Configuration Manual

MSc Research Project
M.Sc. In Cyber Security

Anaz Bin Ashraf
Student ID: 20230443

School of Computing
National College of Ireland

Supervisor: Mr. Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. ……………………ANAZ BIN ASHRAF…………………………………

Student ID: ………………………………20230443……………………………………….……

Programme: …………M.Sc. In Cybersecurity…………………… **Year:** ……….SEPT 2021-SEPT 2022………

Module: …………M.Sc. RESEARCH PROJECT…………………………………………………

Lecturer: ………………MICHAEL PANTRIDGE…………………………………………………

Submission Due Date: …………………15 AUGUST 2022………………………………………………

Project Title: …………………SIGNATURE FORGERY DETECTION……………………………

Word Count: …………835……………… **Page Count:** ………………13……………….…….………

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: …………………………ANAZ BIN ASHRAF…………………………………………

Date: …………………………15 AUGUST 2022……………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Anaz Bin Ashraf
Student ID: 20230443

1 Introduction

The configuration manual provides detailed information about the hardware, software and the tools used for training, testing and implementing the research project. It also provides details about how to run the code in the program and describes about the implementation process and how the results are obtained.

2 Environment Setup

This section provides the hardware and software tools used for running the project of running models to identify the accuracy and whether the signature is forged or not.

-Operating System: Windows 10
-OS Type: 64- bit Operating system, x64-based processor
-Processor: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz, 1800 Mhz, 4 Core(s), 8 Logical Processor(s)
-Memory: 12.0 GB
-Program language: Python 3.7
-Environment: Jupyter Notebook 6.4.12
-IDE: Anaconda 3

3 Installation of Tools

3.1 Jupyter Notebook 3:

It is a browser-based independent application. The major uses of the Jupyter notebook are as follows:
-It helps in creating documents.
-It helps in displaying live codes, equations and other snippets in codes.
-The formatted or edited document can share among other tools in the machine.
-It also has many other features including cleaning of the data, statistical modelling, visualization of data, etc.

3.2 Anaconda 3:

It is an open source python package that allows the data scientists and M.L enthusiasts to ensure that the same packages and dependencies are installed even if there is two different OS used for running. It helps to launch applications without having to use the help of command-line commands.

```
Jupyter Notebook (anaconda3)
[W 2022-08-14 18:57:31.139 LabApp] 'notebook_dir' has moved from NotebookApp to ServerApp. This config will be passed to
 ServerApp. Be sure to update your config before our next release.
[W 2022-08-14 18:57:31.140 LabApp] 'notebook_dir' has moved from NotebookApp to ServerApp. This config will be passed to
 ServerApp. Be sure to update your config before our next release.
[I 2022-08-14 18:57:31.161 LabApp] JupyterLab extension loaded from C:\Users\sudha\anaconda3\lib\site-packages\jupyterla
b
[I 2022-08-14 18:57:31.161 LabApp] JupyterLab application directory is C:\Users\sudha\anaconda3\share\jupyter\lab
[I 18:57:31.182 NotebookApp] Serving notebooks from local directory: C:\Users\sudha
[I 18:57:31.183 NotebookApp] Jupyter Notebook 6.4.5 is running at:
[I 18:57:31.183 NotebookApp] http://localhost:8888/?token=b70c3e7166c1a6511b577499eaa2f465c6f88c3eddfb9806
[I 18:57:31.183 NotebookApp]  or http://127.0.0.1:8888/?token=b70c3e7166c1a6511b577499eaa2f465c6f88c3eddfb9806
[I 18:57:31.183 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 18:57:31.289 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/sudha/AppData/Roaming/jupyter/runtime/nbserver-10468-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=b70c3e7166c1a6511b577499eaa2f465c6f88c3eddfb9806
     or http://127.0.0.1:8888/?token=b70c3e7166c1a6511b577499eaa2f465c6f88c3eddfb9806
[I 18:57:49.502 NotebookApp] Kernel started: 9f7ca600-8f7a-4022-9f94-b01fd3c4f39c, name: python3
[I 18:59:03.009 NotebookApp] Saving file at /Documents/Anaz Project/signature-vgg.ipynb
[I 18:59:06.305 NotebookApp] Saving file at /Documents/Anaz Project/signature-vgg.ipynb
[I 18:59:49.446 NotebookApp] Saving file at /Documents/Anaz Project/signature-vgg.ipynb
[I 19:01:14.895 NotebookApp] Saving file at /Documents/Anaz Project/signature-vgg.ipynb
```

We load the program using python from running the command in cmd and open the Jupyter Notebook by opening them using the terminal and from there we can browse the folder with which we have the code that we wanted to run. After selecting the code file just click on the "Run" button to run the code. It runs the model and we can get the output.

4 Code Execution for the algorithm model VGG16

```
In [1]: import numpy as np
        import cv2
        import os
        import shutil
        import random
        import pandas as pd
        import operator
        import seaborn as sns
        import matplotlib.pyplot as plt
        from sklearn.base import BaseEstimator, ClassifierMixin
        from sklearn.metrics import classification_report, confusion_matrix,ConfusionMatrixDisplay
        import tensorflow as tf
        from tensorflow.keras.models import Sequential
        from tensorflow.keras import optimizers
        from tensorflow.keras import backend as K
        from tensorflow.keras.utils import to_categorical
        from sklearn.preprocessing import LabelEncoder
        from tensorflow.keras.layers import *
        from tensorflow.keras import models
        from tensorflow.keras.models import Model
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

This line imports all the necessary libraries that are needed to start running the libraries. This libraries include tensorflow, numpy, pandas,matplotlib, sklearn, etc.

```
In [2]: if os.path.isdir("Training") or os.path.isdir("Validation"):
            print("Directory exist")
        else:
            os.mkdir("Training")
            os.mkdir("Validation")
            os.mkdir("Training/Fake")
            os.mkdir("Training/Real")
            os.mkdir("Validation/Fake")
            os.mkdir("Validation/Real")
```

```
In [3]: def dirHandler(path, name):
            for i in os.listdir(path):
                for j in os.listdir(path+i):
                    if "forg" in i:
                        shutil.copyfile(path+i+"/"+j, f"{name}/Fake/{j}")
                    else:
                        shutil.copyfile(path+i+"/"+j, f"{name}/Real/{j}")
```

```
In [4]: dirHandler("./archive/sign_data/train/", "Training")
        dirHandler("./archive/sign_data/test/", "Validation")
```

```
In [5]: train_dir = "./Training/"
        val_dir = "./Validation/"
        CATEGORIES = ["Fake","Real"]
```

The dataset for signature images were loaded into the model and categorized as real and fake images.

```
In [6]: def plotImages(x,y):
            plt.figure(figsize=[15,11])
            for i in range(16):
                plt.subplot(4,4,i+1)
                plt.imshow(x[i])
                plt.title(CATEGORIES[y[i]])
                plt.axis("off")
            plt.show()
```

```
In [7]: def split_train_test(data, img, labels):
            for i in data:
                img.append(i[0])
                labels.append(i[1])
```

This line shows code on plotting graphs and code for splitting training test for images from dataset.

In [8]:
```
def get_data(directory, list_dir):
    IMG_SIZE= 200
    for category in CATEGORIES:
        path = os.path.join(directory, category)
        class_num = CATEGORIES.index(category)
        for i, img in enumerate(os.listdir(path)):
            img_array = cv2.imread(os.path.join(path, img))
            new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
            new_array = cv2.cvtColor(new_array, cv2.COLOR_BGR2GRAY)
            list_dir.append([new_array, class_num])
```

In [9]:
```
train_data = []
val_data = []

get_data(train_dir,train_data)
get_data(val_dir,val_data)
```

In [10]:
```
len(train_data), len(val_data)
```

Out[10]:
```
(805, 264)
```

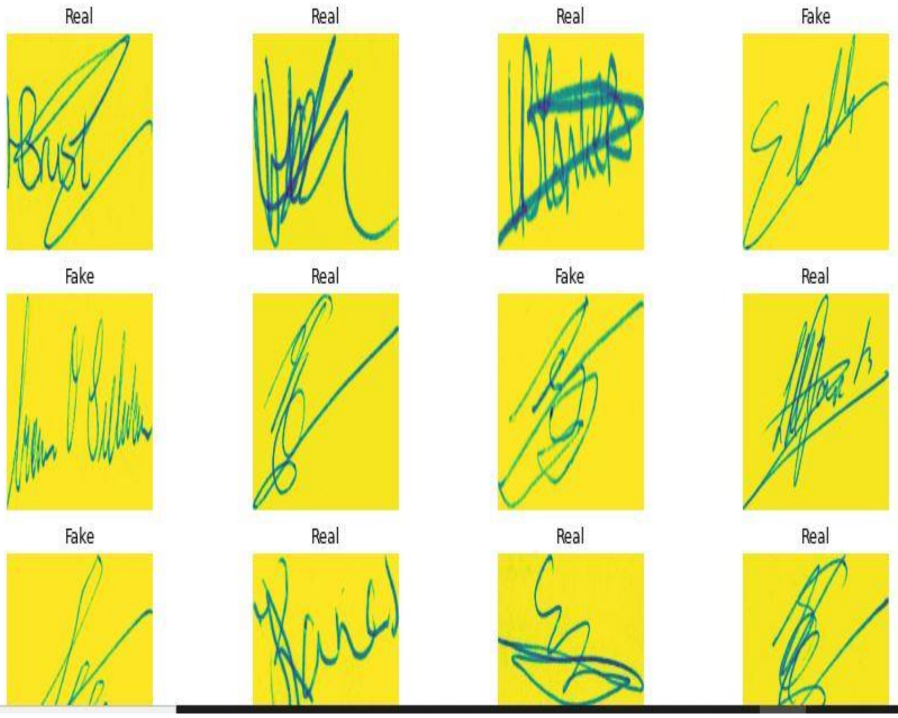
In [11]:
```
random.shuffle(train_data)
random.shuffle(val_data)
```

It provides the details of the data splitted into training and testing data respectively.

```
In [13]: len(X_train), len(X_val), len(y_train), len(y_val)

Out[13]: (805, 264, 805, 264)
```

```
In [14]: plotImages(X_train,y_train)
```



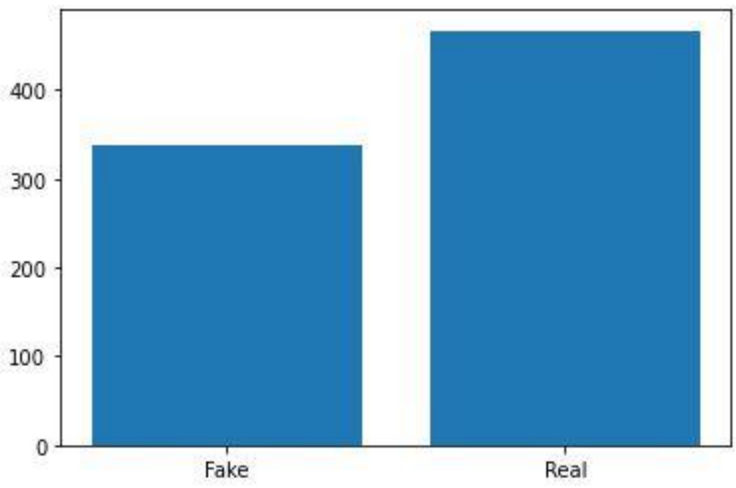
Displaying the sample images from the dataset which are imported into the program.

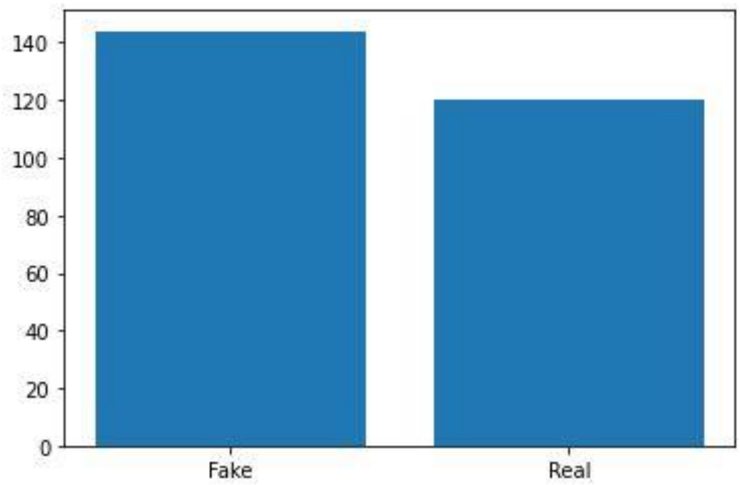
```
In [15]: def plot_bar_chart_diagram(path_data):
             dic = {}
             for file in os.listdir(path_data):
                 dem = 0
                 for x in os.listdir(path_data + "/" + file):
                     dem += 1
                 dic[file] = dem
             print(dic)
             barlist = plt.bar(list(range(len(dic))),
                               list(dic.values()),
                               tick_label=list(dic.keys()))
             plt.show()
```

```
In [16]: plot_bar_chart_diagram("Training")

         {'Fake': 338, 'Real': 467}
```

Plotting the bar graphs for the real and fake images.





The bar diagrams from the above code is given above.

In [19]:
```
model = Sequential()

model.add(Conv2D(64, (3, 3), activation='relu',padding = 'same', input_shape=(200,200,1)))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Conv2D(64, (3, 3),padding = 'same', activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(128, (3, 3),padding = 'same', activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Conv2D(128, (3, 3),padding = 'same', activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(256, (3, 3),padding = 'same', activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Conv2D(256, (3, 3),padding = 'same', activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))
```

This provides the model's definition in the program.

```
In [20]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 200, 200, 64)      640

 batch_normalization (BatchN  (None, 200, 200, 64)     256
 ormalization)

 dropout (Dropout)           (None, 200, 200, 64)      0

 conv2d_1 (Conv2D)           (None, 200, 200, 64)      36928

 batch_normalization_1 (Batc  (None, 200, 200, 64)     256
 hNormalization)

 dropout_1 (Dropout)         (None, 200, 200, 64)      0

 max_pooling2d (MaxPooling2D  (None, 100, 100, 64)     0
```

This provides details on the defined model which gives the details that are vital to the program such as defining the number of layers present in the model.

```
In [21]: model.compile(loss='categorical_crossentropy', optimizer="adam",metrics=['accuracy'])
```

```
In [22]: IMG_SIZE = 200
         X_train = np.array(X_train).reshape(-1,IMG_SIZE, IMG_SIZE,1)
         X_val= np.array(X_val).reshape(-1,IMG_SIZE, IMG_SIZE,1)


         y_train = np.array(y_train)
         y_val = np.array(y_val)
```

```
In [23]: datagen =  ImageDataGenerator(
               rescale=1./255,
               rotation_range=40,
               width_shift_range=0.2,
               height_shift_range=0.2,
               shear_range=0.2,
               zoom_range=0.2,
               horizontal_flip=True,
               fill_mode='nearest')
         datagen.fit(X_train)
```

```
In [25]: history = model.fit(datagen.flow(X_train,y_train, batch_size = 64) ,epochs = 30 , validation_data = datagen.flow(X_val, y_val))

Epoch 1/30
13/13 [==============================] - 925s 71s/step - loss: 0.9607 - accuracy: 0.6733 - val_loss: 1.4922 - val_accuracy: 0.4
545
Epoch 2/30
13/13 [==============================] - 978s 75s/step - loss: 0.5299 - accuracy: 0.8087 - val_loss: 5.5388 - val_accuracy: 0.5
455
Epoch 3/30
13/13 [==============================] - 977s 75s/step - loss: 0.3379 - accuracy: 0.8447 - val_loss: 14.2535 - val_accuracy: 0.
5455
Epoch 4/30
13/13 [==============================] - 1002s 77s/step - loss: 0.3078 - accuracy: 0.8770 - val_loss: 2.1562 - val_accuracy: 0.
5455
Epoch 5/30
13/13 [==============================] - 942s 73s/step - loss: 0.2651 - accuracy: 0.8857 - val_loss: 3.7609 - val_accuracy: 0.5
455
Epoch 6/30
13/13 [==============================] - 964s 74s/step - loss: 0.2518 - accuracy: 0.8969 - val_loss: 3.1568 - val_accuracy: 0.5
455
Epoch 7/30
13/13 [==============================] - 946s 73s/step - loss: 0.2287 - accuracy: 0.9056 - val_loss: 3.3089 - val_accuracy: 0.5
455
Epoch 8/30
13/13 [==============================] - 901s 69s/step - loss: 0.2006 - accuracy: 0.9193 - val_loss: 4.1642 - val_accuracy: 0.5
455
Epoch 9/30
13/13 [==============================] - 956s 73s/step - loss: 0.2520 - accuracy: 0.9056 - val_loss: 1.0527 - val_accuracy: 0.4
545
Epoch 10/30
13/13 [==============================] - 948s 76s/step - loss: 0.2021 - accuracy: 0.9143 - val_loss: 0.7123 - val_accuracy: 0.5
```

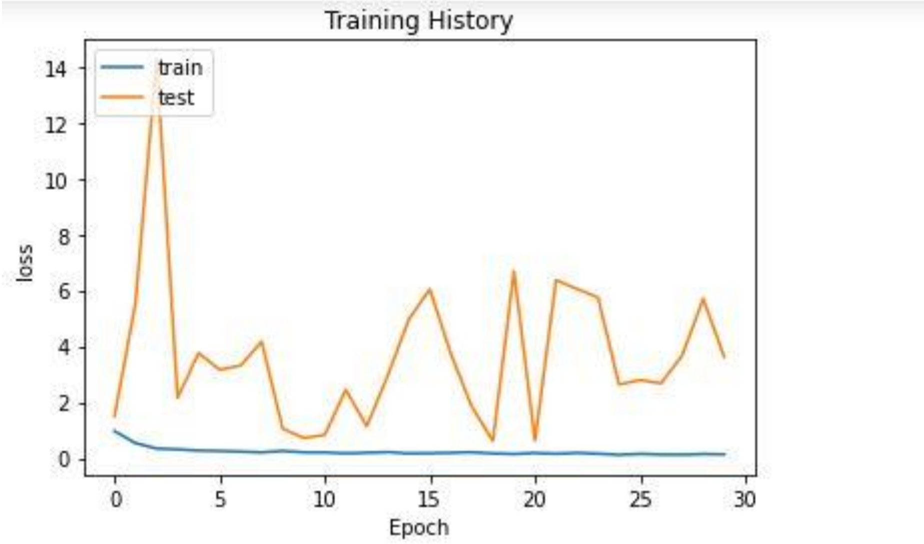
The model is now trained by running 30 successful epochs.

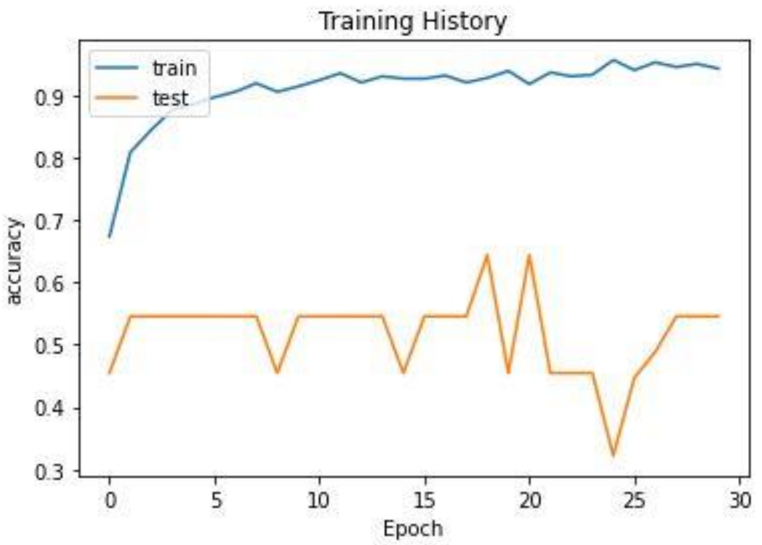
```
455
Epoch 30/30
13/13 [==============================] - 937s 73s/step - loss: 0.1290 - accuracy: 0.9429 - va
455
```

```
In [26]: keys=history.history.keys()

def show_train_history(hisData,train,test):
    plt.plot(hisData.history[train])
    plt.plot(hisData.history[test])
    plt.title('Training History')
    plt.ylabel(train)
    plt.xlabel('Epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

show_train_history(history, 'loss', 'val_loss')
show_train_history(history, 'accuracy', 'val_accuracy')
```





The training history is plotted from running the 30 epochs successfully

```
In [27]: score = model.evaluate(X_val,y_val)
         print("Loss: " + str(score[0]))
         print("Accuracy: " + str(score[1]*100) + "%")

         9/9 [==============================] - 61s 7s/step - loss: 6159.7271 - accuracy: 0.5455
         Loss: 6159.72705078125
         Accuracy: 54.54545617103577%

In [28]: predictions = model.predict(X_val)

         9/9 [==============================] - 60s 6s/step
```

The prediction report of the model is calculated.

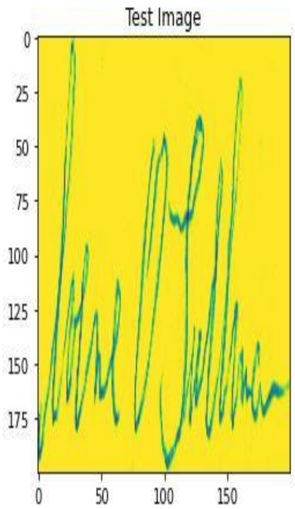
```
In [29]: y_pred = np.argmax(model.predict(X_val), axis=1)
         y_true = np.argmax(y_val, axis=1)
         display(ConfusionMatrixDisplay(confusion_matrix(y_true, y_pred), display_labels=CATEGORIES).plot())

9/9 [==============================] - 59s 6s/step

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2949e7a8160>
```

```
In [30]: img_array = cv2.imread("./Training/Fake/0102014_02.png")
         new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
         new_array = cv2.cvtColor(new_array, cv2.COLOR_BGR2GRAY)
```

```
In [31]: plt.imshow(new_array)
         plt.title("Test Image")
         plt.show()
```



Running a data from the sample and checking whether the signature image is fake or not.

References

[1] Jupyter.org. 2022. *Project Jupyter*. [online] Available at: <https://jupyter.org/> [Accessed 5 August 2022].

[2] Jupyter.org. 2022. *Project Jupyter*. [online] Available at: <https://jupyter.org/> [Accessed 5 August 2022].