

Configuration Manual

MSc

A

cademic Internship

MSc CYBERSECURITY

Oluwaseun Odunibosi

Student ID: X

18123970

School of Computing

National College of Ireland

Supervisor:

Mr V

IKAS SAHNI



**National College of Ireland**

**MSc Project Submission Sheet**

|  |  |
| --- | --- |
| **School of Computing**    **Student Name:** OLUWASEUN ODUNIBOSI    **Student ID:** X18123970 |  |
| **Programme:** Msc CYBERSECURITY    **Module:** ACADEMIC INTERNSHIP    **Lecturer:** VIKAS SAHNI  **Submission Due**  **Date:** 12/12/2019 | **Year:** 2019 |
|  |  |
| **Project Title:** CLASSIFICATION OF EMAIL HEADERS USING RANDOM FOREST ALGORITHM TO DETECT EMAIL SPOOFING    **Word Count:** 1951  **Page Count:** 19 | |

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:** | ……………………………………………………………………………………………………………… |
| **Date:** | ……………………………………………………………………………………………………………… |

**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

Forename Surname Student ID:

# SYSTEM CONFIGURATION

The system configuration of the system used is Windows 10, 64bits Operating System, ICORE 5 with 8BIT RAM and 500 GIG hard drive.

# INSTALLING REQUIRED SOFTWARE

The various software to be used in the research will be installed as shown below,

## DOWNLOAD PYTHON 3.8.0

Python version 3.8.0 must be downloaded on user system following the steps below. Download Python version 3.8.0 from this site.[[1]](#footnote-1)

* Select “Latest python 3 Release – Python 3.8.0”, Scroll down to Files and click on

“Windows X86-64 executables installer” as shown below

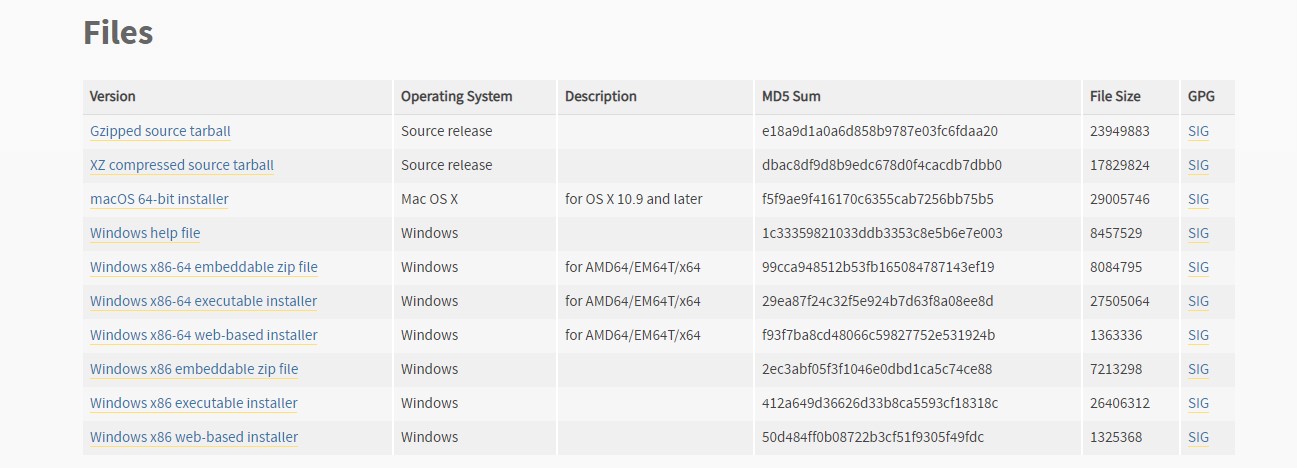


FIG. 1: shows download page of Python.

* Once downloaded, go to the location the file is saved and click on the installer.

* Select “Install launcher for all users (recommended)” and Select ‘Install Now’ which include installation of IDLE, pip and documentation and creates shortcuts and file associations as shown in the Fig. 2.

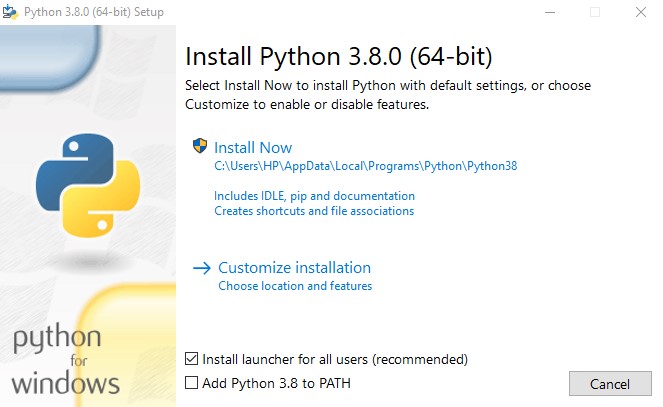


FIG. 2: Shows the installation process

* If “Add python 3.8.0 is selected”, then then the installation directory will be added to your PATH.
* You will see a screen showing the installation process, when installation is finished, a screen showing as shown below in Fig. 3 will appear on the screen.

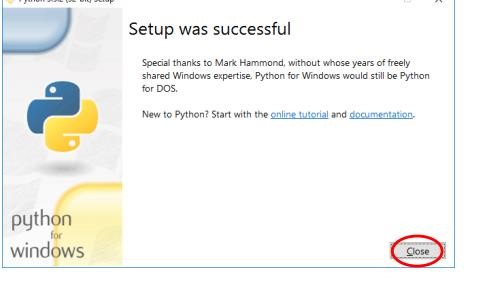


FIG.3: Installation Successful

**Running Python Application**

* Click close and navigate to the windows command prompt.
* Once the command prompt is opened, execute this command to check if python is working and version installed as shown below.

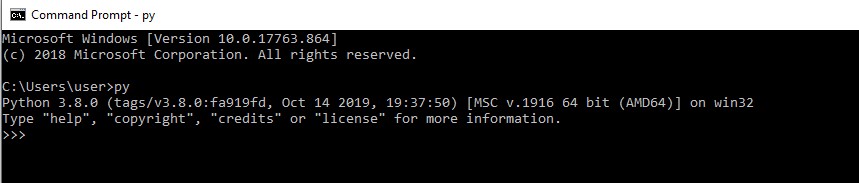


FIG. 4: Shows Python application is working and the Version

User may encounter an alert during installation that instruct user to “Remove the MAX\_PATH Limitation” before installation can continue. This is because Windows has path length of 260 characters and any paths longer than 260 would result in error. If encountered; In latest version of Windows, this limitation, user can expand this limitation using like 32,000 characters. Activate the “enable Win32 long paths” group policy. This can also be resolved by setting the registry key value

(HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem@LongPaths

Enabled) to 1.

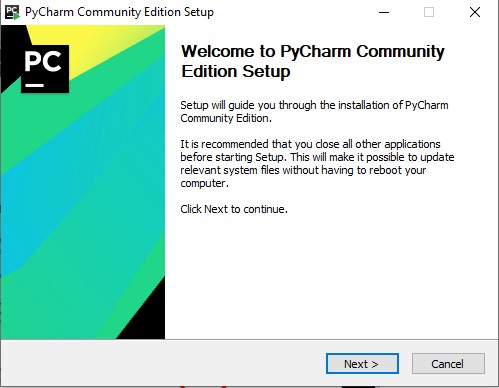
## DOWNLOAD PYCHARM VERSION 2019.3

* Download the PyCharm Version 2019.3 from this site[[2]](#footnote-2)
* Click Download Now.
* Click on the Download Button under the Community Edition. Only free trial of Professional edition will be available if you select that except you want to purchase.



FIG. 5: PyCharm Installation

* Double click on the Downloaded PyCharm, a pop up will appear on the screen as shown below. Click next



* Choose the location where installation will be saved, I recommend choosing the suggested location. Click next when done.

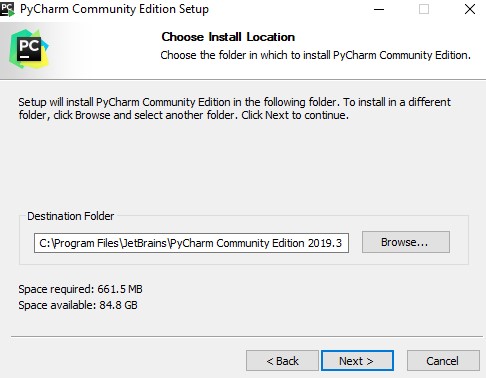


FIG. 7: Location to Save

* Select the first and the third options as shown in the diagram below and click next. The first option will create a desktop shortcut where user can easily navigate to PyCharm. The third option will create association so that when Python file is opened, it will automatically open in PyCharm.

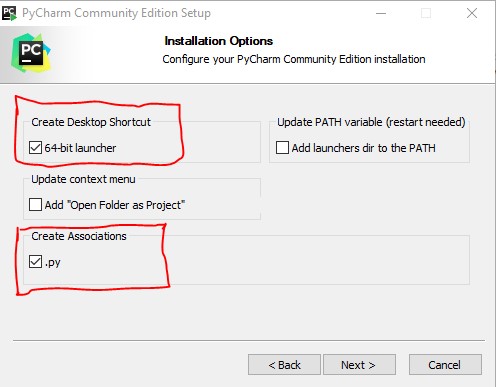


FIG. 8: Installation process

* Select the start menu folder and click Install, I recommend leaving the default on the screen “JetBrains”

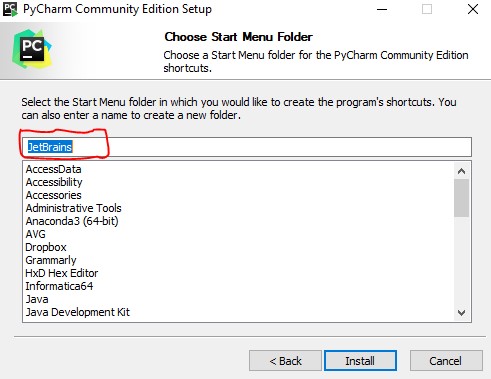


FIG. 9: Selecting Start Menu Folder

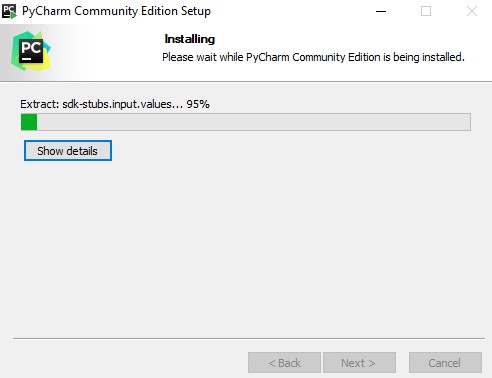


FIG. 9: Shows Installation Process

* Once installation is done, a message that PyCharm is installed is shown. Click

“Finish”. If you want to run the PyCharm application immediately, click “Run

PyCharm Community Edition” first before you click “Finish”

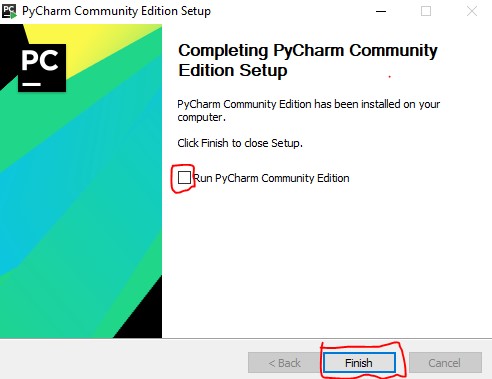


FIG. 10: PyCharm Installation finished.

**Running PyCharm Application**

* User will get a message box asking about import settings when you run the PyCharm application for the first time. Select “Do not Import Settings” and click Ok.

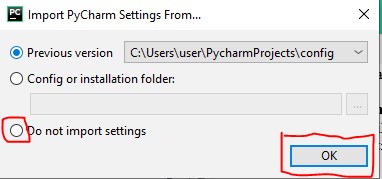


FIG. 11: PyCharm Import Setting

* Some initial configuration will have to be performed when running the PyCharm for the first time as shown below. Select the Customization menu that suit you (Dracula or Light) and click next featured plugins. I recommend skipping remaining setup and Set all as Default shown on the bottom left of the screen.

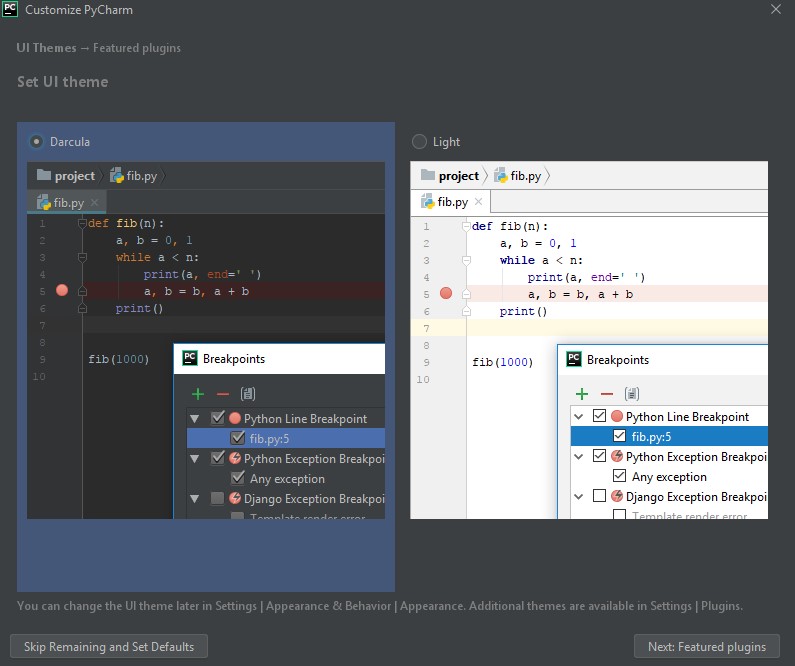


FIG. 12: Configuration Setting

* A intro screen will appear on the screen for PyCharm. Click on “Create New Project”

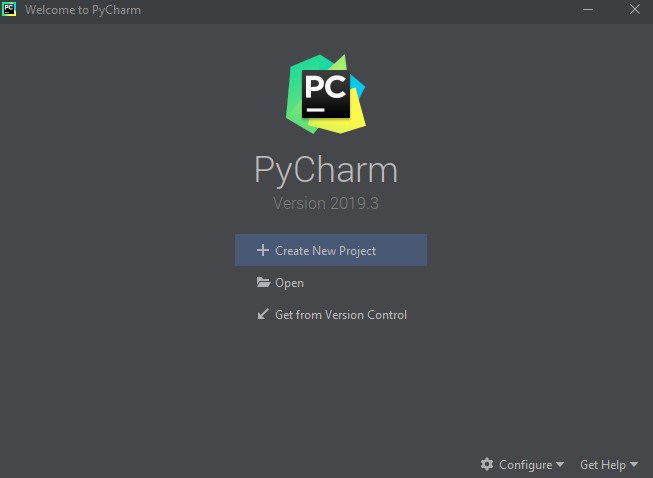


FIG. 13: New Project Screen

* Select location where the project created will be saved, change the name of the “Untitled” to “MyProject” and on the “Project Interpreter: New Virtualenv environment” dropdown and select “Inherit global site-packages” and “Make available to all projects”. PyCharm would have found the Python interpreter installed earlier in section 1 and this must appear in the base interpreter. If nothing is showing in the base interpreter field, then it must be resolved before you click “Create”.

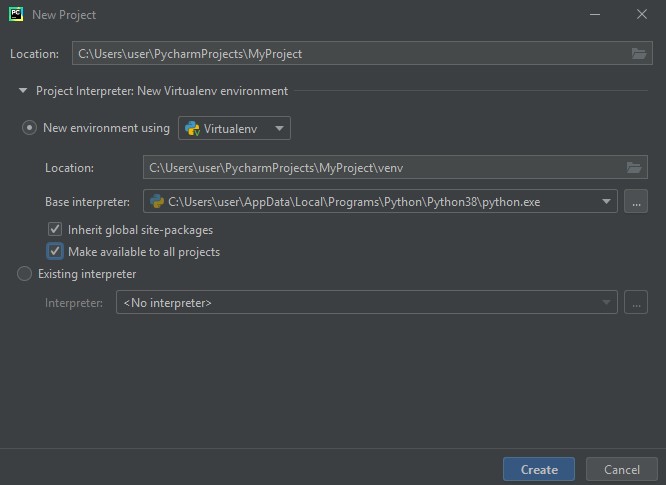


FIG. 14: New Project Creation.

* The PyCharm environment is shown once the “Create” button is selected, close the tips menu so you can have access to the PyCharm main environment.

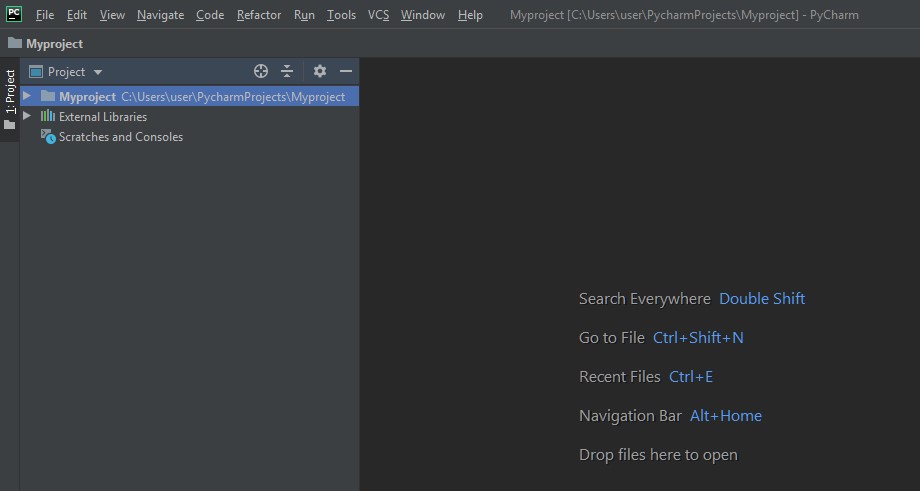


FIG. 15: PyCharm Environment

* To create and run the Email\_Spoofing\_Classifier\_final of this research, navigate “File” menu and select “New” and select “Python File”.

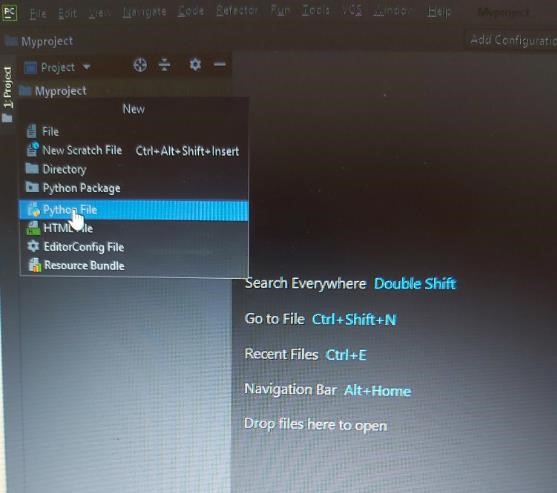


FIG. 16: Creating a program

* A menu will appear where you will need to type in the python file name then click “Python file” as seen below

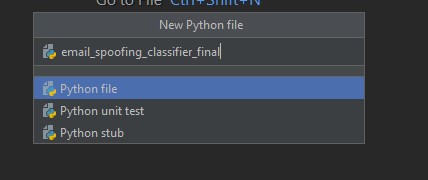


FIG. 17: Python File name Creation

* This brings the user to the new created Python file named “Email\_Spoofing\_Classifier\_final”. This menu is where user will start building the code for the research.

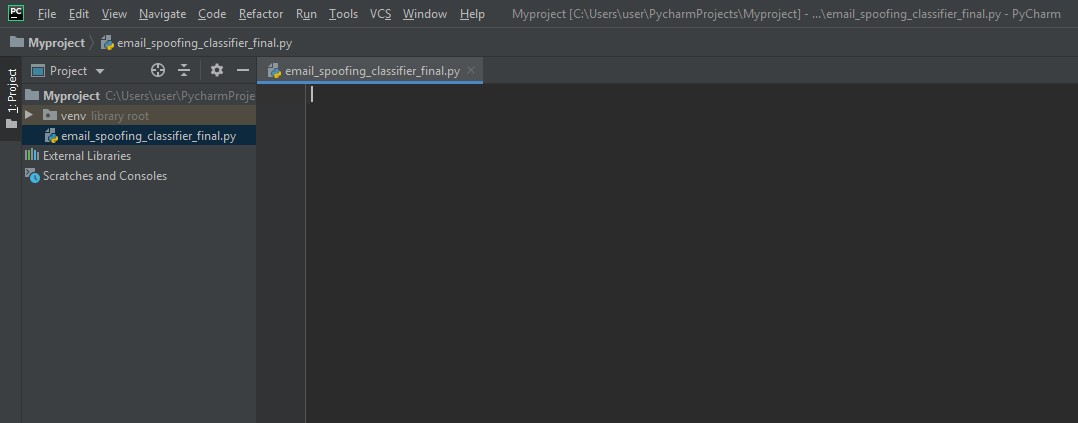


FIG. 18: The Python Menu for coding

# CODE FOR THE RESEARCH WORK

The code below will be pasted on the Email\_Spoofing\_Classifier\_final python menu. The code must be as specified below except the FROM\_EMAIL and FROM\_PWD field which must be changed to user own Gmail username and Password. Note that some of this code are general convention of using the module imported so they are not change. More information can be found in the code reference or in this site.[[3]](#footnote-3)

## Email\_Spoofing\_Classifier\_final python

import smtplib import time import imaplib import email import csv

from datetime import datetime import randforest as rf

import argparse

# GMail login

# -------------------------------------------------

# Tittle:How to Read Email From Gmail Using Python, 2017. . Code Handbook.

# Author: JAY

# Date: 2017

#Availability:https://codehandbook.org/how-to-read-email-from-gmail-using- python/#comment-4217804161

# ------------------------------------------------

SMTP\_SERVER = "imap.gmail.com"

SMTP\_PORT = 993

FROM\_EMAIL = "cainersteph@gmail.com"

FROM\_PWD = "Stephen-1901"

# ------------------------------------------------- #

# Save message to CSV file

#

# ------------------------------------------------ def save\_output(filename, row): with open(filename, 'a') as f:

writer = csv.writer(f)

writer.writerow(row)

# ------------------------------------------------- #

# Read mails from GMAIL Inbox

#

# ------------------------------------------------ def read\_mails(filename, email\_counts):

# Handle connection exception try:

# connect to smtp server

mail = imaplib.IMAP4\_SSL(SMTP\_SERVER)

# Login

mail.login(FROM\_EMAIL, FROM\_PWD)

mail.select('inbox')

# Search for emails

type, data = mail.search(None, 'ALL')

mail\_ids = data[0]

# Process emails

# -------------------------------------------------

# Tittle:How to Read Email From Gmail Using Python, 2017. . Code Handbook.

# Author: JAY

# Date: 2017

#Availability:https://codehandbook.org/how-to-read-email-from-gmail-using- python/#comment-4217804161

# ------------------------------------------------

id\_list = mail\_ids.split()

first\_email\_id = int(id\_list[0])

latest\_email\_id = int(id\_list[-1])

# Fetch emails from latest to old for i in range(latest\_email\_id, first\_email\_id, -1):

typ, data = mail.fetch(str(i), '(RFC822)')

if (email\_counts > 0): for response\_part in data: if isinstance(response\_part, tuple): # Decode message from response string

msg = email.message\_from\_string(response\_part[1].decode('utf8').strip())

# Prepare output to CSV file email\_subject = msg['subject'] email\_from = msg['from'] email\_to = msg['to'] msg\_id = msg['message-id']

date = msg['date']

# Message

csv\_content = [email\_from, email\_to, email\_subject, msg\_id, date]

print("Reading and storing messages in " + filename)

# Save message to CSV file

save\_output(filename, csv\_content)

# Progress report

print()

print("Successfully written to " + filename + ": " + str(csv\_content))

else:

break

email\_counts = email\_counts - 1;

except Exception as e:

print(str(e))

return filename

def main():

# initiate the parser

parser = argparse.ArgumentParser()

parser.add\_argument("-e", "--emails", help="Input the number of emails to read") parser.add\_argument("-d", "--dump", help="Provide an exist email header dump file")

# Execute command line arguments args = parser.parse\_args()

if (args.emails): email\_counts = int(args.emails) else:

# Default email counts

email\_counts = 1000

if (args.dump):

# Read emails for local file filename = args.dump

print("Email header file to classify: {}".format(filename)) else:

# Read emails from gmail

# Create csv file

filename = "csvfile\_{}.csv".format(datetime.now().strftime("%H-%M-%S"))

# CSV Fields

fields = ['From', 'To', 'Subject', 'Message-ID', 'Date'] save\_output(filename, fields)

filename = read\_mails(filename, email\_counts)

# New line

print()

# Prompt the user to decide on data classification

confirm = input("Do you want to classify the data as well? (Yes or No) "); confirm = confirm.lower()

if ("yes".find(confirm) != -1): # classifly data if yes rf.random\_forest\_algo(filename)

if \_\_name\_\_ == "\_\_main\_\_": main()

## Creating the Random Forest Algorithm

Following the procedure used in creating Email\_Spoofing\_Classifier\_final above.

* Repeat procedures carried out in Fig. 16
* Follow procedure on Fig. 17 but change the name of the new python file to be created to “randomforest” as shown below

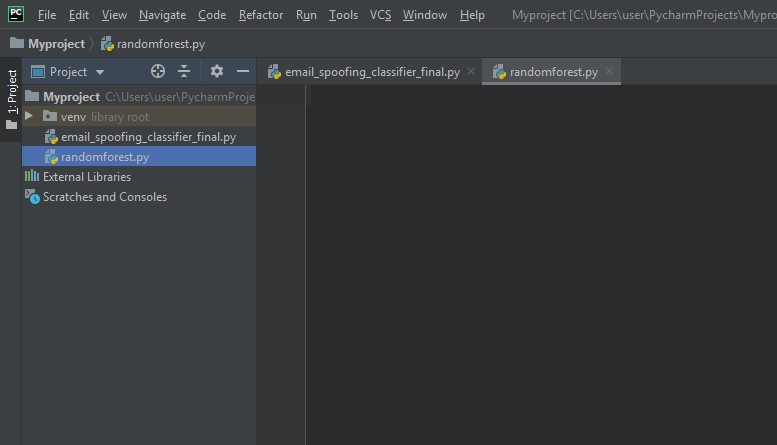


FIG. 19: Creation of randomforest python file

* Put in this code below

|  |  |
| --- | --- |
| • | import warnings  from sklearn.exceptions import DataConversionWarning import pandas as pd  from sklearn.ensemble import RandomForestClassifier from sklearn.datasets import make\_classification from sklearn.model\_selection import train\_test\_split from sklearn import metrics  from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score      #turn off warning error  warnings.simplefilter("ignore")  def random\_forest\_algo(filename): df = pd.read\_csv(filename) df.head()  with pd.option\_context('display.max\_rows', 1000):  print (df)  #Spliting data to test and training set  X, y = make\_classification(n\_samples=1000, n\_features=2, n\_informative=2, n\_redundant=0, random\_state=5)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3) X\_train X\_test  classifier = RandomForestClassifier(n\_estimators=20, random\_state=5) classifier classifier.fit(X, y)  print(classifier.feature\_importances\_)    y\_prediction = classifier.predict(X\_test) y\_prediction |
| # Accuracy  print(metrics.accuracy\_score(y\_test, y\_prediction)) classifier.apply(X)  classifier.get\_params(deep=True) classifier.predict\_log\_proba(X) classifier.predict\_proba(X) classifier.score(X, y)    # Confusion Matrix  print(confusion\_matrix(y\_test,y\_prediction)) print(classification\_report(y\_test,y\_prediction)) print(accuracy\_score(y\_test, y\_prediction)) | |

1. **SECURE LESS APPLICATION ON GMAIL ACCOUNT.**

User must be log in to Gmail account before carrying out the following process.

* + Login to Gmail account from the browser.
  + Sign in with Username and Password.
  + Once the inbox of Gmail is opened, navigate to “Google Account” on the upper right of the gmail menu and click “Manage your Google Account”
  + A new screen is opened showing Personal info, Data and personalization etc. Select “Security” and scroll down to “Less secure app access”. Click on “Turn on” and go back to inbox. It is important to turn on less secure application for our code to capture user received mail.

1. **RUN THE Email\_Spoofing\_Classifier\_final CODE.**

The final stage is to go back to the opened Email\_Spoofing\_Classifier\_final code in PyCharm

* + Highlight the “Email\_Spoofing\_Classifier\_final” and right click on it and on the dropdown menu, click “Run Email\_Spoofing\_Classifier\_final”.

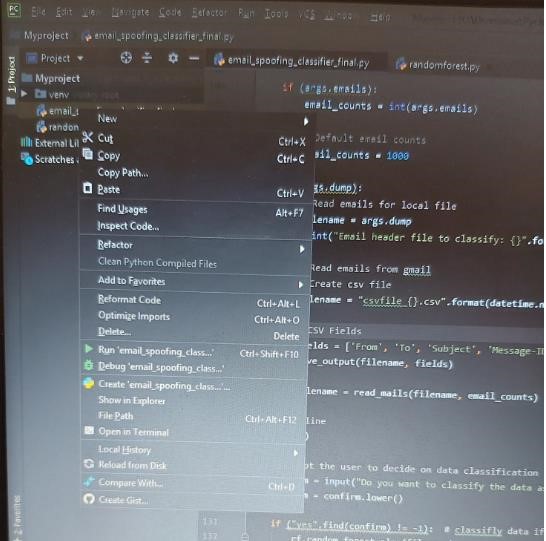


FIG. 22: Executing the code

* + The result of executing the Email\_Spoofing\_Classifier\_final code will be shown at the bottom on the “Run Terminal”.

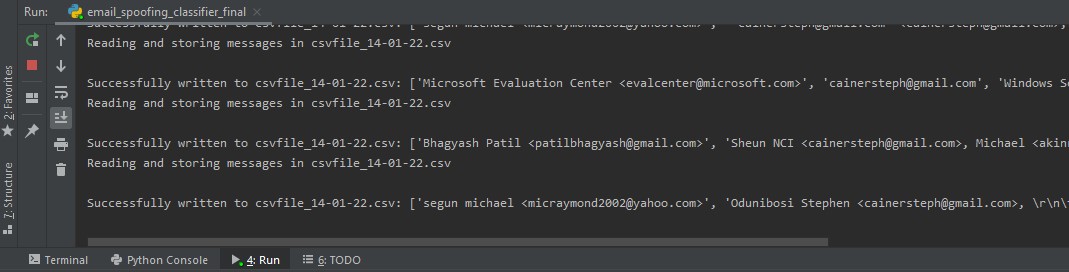


FIG. 23: Results Of code Execution

* + In the course of execution of the code User is prompted on the Run terminal if classification of mails should be carried out. User should write Yes and click enter.

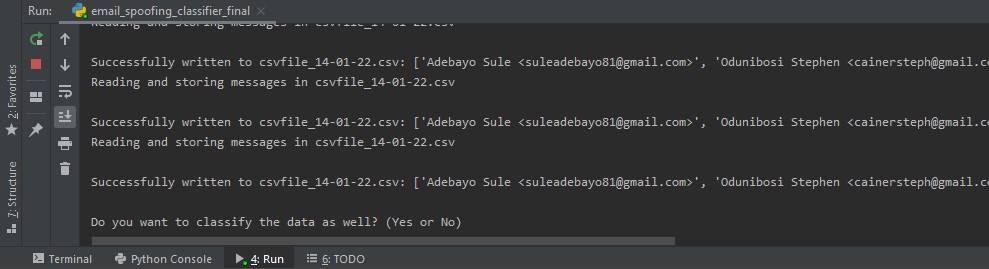


FIG. 24: Confirmation of Classification

* + The extracted email saved in CSV can also be viewed as below.

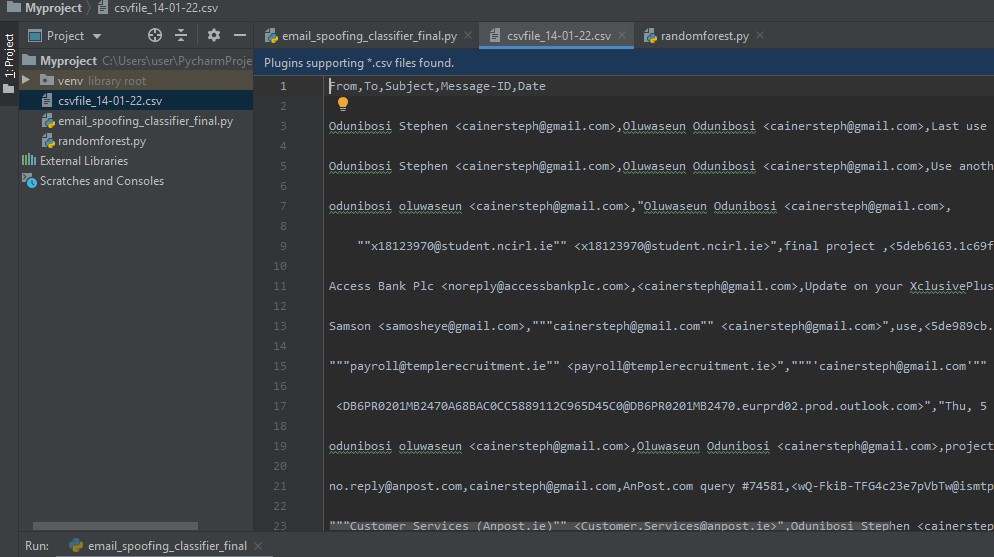


FIG. 25: CSV format email

1. <https://www.python.org/downloads/windows/> [↑](#footnote-ref-1)
2. <https://www.jetbrains.com/pycharm/> [↑](#footnote-ref-2)
3. <https://codehandbook.org/how-to-read-email-from-gmail-using-python/#comment-4217804161> [↑](#footnote-ref-3)