

# CS6P05 - Project

**[Applied Machine Learning to Detection of Steganography.]**  
**Project Final Report - Submission (Software Development Type**  
**Project)**

---

Name: Jamie Hoang

ID Number: 19009101

Date: 09/01/2022

First Supervisor: Mohamed Chahine Ghanem

Second Supervisor: Alexandros Chrysikos

# Declaration

**Module:** CS6P05

**Deadline:** 3pm 9<sup>th</sup> May 2019

**Module Leader:** Dr. Preeti Patel

**Student ID:** 19009101

## PLAGIARISM

You are reminded that there exist regulations concerning plagiarism. Extracts from these regulations are printed below. Please sign below to say that you have read and understand these extracts:

Student signature: Jamie Hoang

Date: 09/05/22

Extracts from University *Regulations* on Cheating, Plagiarism and Collusion

Section 2.3: "The following broad types of offence can be identified and are provided as indicative examples...

- (i) Cheating: including taking unauthorised material into an examination; consulting unauthorised material outside the examination hall during the examination; obtaining an unseen examination paper in advance of the examination; copying from another examinee; using an unauthorised calculator during the examination or storing unauthorised material in the memory of a programmable calculator which is taken into the examination; copying coursework.
- (ii) Falsifying data in experimental results.
- (iii) Personation, where a substitute takes an examination or test on behalf of the candidate. Both candidate and substitute may be guilty of an offence under these Regulations.
- (iv) Bribery or attempted bribery of a person thought to have some influence on the candidate's assessment.
- (v) Collusion to present joint work as the work solely of one individual.
- (vi) Plagiarism, where the work or ideas of another are presented as the candidate's own.
- (vii) Other conduct calculated to secure an advantage on assessment.
- (viii) Assisting in any of the above.

Some notes on what this means for students:

1. Copying another student's work is an offence, whether from a copy on paper or from a computer file, and in whatever form the intellectual property being copied takes, including text, mathematical notation and computer programs.
2. Taking extracts from published sources *without attribution* is an offence. To quote ideas, sometimes using extracts, is generally to be encouraged. Quoting ideas is achieved by stating an author's argument and attributing it, perhaps by quoting, immediately in the text, his or her name and year of publication, e.g. "e = mc<sup>2</sup> (Einstein 1905)". A *references* section at the end of your work should then list all such references in alphabetical order of authors' surnames. (There are variations on this referencing system which your tutors may prefer you to use.) If you wish to quote a paragraph or so from published work then indent the quotation on both left and right margins, using an italic font where practicable, and introduce the quotation with an attribution.

This header sheet should be attached to the work you submit. No work will be accepted without it.

# Summary/Abstract

---

Within this document is a complete compilation of the efforts of this student's final project during the Autumn and Winter seasons of 2021. As well as the Spring and Early Summer seasons of 2022. The document is structured into seven total Chapters. These chapters detail the intent and purpose of this project, technical and comprehensive research around various topics related to this project, what are the requirements of the final product, the design of the final product and a complete summary of the current progress of the project and conclusion of the project as of the publishing of this document.

It should be stated that this project is a Software Development Type Project. In which the overall goal is to create an end product that attempts to resolve an outstanding issue and innovate within the field of forensics.

As the title of this document states "Applied Machine Learning to Detection of Steganography". The end goal of this project is to create a proof of concept to showcase that the application of Machine Learning can be applied to labour-intensive processes such as the Detection of Steganography. Machine Learning has been known to allow for the automation of tasks usually done by humans. As it is a sub-branch of Artificial Intelligence.

This document aims to also showcase the student's skills in various fields. Primarily skills such as Programming, Design, Project Management and other academic skills are needed to convey this student's efforts over these seasons of 2021 and 2022. In addition, this document also aims to convey the competence and quality of the potential product presented by the student.

# Table of Contents

---

<i>Title Page</i> .....	<i>1</i>
<i>Index</i> .....	<i>4</i>
<hr/>	
<i>Chapter 1: Introduction</i> .....	<i>7</i>
<i>Chapter 2: Background Research</i> .....	<i>9</i>
<i>Chapter 3: Requirements Analysis and Specification</i> .....	<i>15</i>
<i>Chapter 4: Software Design</i> .....	<i>15</i>
<i>Chapter 5: Summary and Future work</i> .....	<i>20</i>
<i>Appendices</i> .....	<i>21</i>
<i>References/Bibliography</i> .....	<i>23</i>

# Table of Tables

---

<i>Table 2.1</i> .....	<i>9</i>
<i>Table 2.2</i> .....	<i>12</i>
<i>Table 5.2</i> .....	<i>32</i>

# Table of Figures

---

<i>Figure 1.a</i> .....	8
<i>Figure 2.a</i> .....	10
<i>Figure 2.b</i> .....	10
<i>Figure 2.b.1</i> .....	11
<i>Figure 2.C</i> .....	13
<i>Figure 4.aa</i> .....	16
<i>Figure 4.a</i> .....	16
<i>Figure 4.b</i> .....	17
<i>Figure 4.c</i> .....	17
<i>Figure 4.d</i> .....	17
<i>Figure 4.e</i> .....	17
<i>Figure 4.f</i> .....	18
<i>Figure 4.g</i> .....	18
<i>Figure 4.h</i> .....	19
<i>Figure 4.i</i> .....	19
<i>Figure 4.j</i> .....	19
<i>Figure 22A</i> .....	22
<i>Figure 22B</i> .....	22
<i>Figure 22C</i> .....	23

<i>Figure 22D</i> .....	24
<i>Figure 22E</i> .....	24
<i>Figure 22F</i> .....	24
<i>Figure 22G</i> .....	25
<i>Figure 22H</i> .....	26
<i>Figure 22I</i> .....	26
<i>Figure 22J</i> .....	27
<i>Figure 22K</i> .....	27
<i>Figure 22M</i> .....	27
<i>Figure 22N</i> .....	28
<i>Figure 22O</i> .....	28
<i>Figure 4.1</i> .....	29
<i>Figure 4.2</i> .....	30
<i>Figure 5.1</i> .....	31
<i>Figure 6.1</i> .....	35
<i>Figure 66.a</i> .....	37
<i>Figure 66.B</i> .....	38
<i>Figure 66.C</i> .....	39
<i>Figure 66.D</i> .....	39
<i>Figure XX.1</i> .....	40

<i>Figure XX.2</i> .....	40
<i>Figure XX.3</i> .....	40
<i>Figure XX.4</i> .....	40
<i>Figure XX.5</i> .....	41
<i>Figure XX.6</i> .....	41
<i>Figure XX.7</i> .....	42
<i>Figure XX.8</i> .....	43
<i>Figure XX.9</i> .....	44
<i>Figure XX.10</i> .....	44
<i>Figure XX.11</i> .....	45



# Chapter 1: Introduction

---

## 1.1 - Project topic and rationale

The project aims to bring machine learning to the field of Computer Forensics. Forensic Investigators face strenuous deadlines under the pressure to meet targets. They are faced with processing entire hard drives of suspected individuals which can range into terabytes of data. Limited by physical constraints, forensic investigators face challenges with workloads that scale in size rapidly.

Artificial Intelligence aims to automate tasks undertaken by humans utilizing the processing power of computers. Machine Learning is a sub-branch of this field in which it processes data to generate instructions to achieve a set goal by the developers. Given a single year, the student aims to present an application utilizing machine learning to achieve the desired goals.

This program aims to alleviate the strain on forensic investigators. Be it, they face a tight deadline, therefore, reducing the workload via the proposed program. Or that they reached a dead-end within an investigation therefore the program can offer additional leads for investigators to look into.

With the overall aim to bring the field of Machine Learning to Digital Forensics. There are many facets within the field of Digital Forensics. With the goal of the project to showcase the potential that Machine Learning could have for Forensic Investigators, therefore, the scope of the project will have to focus on one facet of the field.

This program in particular aims to target the field of steganography. The practice of concealing hidden messages within physical objects is to be put shortly. Criminals or other suspected entities

The academic challenge comes in the form that the student possesses little knowledge of programming and principles related to Machine Learning such as mathematics and statistics. In addition, the program, development of said program and targets set for this program are as close to the standard that industrial projects undertaken by corporations within reason.

This project aims to showcase the student's fundamental understanding of modules learnt during his academic career. As well as, a showcase of the student's academic skills cultivated during his lifetime. Which is to adapt to rising challenges, learn completely new fields of disciplines, management of a large-scale project and be able to source the resources necessary to complete the project.

## 1.2 - Project Aims and Objectives

The primary goal of this project is to create a piece of software that utilizes machine learning to detect steganography use within images. Which will be achieved with the objectives listed below.

- To assemble and gather the necessary knowledge to build said application.
  - Re-learning the basics of Python and principles of using a programming language.
  - Learning the fundamentals of machine learning and its mathematical algorithms. To grasp how the application should operate on a principle level.
  - Compiling the necessary Python Libraries and their relevant documentation.
- To document findings and progress in a timely and concise manner.
  - Attending periodical meetings and documentation of all aspects of the project. Such as explaining the theoretical side, explanations of the software and personal reflections from the student.
- To create training data for the machine learning algorithm to learn from. Creating a sample size that is diverse and challenging to the algorithm.
  - This would mean compiling various software and methodologies to apply Steganography to files.
  - The sample data should be in an Excel Spreadsheet for outsiders to review the sample data. With additional metadata on how each individual sample was created.

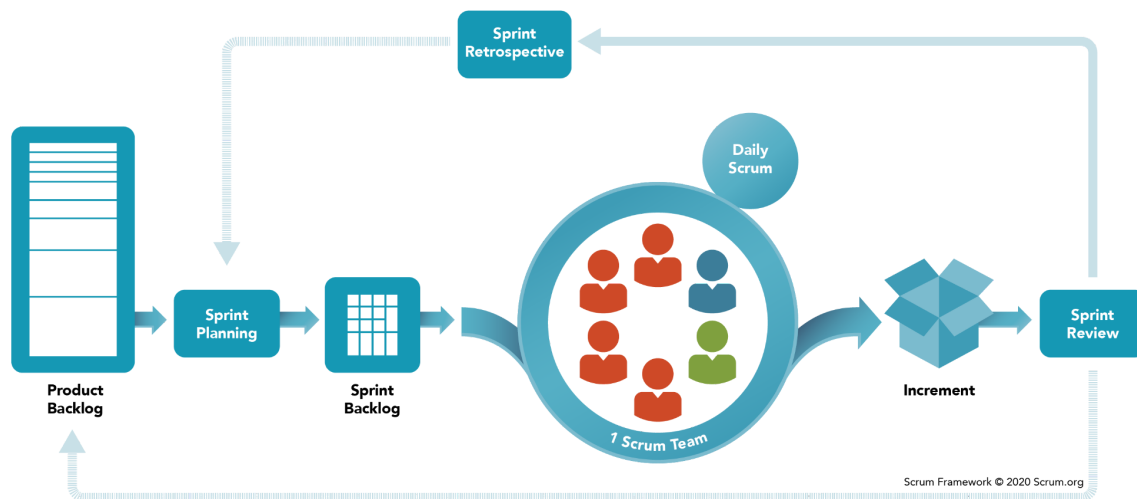
- For the machine learning algorithm to achieve a 95% accuracy rate when assessing files.
  - In the real world, the minimum accuracy rate should be 99%.however due to resource constraints. A 5% error rate is acceptable.
  - Tests to quantify an accuracy rate can be done with internal mechanisms and external tests done with samples separate from the training data.

### 1.3 - Methodology

The aims and objectives of the project are broken down into sequential tasks. As shown in the list below.

- Literature Search
- Literature Review
- Design Software
- Develop and Iterate Versions
- Collect and process samples
- Document Samples
- Train software
- Software Evaluation
- Final Report

We will be using the following software development methodology “SCRUM”. Which allows us to adapt depending on if we are on schedule, behind schedule or exceeding expectations. It is considered to be an iterative methodology that builds upon what happened prior. It breaks down the needs of the product into a backlog. Then it is further broken down into sprints. Which are short bursts of time in which a planned action is set as a target. Such as the implementation of a core function within a piece of software. These sprints are then repeated until there isn't a product backlog anymore. Thus resulting in the final product. Whilst initially meant for small teams. The student feels that it can be applied to a single individual. This can be seen in Figure 1.a.



*Figure 1.a - SCRUM Diagram (Scrum.org, 2021)<sup>1</sup>*

Prioritizing core functions that are key to the success of the project will take precedence over desirable but non-foundational functions. I.e the core systems required for data processing will be prioritized over a graphical interface.

### 1.4 - The report structure

This document is structured into chapters. Each chapter presents a different element of the project.

- Chapter 2: Background Research
  - Chapter 2 describes the principles and theories surrounding the project. As well as providing the foundational and core knowledge needed to advance the project. Specifically, it will provide a technical description of Machine Learning, Steganography and Digital Forensics.
- Chapter 3: Requirements Analysis and Specification
  - Chapter 3 describes the projected outcomes of the project. In the case of this project, it is specifically describing the results of the designed program. Since this project is a software development type project. As well as, describing what problem the program is intended to solve.
- Chapter 4: Software Design
  - Chapter 4 will describe the internal design of the software, a proposal of a graphical interface and any associated components to the software. Such as the format of input data, how data is processed into a suitable format and where it is stored.
- Chapter 5: Summary and Future work
  - Chapter 5 is a complete summary of the current progress of the project as written in January of 2021. As well as describing what remaining targets and goals are left for the project.

## Chapter 2: Background Research

### 2.1 - Literature review of related work

One of the fundamental topics that Background Research needs to critically cover is Machine Learning. Which is a sub-branch of Artificial Intelligence which is defined as the automation of human tasks. (Ed et al, 2021)<sup>2</sup> What classifies it as a sub-branch is a distinguishing feature where specifically it uses data and algorithms to intimate human behaviour. (IBM, 2021)<sup>3</sup> This is contrasted with Symbolic Artificial Intelligence, where human behaviour or rulesets were specifically written into computer programs. It is often referred to as “classical or rule-based AI” (Ben Dickson, 2019)<sup>4</sup>

There is a great depth of technical terminology utilized within Machine Learning. As the field works with vast amounts of data in a complex mathematical and logical format. These terminologies are listed under Table 2.1, which is the table below this paragraph. These tables will be frequently used throughout this chapter to cover various topics.

Terminology	Definition
Label	A classification to define an output. (i.e “Cucumber” and a “Banana”)
Features	A variable that describes one of the outputs. (i.e “ <b>Banana is yellow</b> ” and a “ <b>Cucumber is long</b> ”)
Model	An algorithm or a sum that defines the relationship between labels and features. (i.e “The object in my hand is <b>long</b> and <b>yellow</b> , therefore it must be a <b>banana</b> )

*Table 2.1 - Basic Terminology Table (Google, 2021)<sup>5</sup>*

Upon the identification of distinguishing features within Machine Learning, it is discovered that there are many fields within Machine Learning alone. The three main categories are Unsupervised Learning, Supervised Learning and Reinforcement Learning.

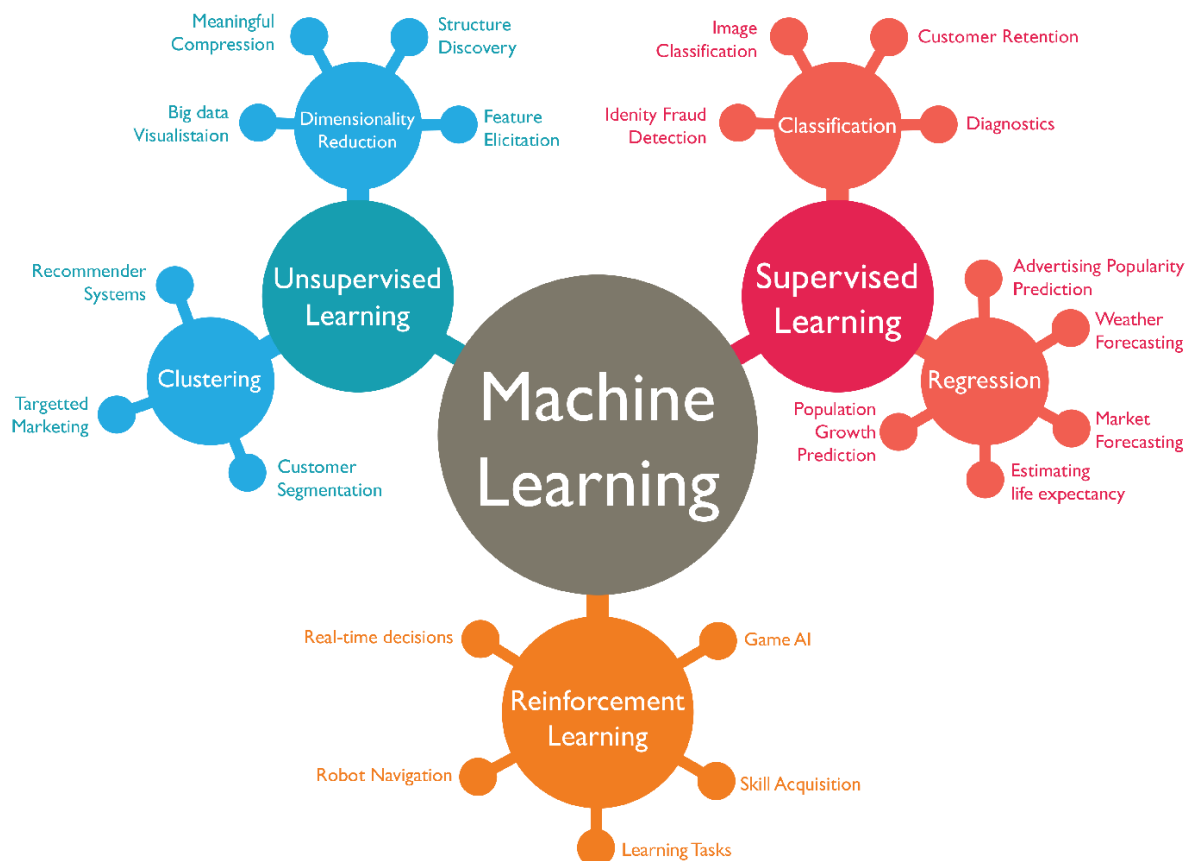


Figure 2.a - Branches of Machine Learning Diagram (AskDataScience, 2018)<sup>6</sup>

Supervised Learning is where programmers possess both features and labels, therefore, can train models based on collected data that defines the relationship between the two. This is further broken down into two sub-branches which is regression and classification. Regression is defined as the generation of a probability that produces a binary outcome. As an example, models predict if an object is a fruit or not based on provided features. (Google, 2021)<sup>7</sup>

A very basic example of regression is  $y = mx + b$ . Where values of either the y or x-axis correspond to either a label or feature. Providing the value for one of the axes allows for the equation to be solved for the other axis value. A visual example of this equation can be viewed under Figure 2.b.

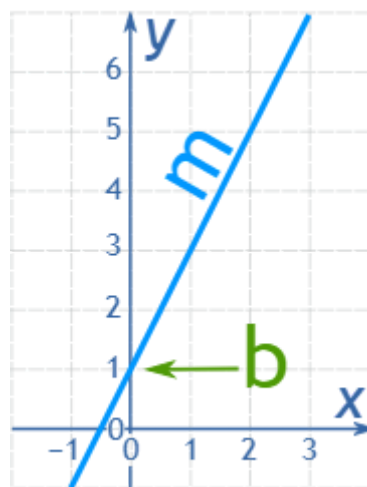


Figure 2.b - An example of  $y=mx+b$  (MathsIsFun, 2021)<sup>8</sup>

The complex nature of Machine Learning is presented when multiple features are introduced therefore a simple

2D graph isn't able to represent a model containing 10 features. An example of a 3D graph can be seen in Figure 2.b.1

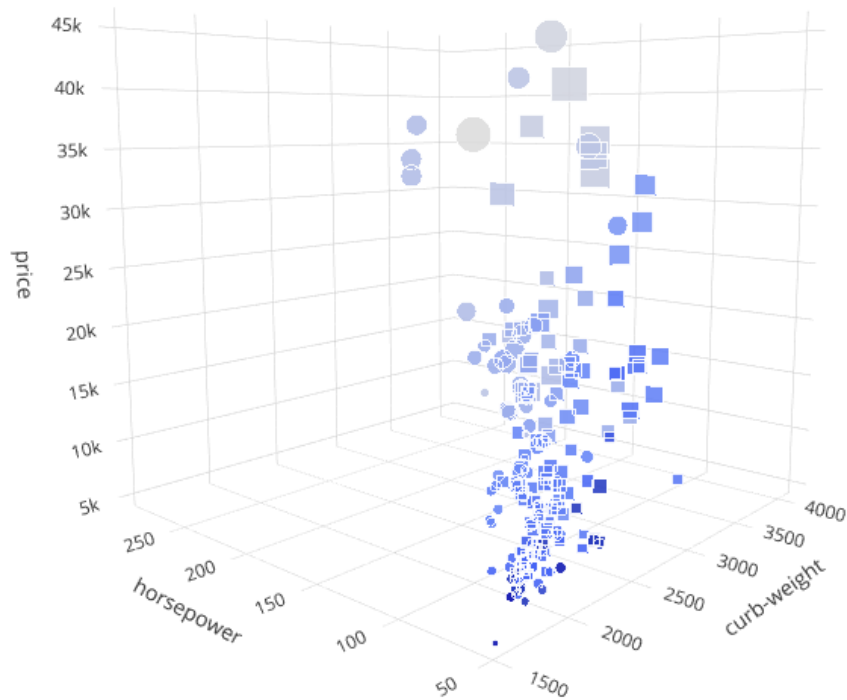


Figure 2.b.1 - An example 3D Graph (Prasad Ostwal, 2019)<sup>9</sup>

Classification in contrast uses regression as a foundation and applies a threshold to the equation which allows for the distinction of if an object is either x or y. For example, if the value is below 0.5. It is a banana. (Google, 2021)<sup>10</sup>

Specifically, we are using TensorFlow which is developed and maintained by Google. Which is a python library that provides the core functions and features necessary for machine learning.

Terminology	Definition						
Tensor	<p>Arrays that are multidimensional. Which is to account for various inputs and outputs that a model is subjected to. An example of this can be seen in Figure 2.c.</p> <div style="text-align: center;"> <div style="display: flex; justify-content: space-around; margin-bottom: 10px;"> <span>A scalar, shape: [ ]</span> <span>A vector, shape: [ 3 ]</span> <span>A matrix, shape: [ 3, 2 ]</span> </div> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"> <p>4</p> </div> <div style="text-align: center;"> <p>2.0 3.0 4.0</p> <p>3</p> </div> <div style="text-align: center;"> <p>3 {</p> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>1</td><td>2</td></tr> <tr><td>3</td><td>4</td></tr> <tr><td>5</td><td>6</td></tr> </table> <p>2</p> </div> </div> </div> <p>Figure 2.c - An example of tensors (TensorFlow, 2021)<sup>11</sup></p>	1	2	3	4	5	6
1	2						
3	4						
5	6						
Shape	<p>The arrangement of dimensions within a tensor. They are displayed within TensorFlow under the following format.</p> <p>{3, 2} - Three objects with two elements within each object.</p>						
Rank	The number of dimensions within a tensor.						

Session	The execution of parts or the entire graph.
Neural Networks	A form of MI that uses a layered representation of Data
Weights	Strength of each connection to a neuron
Bias	A constant applied to each layer of the neural network.
Activation Function	A function that is applied to the weighted sum of the neurons
Loss Function	A value that determines the distance between the desired values and the actual output value given by a model.  (i.e a model predicts 0.7, but we desire a 1. Therefore the Loss Function is equal to $0.3 = 1 - 0.7$ )
Gradient Descent	A process that aims to move the direction of the graph to achieve minimum loss.
Backpropagation	A process that updates all of the weights and bias of the model to achieve Gradient Descent
Optimizer	A function within TensorFlow that performs the processes of Gradient Descent and Backpropagation.
Hyperparameter	Values that the programmer can modify which can affect the model as a whole
Word Embedding	Representing text as numbers. As referred to as “vectorizing the text”

*Table 2.2 - TensorFlow Terminology Table (TensorFlow, 2021)<sup>12</sup> (IBM, 2021)<sup>13</sup> (Tutorialspoint, 2021)<sup>14</sup> (TensorFlow, 2021)<sup>15</sup> (TensorFlow, 2021)<sup>16</sup>*

TensorFlow operates with a format called Tensors. Tensors are defined by shape and rank. In which it creates graphs of defined computations that use definitions written from within the code. Tensors are the format that data is shaped into in order for it to be applied to the graph. Going by an analogy of Classical AI being applied to Machine Learning. Classical AI requires that the ruleset of the desired operations be written into code via a human. (i.e In a game of chess, a piece on the board is in check, move the piece out of check) TensorFlow attempts to replicate the ruleset without human intervention by creating graphs in which tensors can be applied to graphs which then can produce an output.

This graph can reach multiple dimensions which can be beyond perception for the human mind therefore it is better represented as a neural network. An example of this can be seen in Figure 2.C.

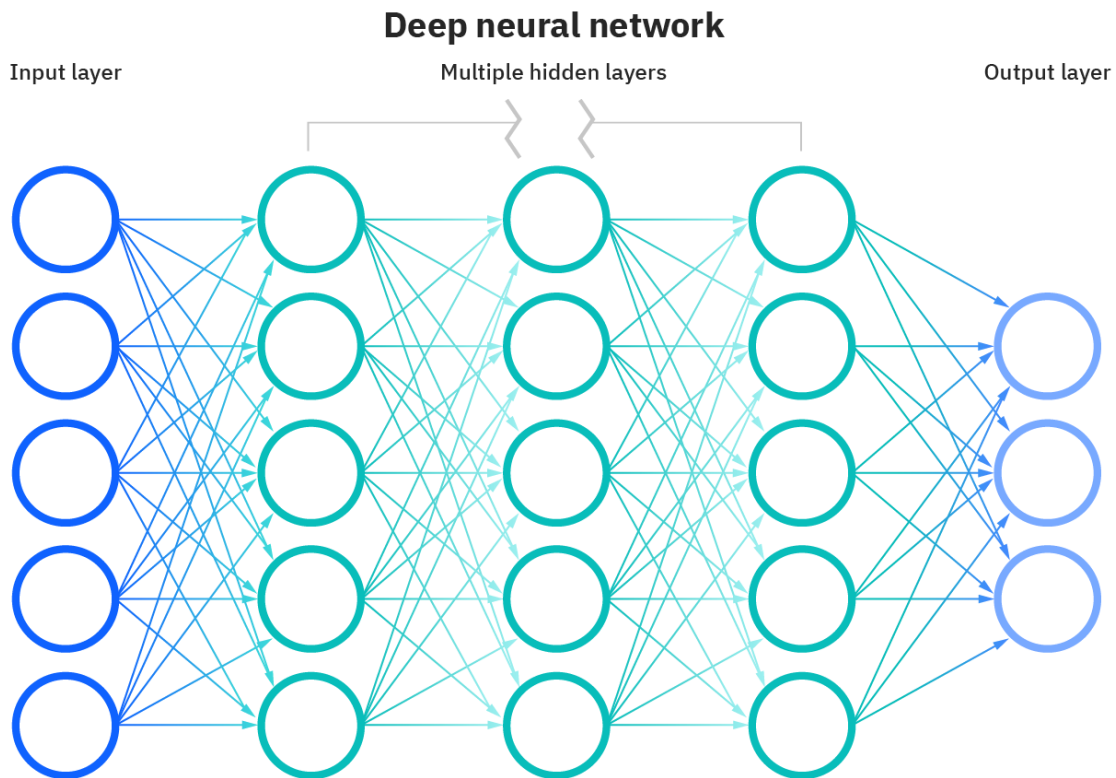


Figure 2.C - Diagram representing Neural Networks (IBM, 2021)<sup>17</sup>

Consisting of three layers which are input, hidden and output. The arrangement of the input and output layers are defined by the programmer to suit the model's objectives and goals. As by its namesake, the hidden layer is a layer that is unknown to the programmer. It holds the computations that define how the program operates. It is also the layer that rapidly adapts and evolves based on the data it trains from. It is a system of neurons connected by weights and bias. Which requires various functions such as Activation Function, Loss Function and Gradient Descent to maintain the model. (IBM, 2021)<sup>18</sup>

The field of Steganography is considered to be the practice of placing hidden messages in plain sight. Almost akin to invisible ink on a piece of paper. One such example of publicly available tools is wbStego which allows for files to be hidden within other files. (wbStego, 2004)<sup>19</sup> It is of the student's opinion that information can simply be hidden within files using a simple hex editor. Tools merely automate this process and displace the logic needed to perform such functions down to simple button presses. Entire files can be embedded within files. Hidden messages within jpeg images. One such example of this method for images is referred to as Least Significant Bit. In which the last bit of a pixel is used to conceal data. (BoliteAKlou, 2018)<sup>20</sup>

## 2.2 - Critical evaluation of related products/solutions

The research conducted into Machine Learning is considered fundamental as the programming basics needed to write within Python. As there are more requirements than simply learning the syntax of the python libraries. Mathematics is a critical fundamental as the model operates based upon key concepts from it. There is a greater emphasis that processing the data requires more effort rather than the creation of the model. The graphs within TensorFlow cannot operate if the data inputted isn't formatted in a manner that allows for the computations to proceed.

A greater effort was placed into the exploration of Machine Learning rather than Steganography rather research into steganography would be only beneficial if the project was a continuous effort. In accordance with the introduction written in Chapter 1. This project aims to promote the use of Machine Learning within the field of digital forensics and not make any advancements as this exceeds the resources available to the student.

However, this project isn't completely abandoning the need for expertise and knowledge from Digital Forensic.

As the field of Anti-Forensic which runs in direct contrast is engaged in a pseudo arms race. The utilization of Machine Learning to train an adaptive program to detect hidden steganography measures could be viewed as an advancement in favour of Digital Forensics.

### 2.3 The scope of the project

As iterated within the introduction section of this document. This scope of the project is particularly aimed at steganography. Therefore the focus of the research is primarily targeted at resources, knowledge and tools needed to achieve the project goals. Specifically, this is the research into the practice of stenography, how it functions and how users utilize it in a real-world scenario.

In addition, we limit the scope of this project via this document's exploration of Machine Learning. Machine Learning is a sub-branch of Artificial Intelligence. There are many branches within Machine Learning therefore when in the design phase of the project. We limit the scope of the research by only researching surface-level details of the many branches of Machine Learning. If the surface-level details provide a probable solution to the problem that our project provides. Further research is then conducted to determine if they should be selected as a key component of the project. As this is how text classification was chosen.

This research approach determines what is suitable to dedicate time to. Which limits time waste within the project. As we quickly determine what is needed, why it is needed and its place within the project. Effort vs Reward is also a factor within determining whether a topic was worthy of investing time in.

In addition, the scope of the research was determined if the information was of any value to the success of this project. For example, whilst efforts and time could be invested into learning the methodologies of steganography. It would not yield any benefit as we are training a model, not creating the model ourselves similar to classical AI.

### 2.4 - Review and justification of theories/models/development platforms/tools selected for use in the project

The choice of the TensorFlow library is primarily due to the usage of Google's service as a platform "Google Collaborate" and its extensive documentation. Such as their quick start guide. Whilst other python libraries do offer documentation. They didn't offer resources on the core principles and fundamentals that a beginner would need to comprehend the functions within their libraries.

Likewise, the time invested into background research was invested in Supervised Learning as it fulfilled specifications of the project. As a summary of the other two branches of Machine Learning. Unsupervised learning is in contrast to supervised learning. Simply training a model without labels within the data set. (IBM, 2021)<sup>21</sup> Whilst Reinforcement learning is the training of a model within an environment, utilizing an agent in which the model controls. (Błażej Osiński and Konrad Budek, 2018)<sup>22</sup> As explained, Unsupervised learning and Reinforcement learning aren't applicable as they don't fulfil the requirements of the project or exceed the scope of the project as defined under Chapter 2.3.

The following core Python Libraries that were chosen are TensorFlow and Keras. For the sake of brevity, the justification for TensorFlow is contained within Chapter 2.1. Which consists of a string of topics that describes the structure of machine learning, overly simplified mathematics for the sake of the reader, tables of terminology and principles which govern how machine learning functions internally. On the other hand, Keras was chosen as it is an additional module run alongside TensorFlow. Which simplifies various mathematical aspects and the complex nature associated with TensorFlow. As stated on their website, it was designed in mind with quick experimentation and flexibility for programmers. (Keras, 2021)<sup>23</sup>

Google Collaborate is chosen as the primary development platform. The biggest point of interest is that it can be run from any modern web browser. This allows for development from various physical locations. Be it a personal computer from the student's living residence, a computer from the university's IT suites or even a computer from a cafe. The hardware that performs the computations of the model is done by using data centres created by Google. This allows for portability as almost any computer with a modern web browser and an internet connection can be used as a workstation by the student.

The following two programs intended for use to create the sample data to train the model are QuickStego and OpenStego. These programs were chosen as they offered vast features without the need for purchasing any licenses. As well as a simplified GUI that is intended for beginners or users intending for simple operations.



(cybernescence ltd, 2020)<sup>24</sup> (Samir Vaidya, 2021)<sup>25</sup> Python as a programming language was chosen due to being familiar with the student as well as offering vast resources that aim to grant greater insight and knowledge to beginner users. In a similar vein to TensorFlow (Srinivas Ramakrishna, 2021)<sup>26</sup>

## Chapter 3: Requirements Analysis and Specification

---

It should be explicitly stated that the program doesn't serve to replace human involvement or supplant human investigators. It is an aid in the decision making of human investigators in terms of an advisory role.

The program is intended to assign a probability value to a sample determining if the sample is suspected of concealing steganographic measures. Any samples above a certain threshold will be flagged for manual review by a human. As well as that, operators of the program can lower the threshold on predictions if they feel that the program isn't producing suitable results.

The program is also expected to continuously be fed additional data in order to develop and adapt to the changing times. In a real-world scenario, it would be fed actual steganography examples. Perhaps within a law enforcement environment. It could be fed actual examples produced by actual criminal enterprises. Rather than mock samples produced by a student using free tools found on the web.

In addition, it is also expected to have the necessary features to standardize, format and preprocess input data so that the model can receive said data. Be able to assign features and labels. Then proceed with training and developing the model. Therefore the only requirements of the end-user are to only have a basic knowledge of how machine learning functions, the hardware needed to provide processing power and the physical samples to feed the learning algorithm to learn from.

## Chapter 4: Software Design

---

### 4.1 - User-Interface design

It should be noted that a graphical user interface is low in terms of prioritization. As the program is intended for functionally rather than accessibility. In addition, it should be noted that the program doesn't significantly benefit from any GUI elements.

The program is intended to be run in a command-line interface with prompts and a help output provided to the user. Whilst minor visual aids within the command line interface could be encoded into the program such as a progress bar and splash screen for a visual aid to the end-user. These are also low on the priority list. It is generally accepted that most highly specialised tools within the Field of Forensics operate within a command-line interface. Especially if they are in an experimental phase not intended for mass adoption. The majority of Forensic Investigators would have the necessary knowledge and experience of operating within such an interface.

### 4.2 - Database tables' structure design

As shown within Figure 4.aa. We can see that the starting point for the data is when it is received as a URL from HTTP Address. Currently, the sample data is hosted on a cloud platform which is to support the portability of Google Collaborate. This means that the student doesn't have to carry a physical medium with them. As long as the student has a functioning connection to the internet which is a requirement for Google Collaborate regardless. The sample data can be simply downloaded to the program. The program will be able to support importing of sample data from the host computer as well.

In addition, the sample data is simply split into three respective categories in which they operate within their respective function.

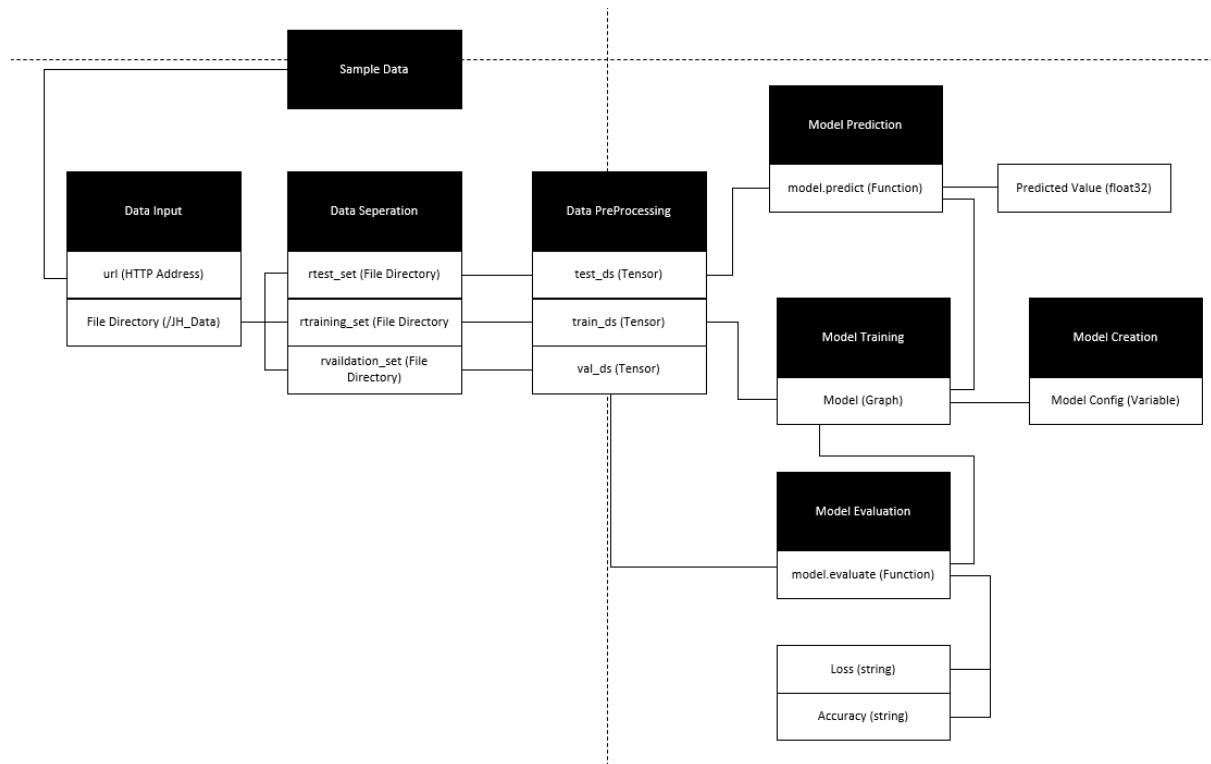


Figure 4.aa -Database table structure design diagram.

### 4.3 - Main components of the software architecture

#### Prototype Phase - Dec 2021

The code structure of the program is finalized with only minor modifications and optimizations expected to follow throughout the course of the project. It should be stated that any code presented under this section of the documentation is subject to changes and may not be in the final build presented.

For the sake of brevity, the entire code isn't displayed as they are only used to aid in the understanding of definitions of each program structure.

[Module Import]  
 [Data Input]  
 [Data Separation]  
 [Data Pre-processing]  
 [Model Creation]  
 [Model Training]  
 [Model Evaluation]  
 [Model Predictions]

Figure 4.a - Program Structure

**Module Import** - This section is dedicated to importing the python libraries necessary for the program to function.

As shown in Figure 4.b, we define which python libraries are needed to be imported into the code using the import syntax.



The sample data will already be separated into categories via different directories therefore we would only need to specify in code which directories are intended for which purposes.

**Data Pre-processing** - This section processes the data so that it can be received by the model in a standardized format.

```
vectorize_layer = layers.TextVectorization(  
    standardize=custom_standardization,  
    max_tokens=max_features,  
    output_mode='int',  
    output_sequence_length=sequence_length)
```

*Figure 4.f - An example of text vectorization*

**Model Creation** - This section creates the model which in turn defines the computations and relations between label and features.

Under the variable “model”. The code can specify what layers should be created for said model. Specifying first the input layer then the output layer. Each layer is represented by a single line for clarity. Such as “layers.dense”. We then call upon the layer written as “model.compile” in which the code specifies how the model will perform gradient descent and backpropagation. This can be seen under Figure 4.g.

```
embedding_dim = 16  
  
model = tf.keras.Sequential([  
    layers.Embedding(max_features + 1, embedding_dim),  
    layers.Dropout(0.2),  
    layers.GlobalAveragePooling1D(),  
    layers.Dropout(0.2),  
    layers.Dense(1)])  
  
model.compile(loss=losses.BinaryCrossentropy(from_logits=True),  
              optimizer='adam',  
              metrics=tf.metrics.BinaryAccuracy(threshold=0.0))
```

*Figure 4.g - Compilation of the model*

**Model Training** - This section inputs the processed data. In turn, this trains and develops the model Which defines the relationship between features and labels.

To call the function to start the training process of the model. The code uses “model.fit”. In which we specify the training data, validation data and the number of epochs. Epochs are the number of iterations the model will train on the dataset. This can be seen in Figure 4.h

```
epochs = 10
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs)
```

*Figure 4.h - Model training function*

**Model Evaluation** - This section inputs another dataset referred to as “Evaluation Set” which determines the accuracy and quality of the trained model.

To call the function to start the evaluation process. The code will simply use the “model.evaluate” where we only need to specify the data that it will be evaluated upon. The code will then be able to output values corresponding to loss and accuracy. This can be seen in Figure 4.i

```
loss, accuracy = model.evaluate(test_ds)

print("Loss: ", loss)
print("Accuracy: ", accuracy)
```

*Figure 4.i - Model Evaluation Function*

**Model Predictions** - This section allows for additional samples to be processed and run through the trained model. In which the trained model can make a prediction

Lastly, we can utilize the trained model as the code simply calls upon the model as it is stored under a variable. Under the export\_model variable, we call upon the trained model, how the predicted data will be processed and the activation function. Bear in mind, we must use the exact same process for processing data when processing prediction otherwise it may result in invalid values. The model is then compiled and the function to call for a prediction is under “export\_model.predict”. It should be noted that all predictions must be in the form of arrays as TensorFlow is intended to make a mass amount of predictions rather than single predictions. This can be seen under Figure 4.j

```
export_model = tf.keras.Sequential([
    vectorize_layer,
    model,
    layers.Activation('sigmoid')
])

export_model.compile(
    loss=losses.BinaryCrossentropy(from_logits=False), optimizer="adam", metrics=['accuracy']
)

examples = [
    |
]

export_model.predict(examples)
```

*Figure 4.j - Model Prediction Function*

## 4.4 - Detailed Software Class Design

### Code Breakdown

#### SampleCreator.py

As shown in Figure 22A below. We see various syntaxes, statements and variables. Currently the program imports the following libraries which are “os”, “sys” and “pathlib”. This grants the functionality for the program to read, write and modify files. In addition, variables related to file management are initialised here.

“Dir” - Which simply gathers the current directory that the py file is contained within. Using the “dirname” syntax.

“Path” - Appends a specific folder to the current directory. Using the “join” syntax

“Files” - Gathers a list of all files contained within a directory listed in the file. Using the “listdir” syntax.

In addition, an IF statement is run where it checks if a folder “CFolder” and “RFolder” is present. If it is not, then it is created using “makedirs” syntax. This snippet of code ensures that the following functionality named relative files is present in the program. This allows for files in proximity to the py file to be stored to be used rather than absolute file paths. I.e “D:\file\folder”. This program in particular targets the CFolder which should be created next to the py file. I.e “D:[Where the .pyfile is stored]\CFolder”

```
1  import os
2  import sys
3
4  from pathlib import Path
5
6  if not os.path.exists('CFolder'):
7      os.makedirs('CFolder')
8
9  if not os.path.exists('RFolder'):
10     os.makedirs('RFolder')
11
12  dir = os.path.dirname(__file__)
13  path = os.path.join(dir, 'CFolder')
14  path_ii = os.path.join(dir, 'RFolder')
15  files = os.listdir(path)
```

Figure 22A - Screenshot #1 of Code Snippet from SampleCreator.py

As shown in Figure 22B below. This snippet of code simply runs a FOR loop. In which it iterates through all the files and changes their extension to a .txt. It simply lists all of the required directories and then renames all files listed within that list using the following syntaxes of “join” and “rename”. When files are converted to the extension “txt”. It provides the raw binary data of the file. However, it varies how the file is rendered depending upon which application opens the said file. For example, if the file was opened via Windows standard “Notepad” it would render each character as if it was represented via UNICODE or what keyboard standard is present on said computer. Using specialist software, it would present raw binary data in human readable format.

```
# --- File extension editor ---
for index, file in enumerate(files):
    os.rename(os.path.join(path, file), os.path.join(path, ''.join([str(index), '.txt'])))
```

Figure 22B - Screenshot #2 of Code Snippet from SampleCreator.py

As shown in Figure 22C below. This snippet of code is a defined function where it reads an inputted txt file. Then converts all of it's contents from the raw binary data to a human readable hex value. It would then be formatted and spaced out accordingly. This is done so that the algorithm would interpret raw hex values. Not the characters assigned via UNICODE. Upon a successful conversation, a print statement is issued.

```
21 # --- .txt to .bin to .txt converter --
22 def func_convert(filename):
23     container = []
24     with open(filename, 'rb') as file:
25         while True:
26             DL = file.read(1)
27             if not DL:
28                 break
29             container.append(int.from_bytes(DL, byteorder='big'))
30             filename = str(ii) + ".txt"
31             with open(os.path.join(path_ii, filename), 'w') as file:
32                 for i in container:
33                     file.write(str(hex(container[i]))+ " ")
34             print(str(ii) + ".txt has been created.")
35             return container
```

Figure 22C - Screenshot #3 of Code Snippet from SampleCreator.py

As shown in Figure 22D below. This snippet of code actually calls upon the functions of the program.

First off, it refreshes current file paths next to the program. Loading them into variables.

In which a FOR loop is run. It would go through every file contained within "CFolder" and output towards "RFolder". Upon the successful completion of the converted file. The ii variable is iterated by 1 and the process is repeated until all files are converted. Then it deletes files that was converted in order to save space. Then the program concludes and shuts down.

It is written within a Try/Catch block. As an error in which files weren't identified due to the fact that they were modified meant technically the file didn't exist. In which the program is commanded to simply restart the program to load the proper file paths.

```

37 # --- call to program --
38 ii = 0
39 try:
40     dir = os.path.dirname(__file__)
41     path = os.path.join(dir, 'CFolder')
42     path_ii = os.path.join(dir, 'RFolder')
43     files = os.listdir(path)
44
45     for index, file in enumerate(files):
46         func_convert(os.path.join(path, file))
47         os.remove(os.path.join(path, file))
48         ii+=1
49
50 except FileNotFoundError:
51     os.execl(sys.executable, sys.executable, *sys.argv)
52

```

Figure 22D - Screenshot #4 of Code Snippet from SampleCreator.py

#### Google Collaborate: Machine Learning Algorithm

As shown within Figure 22E, within this code snippet are merely the libraries that are required for this program. Which includes “os” and “tensorflow”.

```

import os
import tensorflow as tf

from tensorflow.keras import layers
from tensorflow.keras import losses

```

Figure 22E - Screenshot #1

As shown within Figure 22F. This code block enables for the downloading of sample data from a hosted website. Firstly, it loads the url as a variable. Then the syntax “utils.get\_file” downloads the file and applies it to the folder name “JH\_data”. It is then applied to another variable.

```

url = "https://www.mediafire.com/file/wrcp31t4qcvauq3/JH_data.gz/file"

full_dataset = tf.keras.utils.get_file("JH_data", url, untar=True, cache_dir='.', cache_subdir='')

full_dataset_dir = os.path.join(os.path.dirname(full_dataset), 'JH_data')

```


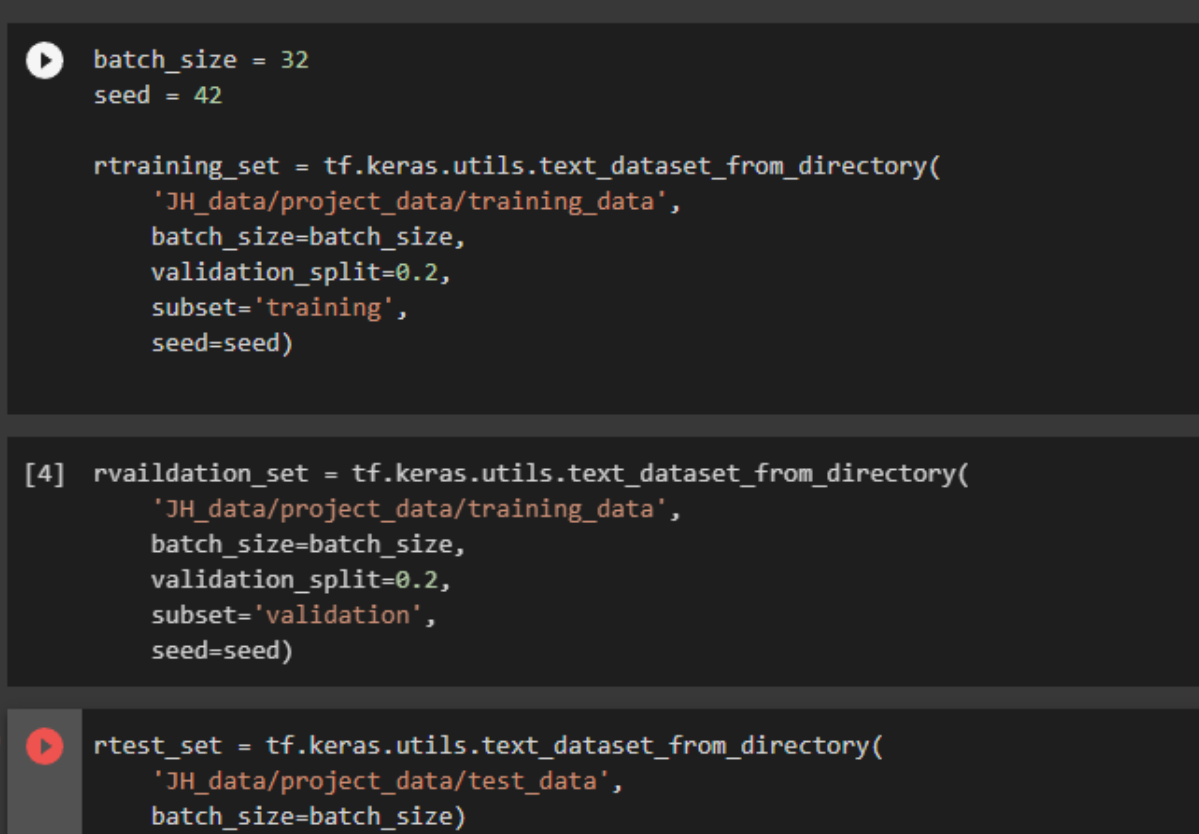
 Downloading data from [https://www.mediafire.com/file/wrcp31t4qcvauq3/JH\\_data.gz/file](https://www.mediafire.com/file/wrcp31t4qcvauq3/JH_data.gz/file)  
13590528/13584645 [=====] - 1s 0us/step  
13598720/13584645 [=====] - 1s 0us/step



Figure 22F - Screenshot #1

As shown within Figure 22G. Using folder structure, variables are assigned to specific folders. A specific syntax is used to configure how the sample data is used within the program. Such as, “validation\_split” which cuts specific amounts of the sample data from being used. As well as, “batch\_size” which determines how much sets of samples are fed to the program. This can be seen under “rtest\_set”, where it is defined as the set of sample data that is being used for testing purposes. There aren’t any specific configurations as we would want to use the entire set without obstructions.



```
▶ batch_size = 32
seed = 42

rtraining_set = tf.keras.utils.text_dataset_from_directory(
    'JH_data/project_data/training_data',
    batch_size=batch_size,
    validation_split=0.2,
    subset='training',
    seed=seed)

[4] rvaildation_set = tf.keras.utils.text_dataset_from_directory(
    'JH_data/project_data/training_data',
    batch_size=batch_size,
    validation_split=0.2,
    subset='validation',
    seed=seed)

▶ rtest_set = tf.keras.utils.text_dataset_from_directory(
    'JH_data/project_data/test_data',
    batch_size=batch_size)
```

Figure 22G - Screenshot #2

As shown within Figure 22H. It is a further step to processing the data. In which the sample data is “vectorized”. Essentially, it can be considered the connection between allowing the program to interpret the sample data properly and to train the program. First, the sample data is given definitions that can categorise the sample data. Then, the program is given the training set to aid in the creation of the program’s algorithm via the “adapt” syntax. Lastly, the relevant defined functions are called and applied to the sets of sample data.

```

▶ max_features = 10000
sequence_length = 250

vectorize_layer = layers.TextVectorization(
    max_tokens=max_features,
    output_mode='int',
    output_sequence_length=sequence_length)

train_text = rtraining_set.map(lambda x, y: x)
vectorize_layer.adapt(train_text)

def vectorize_text(text, label):
    text = tf.expand_dims(text, -1)
    return vectorize_layer(text), label

train_ds = rtraining_set.map(vectorize_text)
val_ds = rtraining_set.map(vectorize_text)
test_ds = raw_test_ds.map(vectorize_text)

```

Figure 22H - Screenshot #3

As shown within Figure 22I. This code block simply keeps the sample data loaded in memory rather than reading it off the storage medium that the sample data is hosted on. In addition, it dynamically changes the value of the amount of data fed towards the model.

```

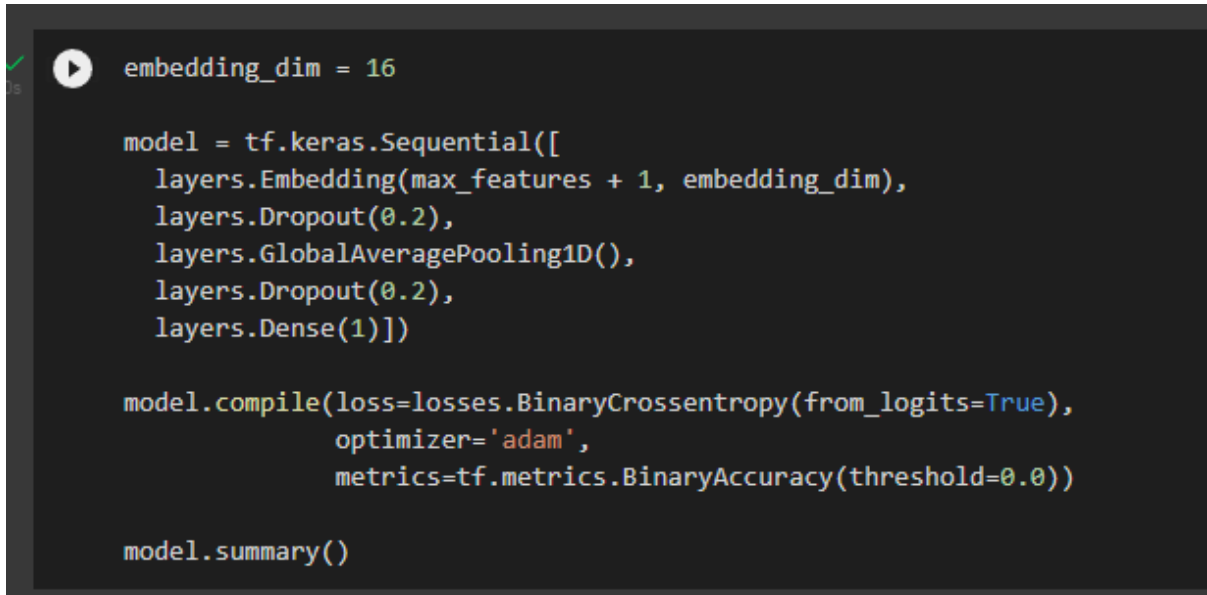
▶ AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
test_ds = test_ds.cache().prefetch(buffer_size=AUTOTUNE)

```

Figure 22I - Screenshot #4

As shown within Figure 22J. This is where the model is structurally created. Various configurations are listed which define how layers are created functionally and it then calls for the model to be created and then printed as a summary.



```

embedding_dim = 16

model = tf.keras.Sequential([
    layers.Embedding(max_features + 1, embedding_dim),
    layers.Dropout(0.2),
    layers.GlobalAveragePooling1D(),
    layers.Dropout(0.2),
    layers.Dense(1)])

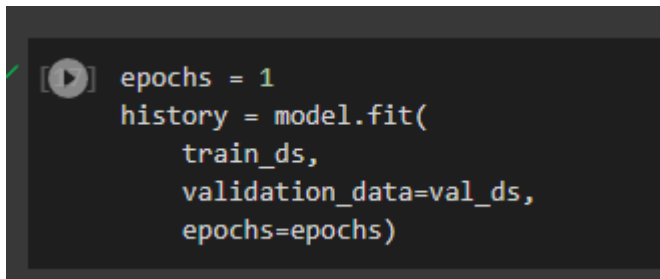
model.compile(loss=losses.BinaryCrossentropy(from_logits=True),
              optimizer='adam',
              metrics=tf.metrics.BinaryAccuracy(threshold=0.0))

model.summary()

```

Figure 22J - Screenshot #5

As shown within Figure 22K. The model is then called to train with the training data and validate its results using the test data. “epochs” is the variable which defines how many rounds the model should train and iterate upon the sample data. The results of this is printed out as an output of the code block.



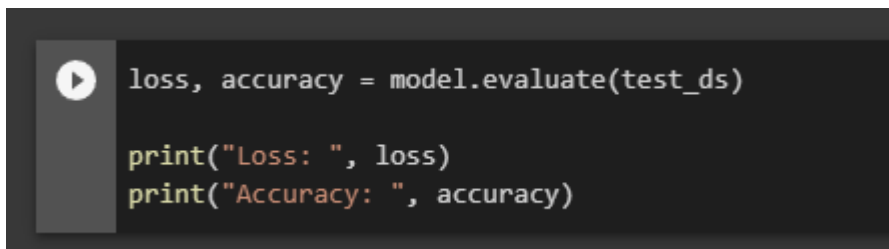
```

epochs = 1
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs)

```

Figure 22K - Screenshot #6

As shown within Figure 22M. The model is then fed the test data. In which, the models will test with the data to see if the model can predict if the samples have steganography measures or not. The results of this are outputted.



```

loss, accuracy = model.evaluate(test_ds)

print("Loss: ", loss)
print("Accuracy: ", accuracy)

```

Figure 22M - Screenshot #7

As shown within Figure 22N. The model is then called upon to go through the contents of “PFolder”. In which, it will highlight any file it suspects of steganographic measures.

```

export_model = tf.keras.Sequential([
    vectorize_layer,
    model,
    layers.Activation('sigmoid')
])

export_model.compile(
    loss=losses.BinaryCrossentropy(from_logits=False), optimizer="adam", metrics=['accuracy']
)

with open ("modelpredict.txt", "r") as myfile:
    modelpredict = myfile.readlines()

export_model.predict(modelpredict)

path = "/content/PFolder"
files = os.listdir(path)
iii = 0

for index, file in enumerate(files):
    try:
        input = os.path.join(path, str(files[index]))
        with open (input, "r") as myfile:
            modelpredict = myfile.readlines()

        predictval = export_model.predict(modelpredict)
        predictval = predictval
        threshold = 0.70

        if predictval > threshold:
            iii+=1
            print(predictval)
            print("Prediction for: " + str(files[index]))

    except IndexError:
        continue
    except IsADirectoryError:
        continue

if iii <= 0:
    print("No Files Detected: Consider lowering Threshold")

```

Figure 22N - Screenshot #8

Lastly, As shown within Figure 22N. This code snippet allows for the algorithm itself to be saved as a “checkpoint”. In which the user can download this folder and reupload it to the program. If they desire to continue development of the model.

```

[12] model.save_weights('./checkpoints/my_checkpoint')

model.load_weights('./checkpoints/my_checkpoint')

```

Figure 22O - Screenshot #9

#### 4.5 - Use Cases realisation

The program itself is rather very linear. Due to many aspects during multiple stages of its development to ensure that there was a consistent and immutable experience for its users. As documented in Artefact Documentation, there are specific functions for specific roles. As shown in Figure 4.1 and 4.2. They are the activity diagram when the user operates the expected programs.

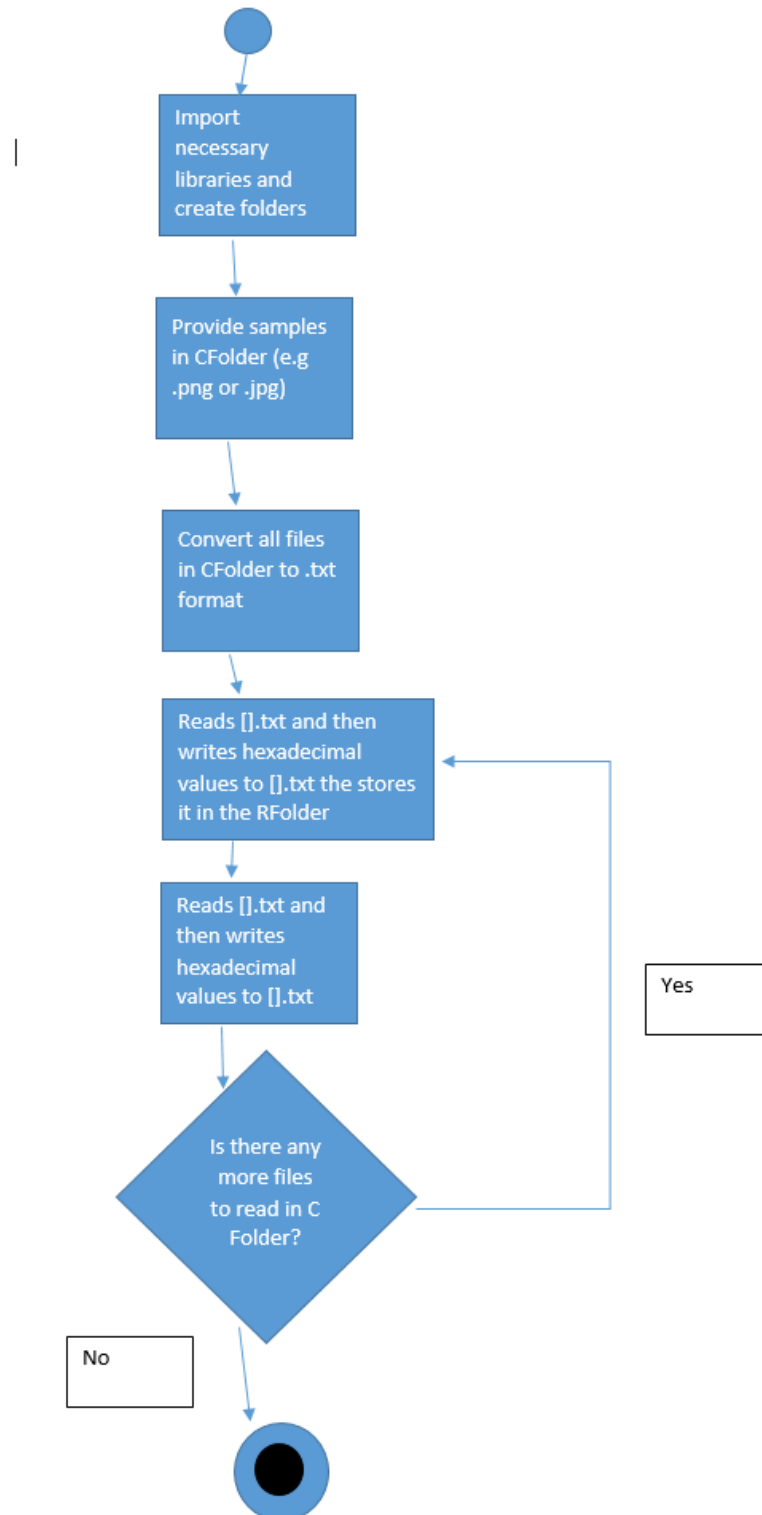


Figure 4.1 - Activity Diagram of SampleCreator.py

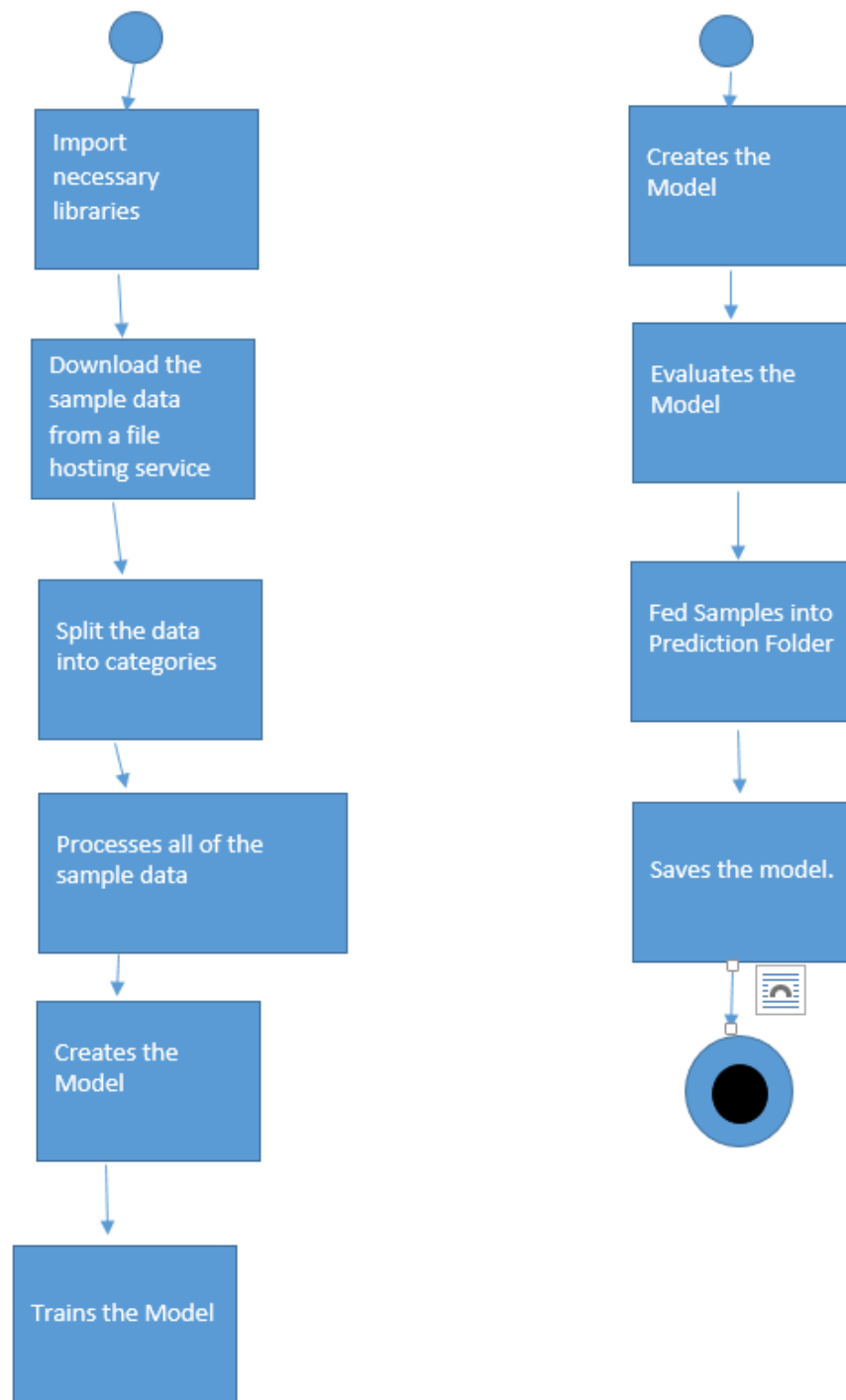


Figure 4.2 - Activity Diagram of Main Program

# Chapter 5: Implementation and Testing or Research Results and Result Analysis

## 5.1 - Software Implementation

A majority of the Implementation was done using the SCRUM framework. It is a very familiar workflow to the student and was mainly used for teams. It can be applied to a singular operator on a project. SCRUM is as described in brief.

There is a product backlog, sprints and tasks. The product backlog needs the product broken down into elements of the list. This is further broken down into sprints or known as tasks. This results in a cycle of “sprints/tasks” which results in iterations of a product. Eventually all of the product backlog is fulfilled which results in a final product. An diagram that demonstrates this concept can be seen in Figure 5.1

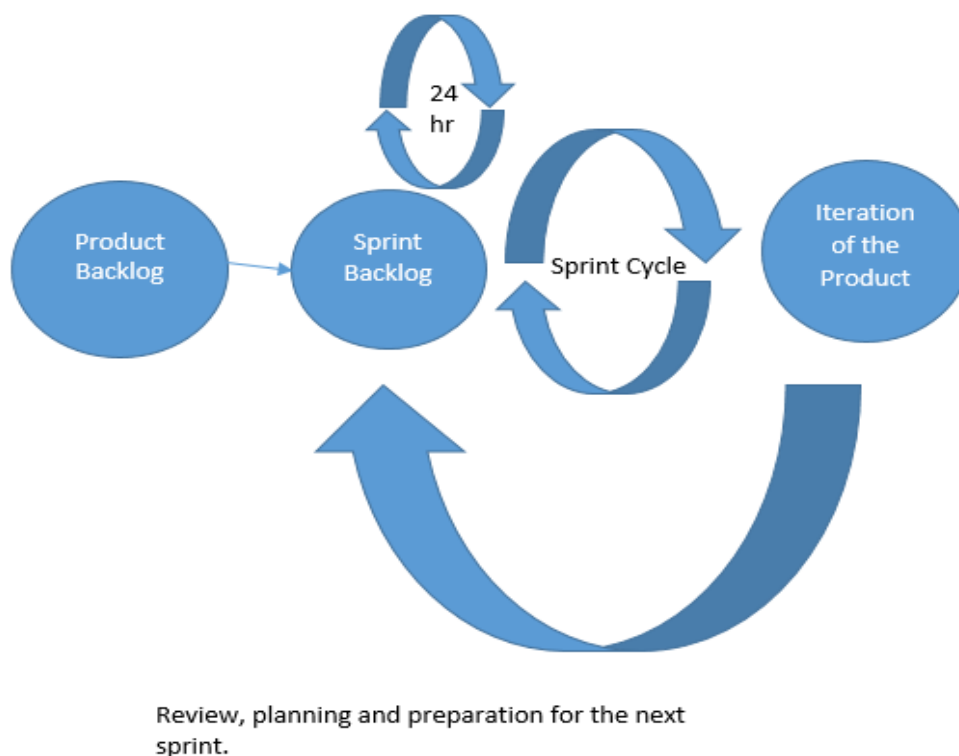


Figure 5.1 - Diagram of SCRUM process.

In addition, many resources were utilised during the Programming phase of the project. Namely, the TensorFlow and various Python Library API documentation. The final product only makes use of two python libraries for full functionality. This being namely the OS library <sup>1</sup>a and the TensorFlow library <sup>2</sup>B. In addition, Documentation and Support Articles relating to the functionality and systems of Google Colaboratory <sup>3</sup>C. Links to the specific documentation utilised during the development of this product was listed under a specific heading in the Reference section of this documentation.

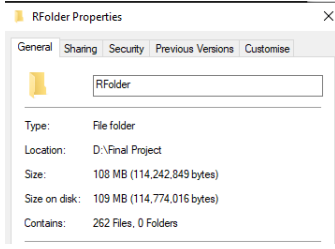
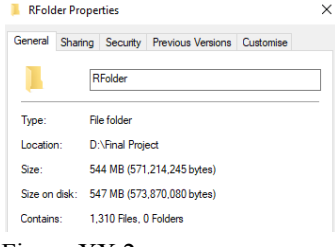
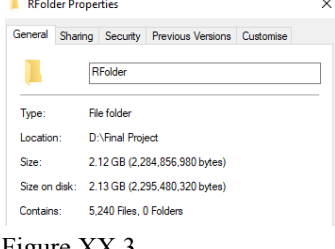
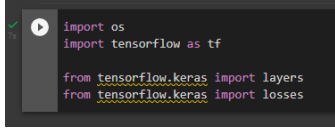
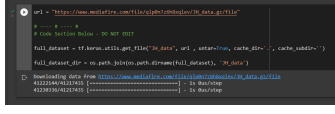
Alternative methodologies and alternatives to TensorFlow were considered, however the justification of the use of SCRUM with Google’s TensorFlow was that this enabled adaptability and ease of use for the student in question. As other machine learning libraries would’ve required additional development overhead. Such as, configuration of additional systems including the student’s host system and development workflow.

Google’s Colaboratory offers a simple package with all of the configuration done and provided by Google themselves on an online platform easily accessible to the student. The sheer amount of context of this statement

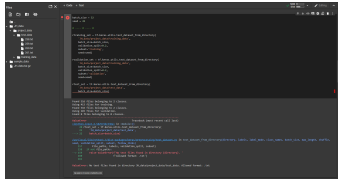

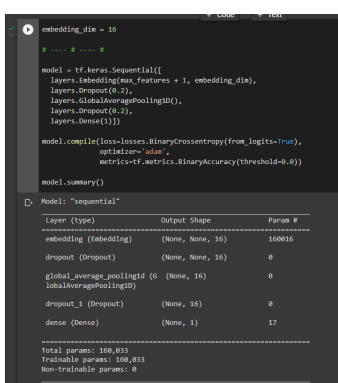
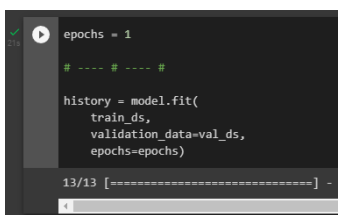
showcases flexibility and quality of Google's Platform "Colaboratory" and it can be argued it played a vital role in the success of this project.

## 5.2 - Software Testing

The following test plan was created. To ensure that the core modules of the program are functioning as envisioned. This can be seen in Table 5.1.

Test Goal	Expected Outcome	Actual Outcome	Comments/Screenshots
Conversion of 20 MB worth of Samples (SampleCreator.py)	Produced .txt files containing hexadecimal values of the original files.  The size of RFolder is expected to increase by a factor of 3.	As expected. However, the size of RFolder increased by a factor of 5. Resulting in a folder size of 108 MB	 Figure XX.1
Conversion of 100 MB worth of Samples. (SampleCreator.py)	Produced .txt files containing hexadecimal values of the original files.  The size of RFolder is expected to increase by a factor of 5.	As expected.	 Figure XX.2
Conversion of 1 GB worth of Samples (SampleCreator.py)	Produced .txt files containing hexadecimal values of the original files.  The size of RFolder is expected to increase by a factor of 5.	The program was interrupted as the host machine suffered a crash.  It appeared that it produced 2 GB of samples before suffering a crash.	Further testing was suspended due to the health of the Host Machine.   Figure XX.3
Code Snippet: Library Import	No errors produced. A green tick mark to indicate that the execution of the code snippet was successful	As expected.	 Figure XX.4
Code Snippet: Pulling the data	No errors produced. A green tick mark to indicate that the execution of the code snippet was	As expected.	 Figure XX.5



	successful. There should also be output in the console to show that this was successful		
Code Snippet: Separating Data into sets	No errors produced. A green tick mark to indicate that the execution of the code snippet was successful	An error was produced declaring that there were no .txt files present in the data. Despite that there was.	 <p>Figure XX.6</p> <p>Edits have been made to the code to allow for continued testing and operation of the program.</p>
Code Snippet: Processing the data	No errors produced. A green tick mark to indicate that the execution of the code snippet was successful	As expected.	 <p>Figure XX.7</p>
Code Snippet: Model Creation	No errors produced. A green tick mark to indicate that the execution of the code snippet was successful	As expected.	 <p>Figure XX.8</p>
Code Snippet: Model Training	No errors produced. A green tick mark to indicate that the execution of the code snippet was successful	As expected.	 <p>Figure XX.9</p>
Code Snippet: Model Evaluation	No errors produced. A green tick mark to indicate that the	This test couldn't be conducted due to error in Test	N/A

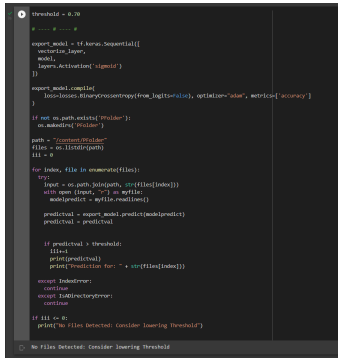
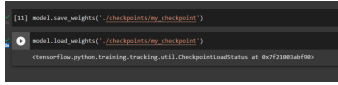
	execution of the code snippet was successful	“Code Snippet: Separating Data into sets”	
Code Snippet: Model Prediction	No errors produced. A green tick mark to indicate that the execution of the code snippet was successful	As expected.	 <p>Figure XX.10</p>
Code Snippet: Reloading/Saving the Model	No errors produced. A green tick mark to indicate that the execution of the code snippet was successful	As expected.	 <p>Figure XX.11</p>

Table 5.1 - Testing Table

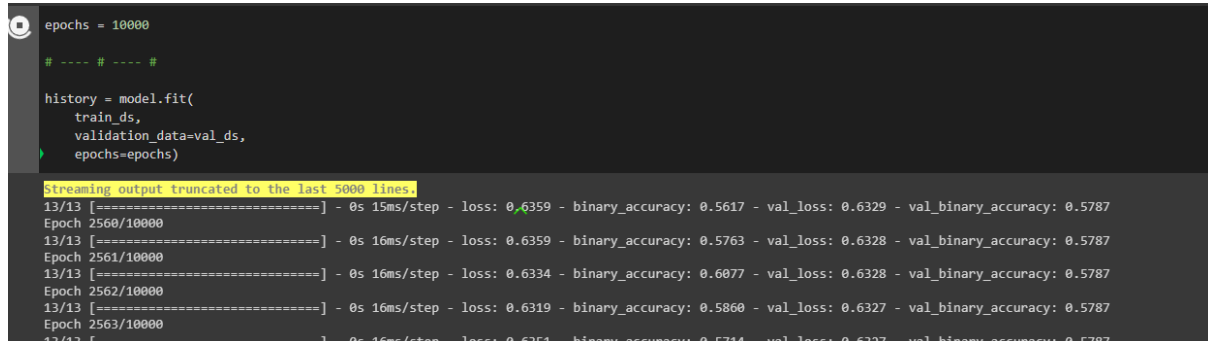
## Chapter 6: Evaluation of Results

There is a clear distinction between goals of the program and the capabilities of the program. This distinction is made primarily to reinforce and set what would be considered to the acceptable standards of the program. This being if the initial statement written in Chapter 3 of this document was met.

“The program is intended to assign a probability value to a sample determining if the sample is suspected of concealing steganographic measures” - Excerpt from Chapter 3

The program is indeed capable of assigning said value and indeed it did assign a value. However, the value it assigned to the file in question was logically incorrect. Whilst the adage “a bad workman blames his tools” rings true. This is a rare circumstance where sample data is a critical factor in operation of this program.

As shown in Figure 6.1 below. A separate test was conducted which was intended to see if the **goals**, not the capabilities of the program could be realised with initial sample data offered. However, it showed that the goals of the program weren't released. As it showed after 2000 iterations of analysis on the sample data done by the program. It was stuck on an accuracy rating of 57.87 percent.



```

epochs = 10000

# ---- # ---- #

history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs)

Streaming output truncated to the last 5000 lines.
13/13 [=====] - 0s 15ms/step - loss: 0.6359 - binary_accuracy: 0.5617 - val_loss: 0.6329 - val_binary_accuracy: 0.5787
Epoch 2560/10000
13/13 [=====] - 0s 16ms/step - loss: 0.6359 - binary_accuracy: 0.5763 - val_loss: 0.6328 - val_binary_accuracy: 0.5787
Epoch 2561/10000
13/13 [=====] - 0s 16ms/step - loss: 0.6334 - binary_accuracy: 0.6077 - val_loss: 0.6328 - val_binary_accuracy: 0.5787
Epoch 2562/10000
13/13 [=====] - 0s 16ms/step - loss: 0.6319 - binary_accuracy: 0.5860 - val_loss: 0.6327 - val_binary_accuracy: 0.5787
Epoch 2563/10000
13/13 [=====] - 0s 16ms/step - loss: 0.6351 - binary_accuracy: 0.5714 - val_loss: 0.6327 - val_binary_accuracy: 0.5787

```

*Figure 6.1 - Screenshot of terminal output*

It is of the opinion of the student that this program's greater potential could've been showcased. If the sample data was of greater volume, in order to allow for an algorithm to be formulated by the program. It is further demonstrated if the program was to use an alternative sample set which is the URL documented below

URL: [https://www.mediafire.com/file/wrcp31t4qcvauq3/JH\\_data\\_old.gz/file](https://www.mediafire.com/file/wrcp31t4qcvauq3/JH_data_old.gz/file)

The following sample set was used as a placeholder during the development of the program to serve for debugging purposes until the actual sample set intended for steganography detection was created. In theory, this program could be used for categorising movie reviews. Which was what the placeholder sample set was intended for.

Whilst this alternative sample set could have been used to demonstrate greater potential of this program. It would not have been in the spirit of this project. Which was once again “assign a probability value to a sample determining if the sample is suspected of concealing steganographic measures”.

Lastly, a last minute fault was found with the program. As shown in the “Code Snippet: Processing the data” which led to another test called “Code Snippet: Model Evaluation” having to be cancelled as relied upon “testing data” to be processed which it was not able to. Whilst it is regrettable that this feature isn't functional. It isn't core to the program. As this can be done manually in the Prediction section of the program and other functions also include evaluation features to measure the accuracy of the model. This can be seen in Figure 6.1.

To conclude, the main program is fully capable to realise overall goals set Chapters 1 and 3 of this document. The sample data is holding that overall goal back.

## Chapter 7: Conclusions

### 7.1 - A summary of what has been achieved in the project

All of the core systems and functionalities were created as envisioned by the student. Which allowed for a pipeline to the overall goal of trying to create a machine learning system that could differentiate between samples. In the opinion of the student, it would have been simply a matter of time and resources as all of the fundamental building blocks were in place.

The student feels a great sense of achievement despite spending almost only half a year on the topic of machine learning to be able to comprehend, design and implement a complete system. This project as a whole demonstrated the student's resolve and adaptability in unknown concepts and his skillset.

It is with a sense of regret that the results showcased in the prior chapter fell very under the target accuracy rating. The student does recognize that attempting to match industry standards was a very lofty ambition. It is of the opinion of the student that despite the results produced. It can be considered to be a very impressive first foray into the world of Machine Learning. Considering it encompasses an entire field of research. For the program to recognize half of its samples containing steganographic measures.

It should be stated that this project is a software development project however this project did have some light factors of an investigative project. As the student did wish for the machine learning program to iterate and create an algorithm to see if this program can replicate a task mainly done manually by humans.

## 7.2 - Reflections and lessons learned

A lot of very valuable lessons were learnt during the course of the project. Namely, mostly towards the psychological needs of the student. As they played a key factor in the success of this project. As documented within the Appendix section of this document. A major hiatus was taken during the months of January and February to better improve the mental wellbeing of the student.

Technical lessons were also learnt from this project. Namely revolving around time. Whilst in the background, advancements were made to TensorFlow which came to pass during the course of the project. Which resulted in one of the modules within the program being unable to function.

Overall, the project demonstrated a great lesson in awareness of both environmental factors and psychological wellbeing.

## 7.3 - Future work

A lot more functionalities and improvements would've been achieved, if the student was permitted, granted various resources and given a greater incentive to continue.

Namely, a greater sample set could have been created. The current sample set only contains 524 samples. It is expected that greater results could have been achieved if the program was given a greater bandwidth of data to analyse. This is initially why over three months were dedicated to the creation of sample data.

In addition, TensorFlow in-house evaluation tools would have been promptly restored. To allow for greater analysis. Albeit admittedly by the student. The extent of TensorFlow's tools were only used in a very basic form.

Lastly, the program could've been exported to an offline version to be used locally on a host machine. Rather than being restricted to Google's platform. However, this would have been a desired goal but not within the scope of the project. As this would've added unnecessary development overhead to ensure a consistent experience across different configurations and environments.

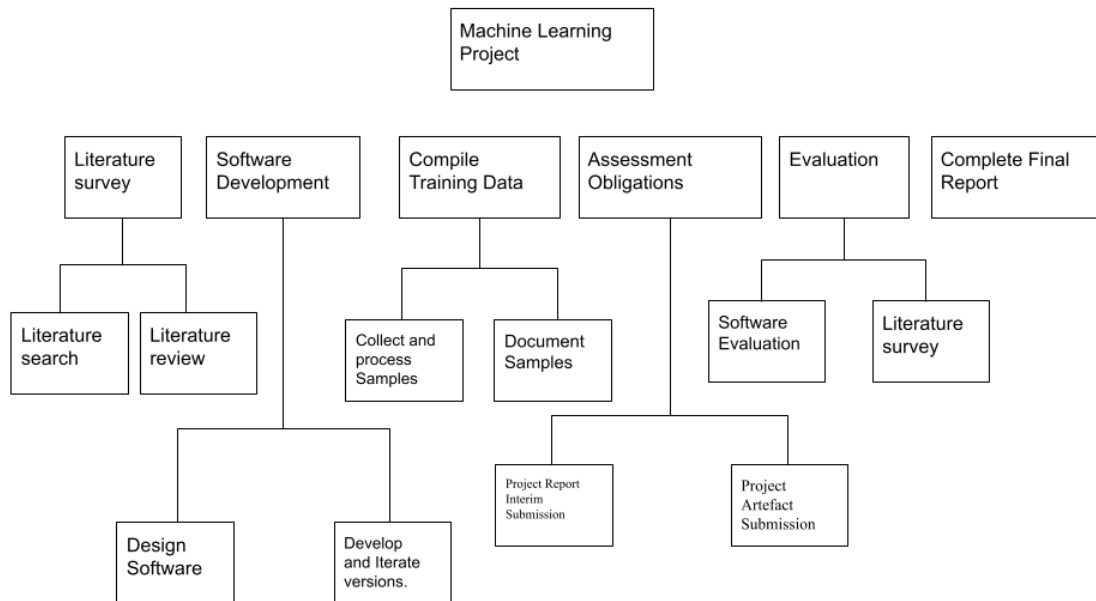
# Appendices

---

## Appendix 1: Project Management

### 1.1 The original project plan from the Proposal

Within this section of the document, It is visually illustrated how the tasks are structured and projected time for each specified task. As seen in Figure 66.a below, the project is broken down into a Work Breakdown Structure.



*Figure 66.a - Work Breakdown Structure*

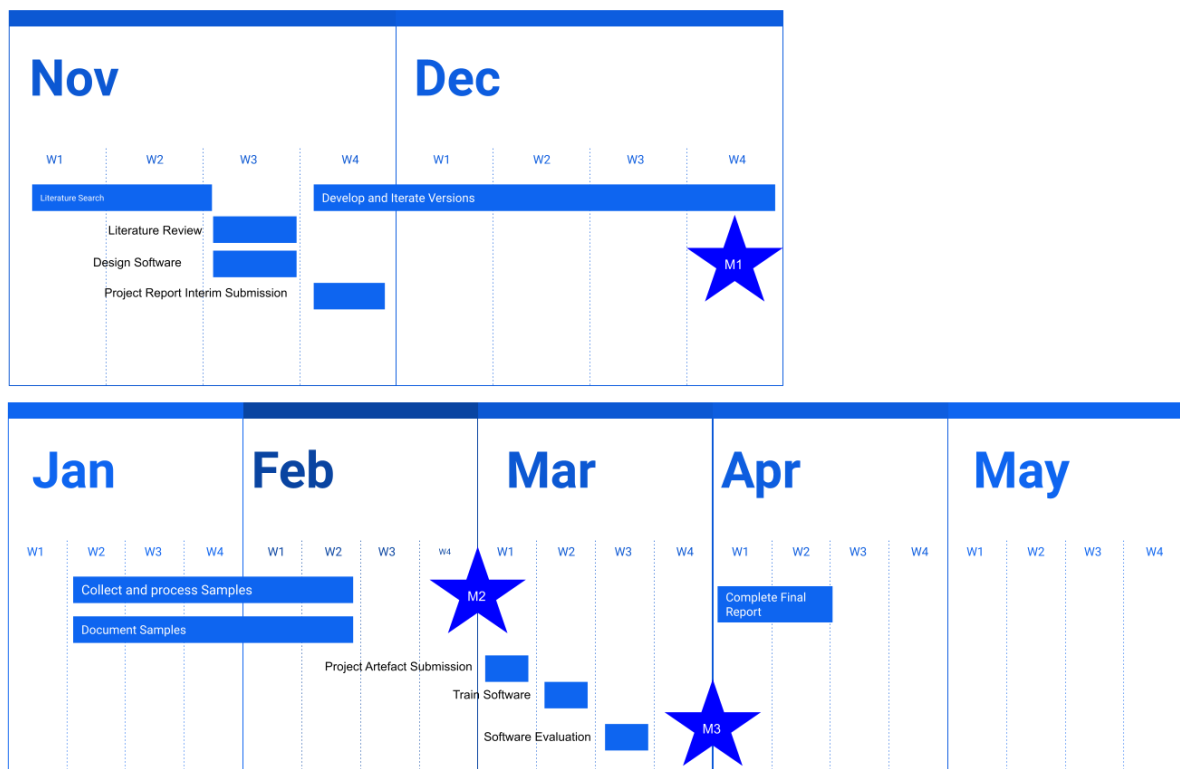
Tasks were created in accordance with the SMART Technique. Which is an acronym for Specific, Measurable, Achievable, Realistic and Timely. As all of the tasks listed follow these qualities. Which demonstrates the feasibility of this project. The projected duration of each task is listed below. Within Figure 6.b.

No.	Activities	Estimate Duration	Activity Description
1	Literature Search	2 Weeks	Conducting research and reading into various resources, compiling needed knowledge and skillsets through whatever means necessary.
2	Literature Review	1 Week	Reviewing said research, drawing conclusions and formulating theories or conclusions.
3	Design Software	1 Week	Planning the structure of the software, as well as its core functions.
4	Develop and Iterate Versions	5 Weeks	Development phase of the software. Where the software is created and improved upon in each instalment.
5	Collect and process Samples	5 Weeks	The creation of steganography samples utilizing various tools.
6	Document Samples	5 Weeks	Recording and assigning tags to various samples so that they can be independently reviewed.
7	Project Report Interim Submission	1 week	As part of the Assignment obligations, it is the first report after the project proposal.
8	Project Artefact Submission	1 week	As part of the Assignment obligations, it is the second report after the project proposal.
9	Train Software	1 Week	The feeding of samples into the software, so that it can further develop and establish links with the data provided.

10	Software Evaluation	1 Week	The testing of the software to see if the project goals were met.
11	Complete Final Report	2 week	The documentation summarizes the findings and results of this project.
	Total Duration	25 Weeks	

*Figure 66.b - Task Listing Table*

Lastly, the tasks are plotted out on a Gantt chart. Which illustrates when tasks will be completed during the course of the year. Some tasks can be completed in parallel with other tasks. In addition, milestones are listed in which a batch of tasks must be completed before the proposed deadline date. This ensures that the project is on track and schedule.



Milestone No.	
1	31st of December of 2021
2	28th of February of 2022
3	31st of March of 2022

*Figure 66.C - Milestone Table and Gantt Chart*

Below is an example of a table with a reference, which will automatically be pickup by the Word

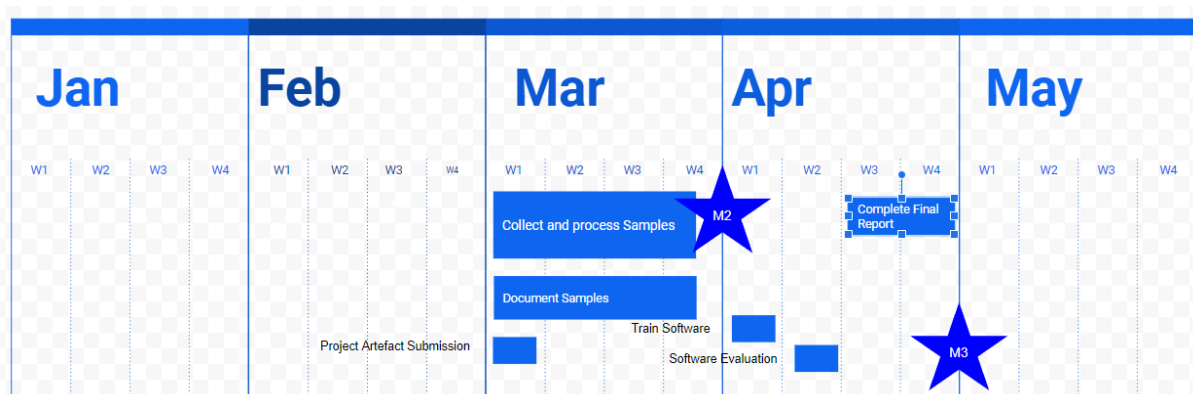
generator of the List of Tables

## 1.2 Review of your current progress

The project was met on time with the majority of all of the envisioned targets met. This despite the major hiatus during the months of January and February. As the student was cautious to allow for “buffer” months in the eventuality of a critical failure within the project. This allowed for a revised plan and course of action for the student to complete the project as there was plenty of time to allow for room for such plans.

## 1.3 Amendments to the original plan

As documented in the Project Artefact. This Gantt Chart displayed in Figure 66.D below was the revised plan and course of action taken due to the absence of work done during the months of January and February.



*Figure 66.D - Revised Gantt Chart*

## 1.4 Lessons learned in project management

The foresight of the student includes “buffer” weeks and a greater work ethic instilled by the student to ensure that the project was completed on time. The entire project is considered to be reinforcement of the student’s skill sets in adaptability and planning.

## Testing Screenshots

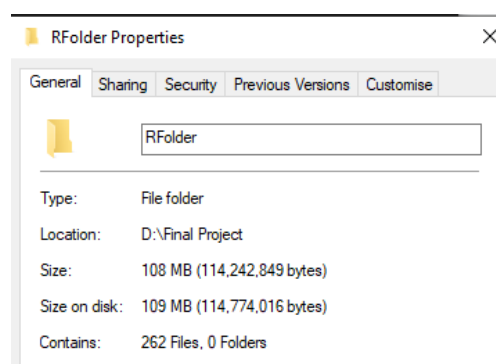


Figure XX.1 - Screenshot of Test Result #1

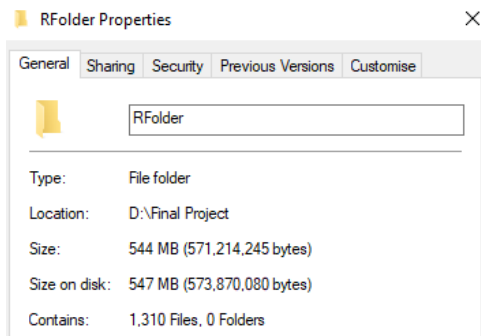


Figure XX.2 - Screenshot of Test Result #2

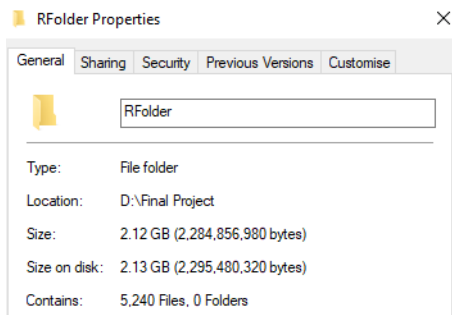


Figure XX.3 - Screenshot of Test Result #3

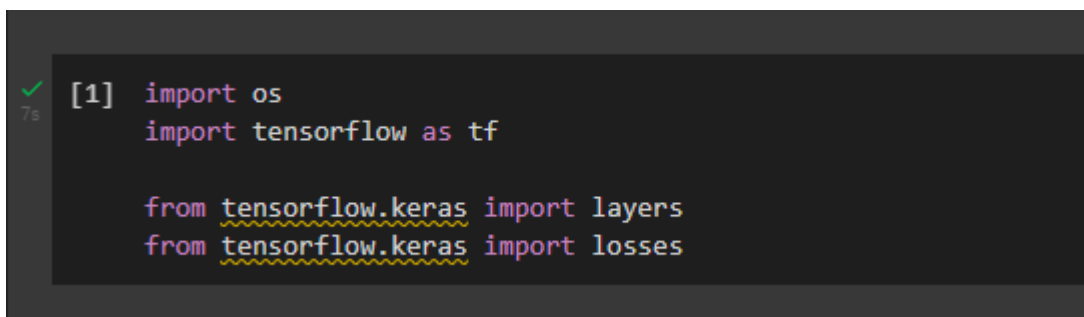


Figure XX.4 - Screenshot of Test Result #4



Figure XX.5 - Screenshot of Test Result #5



```

batch_size = 32
seed = 42

# ---- # ---- #

rtraining_set = tf.keras.utils.text_dataset_from_directory(
    'JH_data/project_data/training_data',
    batch_size=batch_size,
    validation_split=0.2,
    subset='training',
    seed=seed)

rvalidation_set = tf.keras.utils.text_dataset_from_directory(
    'JH_data/project_data/training_data',
    batch_size=batch_size,
    validation_split=0.2,
    subset='validation',
    seed=seed)

rtest_set = tf.keras.utils.text_dataset_from_directory(
    'JH_data/project_data/test_data',
    batch_size=batch_size)

Found 516 files belonging to 2 classes.
Using 413 files for training.
Found 516 files belonging to 2 classes.
Using 103 files for validation.
Found 0 files belonging to 0 classes.

ValueError                                Traceback (most recent call last)
<ipython-input-4-582271b4536e> in <module>()
     20 rtest_set = tf.keras.utils.text_dataset_from_directory(
     21     'JH_data/project_data/test_data',
--> 22     batch_size=batch_size)

/usr/local/lib/python3.7/dist-packages/keras/preprocessing/text_dataset.py in text_dataset_from_directory(directory, labels, label_mode, class_names, batch_size, max_length, shuffle, seed, validation_split, subset, follow_links)
    157     file_paths, labels, validation_split, subset)
    158     if not file_paths:
--> 159         raise ValueError(f'No text files found in directory {directory}. '
    160                         f'Allowed format: .txt')
    161

ValueError: No text files found in directory JH_data/project_data/test_data. Allowed format: .txt

```

Figure XX.6 - Screenshot of Test Result #6

```

[7] max_features = 10000
    sequence_length = 250

# ---- # ---- #

vectorize_layer = layers.TextVectorization(
    max_tokens=max_features,
    output_mode='int',
    output_sequence_length=sequence_length)

train_text = rtraining_set.map(lambda x, y: x)
vectorize_layer.adapt(train_text)

def vectorize_text(text, label):
    text = tf.expand_dims(text, -1)
    return vectorize_layer(text), label

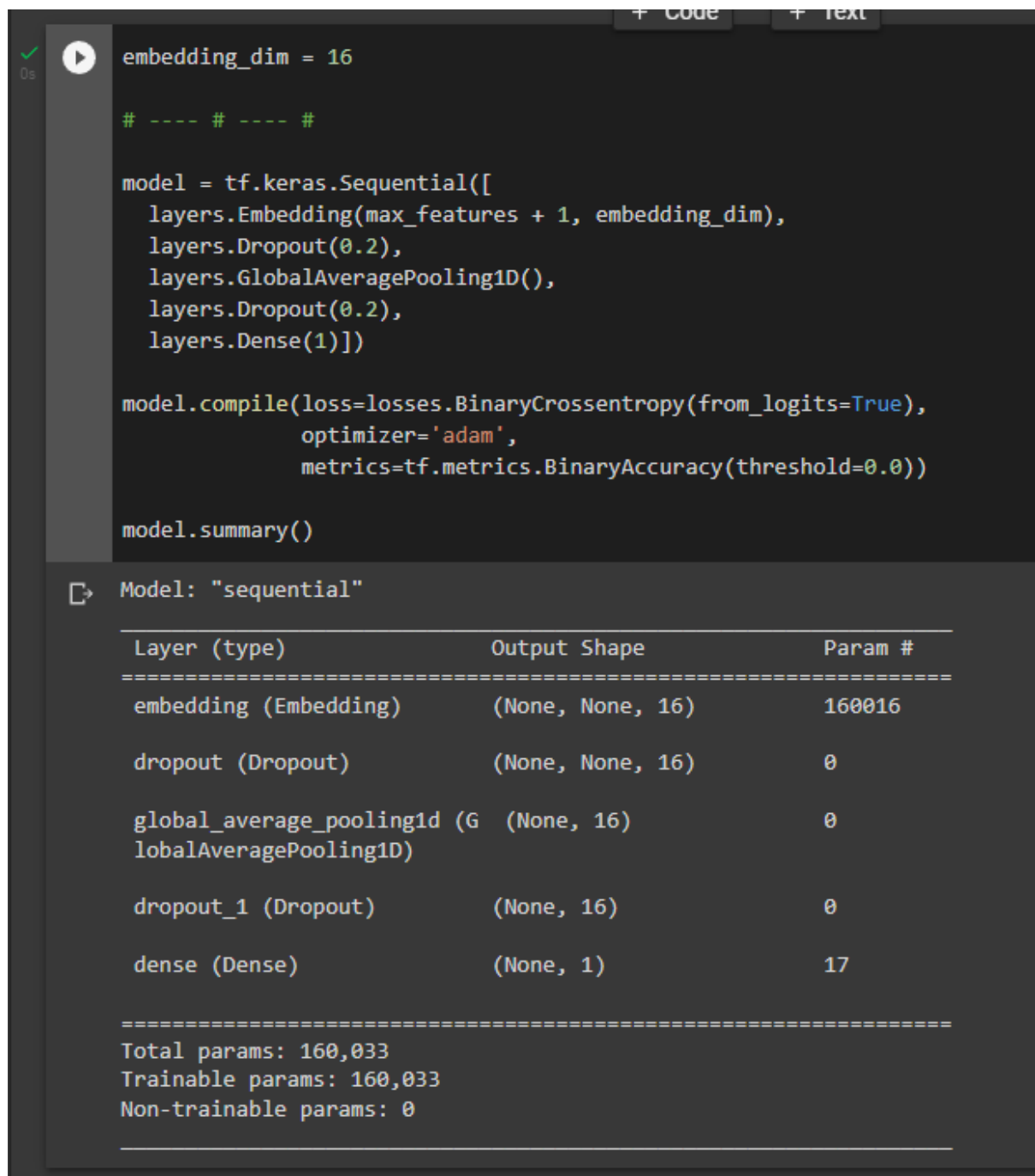
train_ds = rtraining_set.map(vectorize_text)
val_ds = rtraining_set.map(vectorize_text)
#test_ds = rtest_set.map(vectorize_text)

AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
#test_ds = test_ds.cache().prefetch(buffer_size=AUTOTUNE)

```

Figure XX.7 - Screenshot of Test Result #7



The screenshot shows a Jupyter Notebook interface with a code cell and its output. The code cell contains Python code for building and summarizing a Keras model. The output cell displays the model's summary as a table and then provides the total, trainable, and non-trainable parameter counts.

```
embedding_dim = 16

# ---- # ---- #

model = tf.keras.Sequential([
    layers.Embedding(max_features + 1, embedding_dim),
    layers.Dropout(0.2),
    layers.GlobalAveragePooling1D(),
    layers.Dropout(0.2),
    layers.Dense(1)])

model.compile(loss=losses.BinaryCrossentropy(from_logits=True),
              optimizer='adam',
              metrics=tf.metrics.BinaryAccuracy(threshold=0.0))

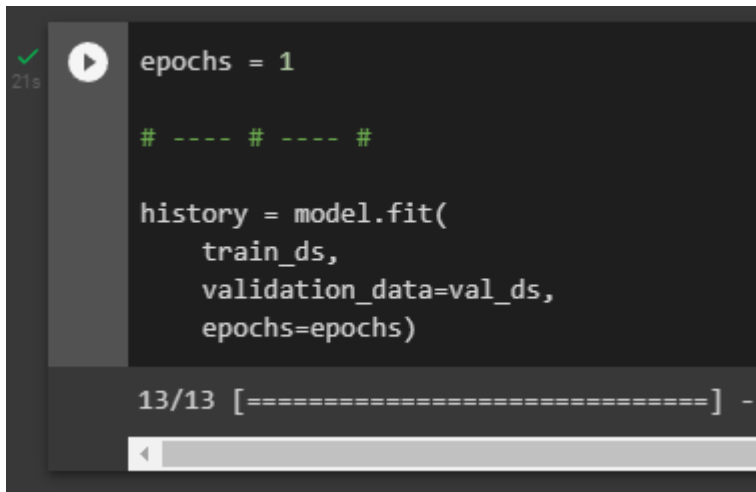
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, None, 16)	160016
dropout (Dropout)	(None, None, 16)	0
global_average_pooling1d (GlobalAveragePooling1D)	(None, 16)	0
dropout_1 (Dropout)	(None, 16)	0
dense (Dense)	(None, 1)	17

=====  
Total params: 160,033  
Trainable params: 160,033  
Non-trainable params: 0  
=====

Figure XX.8 - Screenshot of Test Result #8



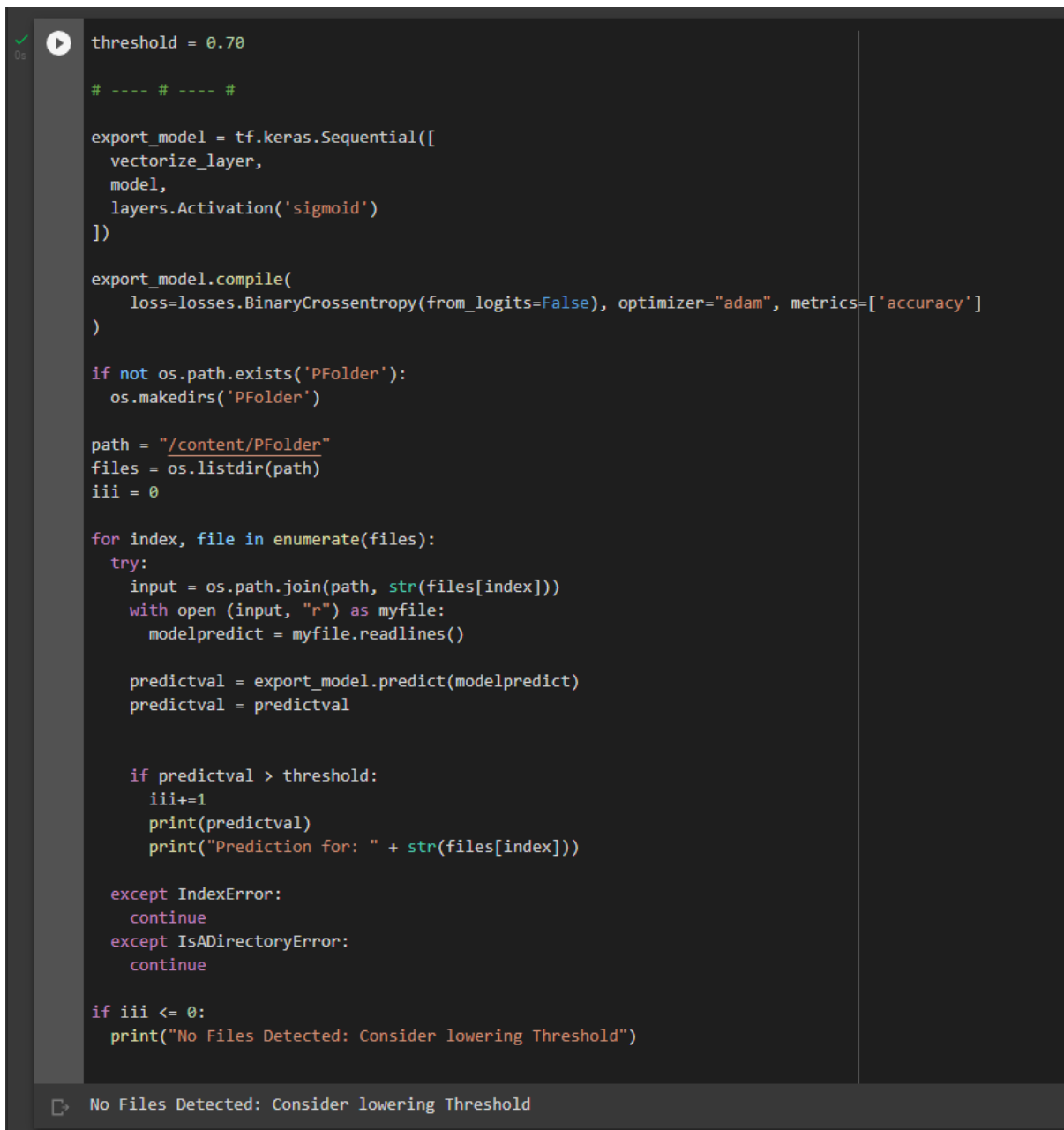
```
✓ 21s epochs = 1

# ---- # ---- #

history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs)

13/13 [=====] -
```

Figure XX.9 - Screenshot of Test Result #9



```
✓ 0s threshold = 0.70

# ---- # ---- #

export_model = tf.keras.Sequential([
    vectorize_layer,
    model,
    layers.Activation('sigmoid')
])

export_model.compile(
    loss=losses.BinaryCrossentropy(from_logits=False), optimizer="adam", metrics=['accuracy']
)

if not os.path.exists('PFolder'):
    os.makedirs('PFolder')

path = "/content/PFolder"
files = os.listdir(path)
iii = 0

for index, file in enumerate(files):
    try:
        input = os.path.join(path, str(files[index]))
        with open(input, "r") as myfile:
            modelpredict = myfile.readlines()

        predictval = export_model.predict(modelpredict)
        predictval = predictval

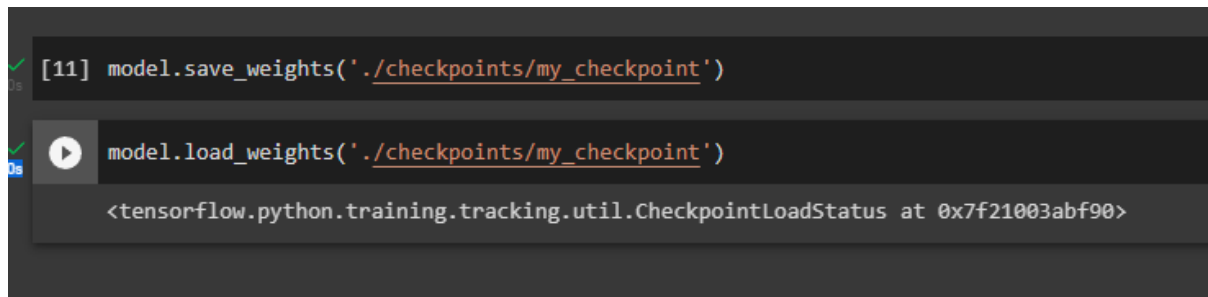
        if predictval > threshold:
            iii+=1
            print(predictval)
            print("Prediction for: " + str(files[index]))

    except IndexError:
        continue
    except IsADirectoryError:
        continue

if iii <= 0:
    print("No Files Detected: Consider lowering Threshold")

No Files Detected: Consider lowering Threshold
```

Figure XX.10 - Screenshot of Test Result #10



```
[11] model.save_weights('./checkpoints/my_checkpoint')

model.load_weights('./checkpoints/my_checkpoint')

<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7f21003abf90>
```

Figure XX.11 - Screenshot of Test Result #11

## References/Bibliography

- <sup>1</sup> Scrum.org (2021) The Scrum Framework. Available at: <https://www.scrum.org/resources/what-is-scrum> (Accessed: 20/10/2021)
- <sup>2</sup> Ed Burns, Nicole Laskowski and Linda Tucci (2021) What is artificial intelligence? Available at: <https://searchenterpriseai.techtarget.com/definition/AI-Artificial-Intelligence#:~:text=Artificial%20intelligence%20is%20the%20simulation,speech%20recognition%20and%20machine%20vision>. (Accessed at: 25/11/2021)
- <sup>3</sup> IBM (2021) What is machine learning? Available at: <https://www.ibm.com/uk-en/cloud/learn/machine-learning> (Accessed at: 25/11/2021)
- <sup>4</sup> Ben Dickson (2019) What is symbolic artificial intelligence? Available at: <https://bdtechtalks.com/2019/11/18/what-is-symbolic-artificial-intelligence/> (Accessed at: 25/11/2021)
- <sup>5</sup> Google (2021) Framing: Key ML Terminology. Available at: <https://developers.google.com/machine-learning/crash-course/framing/ml-terminology> (Accessed at: 25/11/2021)
- <sup>6</sup> AskDataScience (2018) What are the main branches of Machine Learning? Available at: <https://askdatascience.com/13/what-are-the-main-branches-of-machine-learning> (Accessed at: 25/11/2021)
- <sup>7</sup> Google (2021) Logistic Regression. Available at: <https://developers.google.com/machine-learning/crash-course/logistic-regression/video-lecture> (Accessed at: 25/11/2021)
- <sup>8</sup> MathsIsFun (2021) Equation of a Straight Line Available at: [https://www.mathsisfun.com/equation\\_of\\_line.html](https://www.mathsisfun.com/equation_of_line.html) (Accessed at: 25/11/2021)
- <sup>9</sup> Prasad Ostwal (2019) Multi-dimension plots in Python — From 3D to 6D. Available at: <https://medium.com/@prasadostwal/multi-dimension-plots-in-python-from-2d-to-6d-9a2bf7b8cc74> (Accessed at: 25/11/2021)
- <sup>10</sup> Google (2021) Classification. Available at: <https://developers.google.com/machine-learning/crash-course/classification/video-lecture> (Accessed at: 25/11/2021)
- <sup>11</sup> TensorFlow (2021) Introduction to Tensors. Available at: <https://www.tensorflow.org/guide/tensor> (Accessed at: 25/11/2021)
- <sup>12</sup> TensorFlow (2021) Introduction to Tensors. Available at: <https://www.tensorflow.org/guide/tensor> (Accessed at: 25/11/2021)
- <sup>13</sup> IBM (2021) What are neural networks? Available at: <https://www.ibm.com/cloud/learn/neural-networks> (Accessed at: 25/11/2021)
- <sup>14</sup> Tutorialspoint (2021) TensorFlow - Optimizers. Available at: [https://www.tutorialspoint.com/tensorflow/tensorflow\\_optimizers.htm](https://www.tutorialspoint.com/tensorflow/tensorflow_optimizers.htm) (Accessed at: 25/11/2021)
- <sup>15</sup> TensorFlow (2021) Hyperparameter Tuning with the HPparams Dashboard Available at: [https://www.tensorflow.org/tensorboard/hyperparameter\\_tuning\\_with\\_hparams](https://www.tensorflow.org/tensorboard/hyperparameter_tuning_with_hparams) (Accessed at: 25/11/2021)

- <sup>16</sup> TensorFlow (2021) Word embeddings. Available at: [https://www.tensorflow.org/text/guide/word\\_embeddings](https://www.tensorflow.org/text/guide/word_embeddings) (Accessed at: 25/11/2021)
- <sup>17</sup> IBM (2021) What are neural networks? Available at: <https://www.ibm.com/cloud/learn/neural-networks> (Accessed at: 25/11/2021)
- <sup>18</sup> IBM (2021) What are neural networks? Available at: <https://www.ibm.com/cloud/learn/neural-networks> (Accessed at: 25/11/2021)
- <sup>19</sup> wbStego(2004) Welcome to wbStego Steganography Tool! Available at: <http://wbstego.wbailer.com> (Accessed at: 25/11/2021)
- <sup>20</sup> BoiteAKlou (2018) Steganography Tutorial: Least Significant Bit (LSB). Available at: <https://www.boiteaklou.fr/Steganography-Least-Significant-Bit.html> (Accessed at: 25/11/2021)
- <sup>21</sup> IBM (2021) What is unsupervised learning? Available at: <https://www.ibm.com/cloud/learn/unsupervised-learning> (Accessed at: 25/11/2021)
- <sup>22</sup> Błażej Osiński and Konrad Budek (2018) What is reinforcement learning? The complete guide Available at: <https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/> (Accessed at: 25/11/2021)
- <sup>23</sup> Keras (2021) About Keras. Available at: <https://keras.io/about/> (Accessed at: 25/11/2021)
- <sup>24</sup> cybernecence ltd (2020) Free Steganography Software - QuickStego Available at: <http://quickcrypto.com/free-steganography-software.html> (Accessed at: 25/11/2021)
- <sup>25</sup> Samir Vaidya (2021) Using OpenStego. Available at: <https://www.openstego.com> (Accessed at: 25/11/2021)
- <sup>26</sup> Srinivas Ramakrishna (2021) Python for Non-Programmers. Available at: <https://wiki.python.org/moin/BeginnersGuide/NonProgrammers> (Accessed at: 25/11/2021)

## Endnotes

- <sup>1a</sup> TensorFlow (2022) API Documentation. Available At: [https://www.tensorflow.org/api\\_docs](https://www.tensorflow.org/api_docs) (Accessed at: 03/05/2022)
- <sup>2b</sup> Python (2022) os — Miscellaneous operating system interfaces. Available at: <https://docs.python.org/3/library/os.html> (Accessed at: 03/05/2022)
- <sup>3c</sup> Google (2022) Welcome To Colaboratory Available at: [https://colab.research.google.com/?utm\\_source=scs-index](https://colab.research.google.com/?utm_source=scs-index) (Accessed at: 03/05/2022)