClass(outcomes, "hypothyroid", load);
  hyper.generalProbabilityClass(outcomes, "hyperthyroid", load);
  no.generalProbabilityClass(outcomes, "no", load);


def saveGeneralProbs(self):
  try:
    dict = {
      "hypothyroid": hypo.GeneralProbability,
      "hyperthyroid": hyper.GeneralProbability,
      "no": no.GeneralProbability
    };

    with open("../../NaiveBayes/GenProbs.json", "w") as file:
      json.dump(dict, file);

    return True;

  except Exception as e: return False;
```

```
  def evidence(self, i):
      return (
        (
          hypo.GeneralProbability*

hypo.CondProbAtts["age"][i]*hypo.CondProbAtts["tsh"][i]*hypo.CondP
robAtts["t3"][i]*

hypo.CondProbAtts["tt4"][i]*hypo.CondProbAtts["t4u"][i]*hypo.CondP
robAtts["fti"][i]*

hypo.CondProbAtts["sexes"]["males"]*hypo.CondProbAtts["sexes"]["fe
males"]
        ) + (
          hyper.GeneralProbability*

hyper.CondProbAtts["age"][i]*hyper.CondProbAtts["tsh"][i]*hyper.Co
ndProbAtts["t3"][i]*

hyper.CondProbAtts["tt4"][i]*hyper.CondProbAtts["t4u"][i]*hyper.Co
ndProbAtts["fti"][i]*

hyper.CondProbAtts["sexes"]["males"]*hyper.CondProbAtts["sexes"]["
females"]
        ) + (
          no.GeneralProbability*

no.CondProbAtts["age"][i]*no.CondProbAtts["tsh"][i]*no.CondProbAtt
s["t3"][i]*

no.CondProbAtts["tt4"][i]*no.CondProbAtts["t4u"][i]*no.CondProbAtt
s["fti"][i]*

no.CondProbAtts["sexes"]["males"]*no.CondProbAtts["sexes"]["female
s"]
        )
      );

  def results(self, actual, predicted):
      # CONFUSION MATRIX
      PDactual = pd.Series(actual, name = "Actual");
      PDpredicted = pd.Series(predicted, name = "Predicted")
```

```
      print(pd.crosstab(PDactual, PDpredicted, rownames =
["Actual"], colnames = ["Predicted"], margins = True));

      # % Correct
      TotalCorrect, CorrectHypo, CorrectHyper, CorrectNo = 0, 0,
0, 0;
      PredictedHypo, PredictedNo, PredictedHyper = 0, 0, 0;

      for i in range(len(actual)):
        if(actual[i] == predicted[i]):
          TotalCorrect += 1
          if(predicted[i] == "hypothyroid"): CorrectHypo += 1;
          elif(predicted[i] == "hyperthyroid"): CorrectHyper += 1;
          else: CorrectNo += 1;

        if(predicted[i] == "hypothyroid"): PredictedHypo += 1;
        elif(predicted[i] == "hyperthyroid"): PredictedHyper += 1;
        else: PredictedNo += 1;


      PercentCorrect = round((TotalCorrect/len(actual))*100, 2);
      print("Total correctly predicted: " + str(PercentCorrect) +
"%");
      print("Total incorrectly predicted: " + str(round(100 -
PercentCorrect, 2)) + "%");
      print("Out of all patients that were predicted
'hypothyroid', correct % predictions were: " +
str(round((CorrectHypo/PredictedHypo)*100, 2)) + "%");
      print("Out of all patients that were predicted
'hyperthyroid', correct % predictions were: " +
str(round((CorrectHyper/PredictedHyper)*100, 2)) + "%");
      print("Out of all patients that were predicted 'no', correct
% predictions were: " + str(round((CorrectNo/PredictedNo)*100, 2))
+ "%");

  def main(self):
    which = int(input("Testing [1] or training [2]? "));
    if(which == 1):
      data = self.readCSV("test");
      res = self.getAtts(data);
      atts, classes = res[0], res[1];
      ########################
```

```
      TempAtts = atts;
      del TempAtts[1];

      self.generalProbabilities(classes, True)
      hypo.findingAttsGivenClasses(TempAtts, classes,
"hypothyroid", data);
      hypo.pSexGivenClass(self.getAtts(data)[0], classes,
"hypothyroid");
      hyper.findingAttsGivenClasses(TempAtts, classes,
"hyperthyroid", data);
      hyper.pSexGivenClass(self.getAtts(data)[0], classes,
"hyperthyroid");
      no.findingAttsGivenClasses(TempAtts, classes, "no", data);
      no.pSexGivenClass(self.getAtts(data)[0], classes, "no");


      predicted, actual = [], [];
      for i in range(len(hypo.CondProbAtts["age"])):
        evidence = self.evidence(i);
        ProbHypo, ProbHyper, ProbNo = hypo.SpecificProb(evidence,
i), hyper.SpecificProb(evidence, i), no.SpecificProb(evidence, i);
        print("COUNT:", i)
        if((ProbHypo > ProbHyper) and (ProbHypo > ProbNo)):
          print("================");
          print("Prediction: hypothyroid");
          print("Actual:", classes[i]);
          print("================");
          predicted.append("hypothyroid");
          actual.append(classes[i]);

        elif((ProbHyper > ProbHypo) and (ProbHyper > ProbNo)):w
          print("================");
          print("Prediction: hyperthyroid");
          print("Actual:", classes[i]);
          print("================");
          predicted.append("hyperthyroid");
          actual.append(classes[i]);

        else:
          print("================");
          print("Prediction: no");
          print("Actual:", classes[i]);
```

```
            print("================");
            predicted.append("no");
            actual.append(classes[i]);

        self.results(actual, predicted);

    else:
        data = self.readCSV("training");
        res = self.getAtts(data);
        atts, classes = res[0], res[1];
        self.generalProbabilities(classes, False);
        self.saveGeneralProbs();

hypo, hyper, no = classOutcome(), classOutcome(), classOutcome();
obj = main();
obj.main();
```

ClassOutcomes.py:

```
############ Importing Modules ############
import numpy as np;
import json;
#####################################

class classOutcome():
  def __init__(self):
    self.means = [0, 0, 0, 0, 0, 0]; #age, tsh, t3, tt4, t4u, fti
    self.variances = [0, 0, 0, 0, 0, 0]; #age, tsh, t3, tt4, t4u,
fti
    self.pMaleClass, self.pFemaleClass = 0, 0;
    self.TotalMales, self.TotalFemales = 0, 0;
    self.GeneralProbability = 0;
    self.CondProbAtts = {
      "sexes": {},
      "age": [],
      "tsh": [],
      "t3": [],
      "tt4": [],
      "t4u": [],
      "fti": []
    };
```

```
  def findMean(self, index, atts, WhichClass, outcomes):
    mean = 0;
    NumOfAges = 0;
    for i in range(len(atts[index])):
      try:
        if(outcomes[i] == WhichClass):
          mean += float(atts[index][i]);
          NumOfAges += 1;

      except Exception as e: print(e);

    mean /= NumOfAges;
    return mean;

  def findVariance(self, index, atts, mean, outcomes, WhichClass):
    MeanOfSquares, length = 0, 0;
    squares = []
    for i in range(len(atts[index])):
      try:
        if(outcomes[i] == WhichClass):
          MeanOfSquares += (np.square(atts[index][i]));
          squares.append(np.square(atts[index][i]))
          length += 1;

      except Exception as e: print(e);

    MeanOfSquares /= length;
    return np.absolute(MeanOfSquares - (mean*mean));

  def pSexGivenClass(self, atts, outcomes, WhichClass):
    sexes = atts[1];

    for i in range(len(sexes)):
      if((outcomes[i] == WhichClass) and sexes[i] == "M"):
        self.TotalMales += 1
      elif((outcomes[i] == WhichClass) and sexes[i] == "F"):
        self.TotalFemales += 1;

    self.CondProbAtts["sexes"]["males"] =
self.TotalMales/len(sexes);
    self.CondProbAtts["sexes"]["females"] =
```

```
self.TotalFemales/len(sexes);

  def attGivenClass(self, mean, variance, val):
    thing =
(1/(np.sqrt(2*np.pi*variance)))*np.exp(((-(np.square(float(val) -
mean)))/(2*variance)));
    return thing;

  def generalProbabilityClass(self, outcomes, WhichClass, load):
    if(load == False): #used for training
      ThisClass = 0;
      for i in range(len(outcomes)):
        if(outcomes[i] == WhichClass): ThisClass += 1;

      self.GeneralProbability = ThisClass/len(outcomes);
    else: #used for testing
      with open("../../GenProbs.json") as file:
        data = json.load(file);
        self.GeneralProbability = data[WhichClass];

  def findingAttsGivenClasses(self, atts, outcomes, WhichClass,
data):
    for i in range(len(atts)):
      self.means[i] = self.findMean(i, atts, WhichClass,
outcomes);
      self.variances[i] = self.findVariance(i, atts,
self.means[i], outcomes, WhichClass);

    # exit()

    for i in range(len(data)):
      for j in range(len(data[i])):
        if("Age" in data[i]): break;
        if(j == 0):

self.CondProbAtts["age"].append(self.attGivenClass(self.means[0],
self.variances[0], data[i][j]));
        elif(j == 2):

self.CondProbAtts["tsh"].append(self.attGivenClass(self.means[1],
self.variances[1], data[i][j]));
        elif(j == 3):
```

```
self.CondProbAtts["t3"].append(self.attGivenClass(self.means[2],
self.variances[2], data[i][j]));
        elif(j == 4):

self.CondProbAtts["tt4"].append(self.attGivenClass(self.means[3],
self.variances[3], data[i][j]));
        elif(j == 5):

self.CondProbAtts["t4u"].append(self.attGivenClass(self.means[4],
self.variances[4], data[i][j]));
        elif(j == 6):

self.CondProbAtts["fti"].append(self.attGivenClass(self.means[5],
self.variances[5], data[i][j]));

  def SpecificProb(self, evidence, i):
    return ((float(self.GeneralProbability)*

float(self.CondProbAtts["age"][i])*float(self.CondProbAtts["tsh"][
i])*float(self.CondProbAtts["t3"][i])*

float(self.CondProbAtts["tt4"][i])*float(self.CondProbAtts["t4u"][
i])*float(self.CondProbAtts["fti"][i])*

float(self.CondProbAtts["sexes"]["males"])*float(self.CondProbAtts
["sexes"]["females"])/float(evidence)));
```

Appendix E: Python Code for DNN

This program uses Tensorflow for training and testing the DNN.

Main.py:

```
########### Random Info ###########
# Neural Networks Classifier EPQ
# Saatvik Kambhampati
# 14/07/2021 -
###################################

########### Importing Modules ###########
import csv;
import json;
import numpy as np;
import pandas as pd;
import tensorflow as tf;
import matplotlib as plt;
import tensorflow.compat.v2.feature_column as fc;
#########################################

# Attributes
attributes = ["Age", "Sex", "TSH", "T3", "TT4", "T4U", "FTI"];
#Sex: male: 1, female: 0
classifications = ["hypothyroid", "hyperthyroid", "no"];

# Read CSV
train = pd.read_csv("/content/training.csv");
test = pd.read_csv("/content/test.csv");
TrainLabels = train.pop('Class'); #No: 0, Hypo: 1, Hyper: 2
TestLabels = test.pop('Class');
print(train)

# Input Function
def inputFunc(atts, labels, training = True, batch = 256):
  dataset = tf.data.Dataset.from_tensor_slices((dict(atts),
labels));

  if (training):
    dataset = dataset.shuffle(1000).repeat();
```

```
  return dataset.batch(batch);

# Feature columns to describe how to use the input.
FeatureColumns = [];
for i in train.keys():
  FeatureColumns.append(tf.feature_column.numeric_column(key =
i));

print(FeatureColumns);


# Creating neural network model, 2 hidden layers, with 39 and 10
hidden node each.
model = tf.estimator.DNNClassifier(
  feature_columns = FeatureColumns,
  hidden_units = [30, 10],
  n_classes = 4
);

# Training the model
model.train(
  input_fn = lambda: inputFunc(train, TrainLabels, training =
True),
  steps = 5000
);

# Testing the model
res = model.evaluate(
    input_fn = lambda: inputFunc(test, TestLabels, training =
False)
);

print('\nTest set accuracy: {accuracy:0.3f}\n'.format(**res))

#List of prediction labels

predictions = list(model.predict(input_fn = lambda:
inputFunc(test, TestLabels, training = False)));


PredictionLabels = [];
for i in range(len(predictions)):
```

```
PredictionLabels.append(np.argmax(predictions[i]["probabilities"])
);

PDactual = pd.Series(TestLabels, name = "Actual");
PDpredicted = pd.Series(PredictionLabels, name = "Predicted");
print(pd.crosstab(PDactual, PDpredicted, rownames = ["Actual"],
colnames = ["Predicted"], margins = True));
```