

Report on Casey's Metadata analyser

*A report covering how to create a functioning python tool which is designed to
analyse and extract metadata.*

Casey Donaldson

CMP320: Advance Hacking

2023/24

Note that Information contained in this document is for educational purposes.

Abstract

This report is about the development of a python scripting project. Focusing on metadata importance within files and the vulnerabilities that it causes. This particular project works with Jpeg metadata which contains GPS information, make and model of the device used for the picture and more. The data held within these images can be used to help law enforcement and developers. However, this type of data creates risks as it could be used to reveal their location and their hardware used. Although some websites remove data not all platforms do, for example Google Chat.

This report will also discuss the application and its ability to generate maps based on GPS data if present and the process of creating a CSV file with all extracted metadata. The study will also explain how to use the application and its many arguments which can be used. Additionally, the report will explain the extensive testing that was conducted on the application which will reveal the robustness of the program. The other area of this study will examine the source code and external libraries, and the practices used to ensure the program is expandable and reuseable. Furthermore, the report will also explain how the program generates the results. This will provide a comprehensive understanding of the program for the readers.

The last sections of the report will evaluate the program including coverage, useability, and performance. Once the evaluation is completed, the study will analyse the advantages and limitations of the application in an attempt to create a non-bias result section. The countermeasures section will inform the reader on how to avoid personal metadata being exposed and used. Furthermore, additional work which can be conducted will be discussed, highlighting areas which could be implemented for example modifying the metadata and AI driven image location identifier.

Contents

1	Introduction.....	1
1.1	Background	1
1.2	Aim	3
1.3	Ethical use of script.....	4
2	Procedure	5
2.1	Overview of report	5
2.2	Application background	5
2.3	Application in use	6
2.3.1	Help Command “-h –help”	6
2.3.2	Test on image file with no other flags.....	6
2.3.3	Test on image without metadata and other flags.	9
2.3.4	Test on image with metadata and output flag.	9
2.3.5	Test deletion on image.....	10
2.3.6	Test script on folder of files.	12
2.3.7	Additional file types	14
2.4	Review of code and practices	14
2.4.1	Imports	14
2.4.2	Banner	15
2.4.3	Argument parser	16
2.4.4	Main section	17
2.4.5	Get file extension function.	18
2.4.6	Image metadata extractor function.....	19
2.4.7	Delete and modify metadata flags.	20
2.4.8	Extracting metadata.....	20
2.4.9	Displaying metadata.	21
2.4.10	Dictionary to CSV function.....	23
2.4.11	Image locator function.	23
2.5	Evaluation	26
2.5.1	Coverage.....	26
2.5.2	Performance	27
2.5.3	Useability.....	27

3	Discussion.....	28
3.1	General Discussion	28
3.2	Limitations and Advantages	28
3.2.1	Limitations	28
3.2.2	Advantages	28
3.3	Countermeasures	29
3.4	Future Work.....	30
3.4.1	Modify metadata function.....	30
3.4.2	Implementing more file types.....	30
3.4.3	Add alerts for unusual files.	30
3.4.4	Improving useability	30
3.4.5	Improving performance.	31
4	Reference list.....	32
	Appendices	34
	Appendix A – Full code	34
	Appendix B – Modify Metadata function	41
	Appendix C – Further Screen shots.....	42

1 INTRODUCTION

1.1 BACKGROUND

This report will cover what was done in relation to the scripting project but before continuing the reader must first understand metadata.

Metadata is highly valuable to almost any organization. This type of data is held on almost every file and is used to inform users about the file. Different types of files will contain different metadata, except text files as these do not contain any data. The program in its current stage only works with JPEG files. Jpeg metadata could contain basic information like image size, height, and width but there is far more valuable information to be found. This could be GPSInfo, Make, Model and loads more.

This information can be highly beneficial to several types of users, for example, police, digital forensics investigator, developers, or even less technical people trying to solve some issues on the computer. Hackers also could use this data for an array of tasks. A concerning issue is the GPSInfo held within an image file. This could allow a hacker to obtain the location of their target.

There is an estimate of 750 Billion photos exists on the internet, although this number is micro in comparison to the actual number of photos unshared worldwide. (Broz, 2022). Unfortunately trying to find the number of images with metadata relating to GPSInfo, etc, is impossible to find currently. However, taking a fraction of images on the internet presently will still equate to an extreme number. Take note that images offline which have metadata will still remain and could be used for a forensics investigation. Some images are stripped of data when uploading the image to an online platform. Below is a graph showing the number of photos taken a year and this reveals that 750 Billion online images are a fraction compared to the actual number of images.

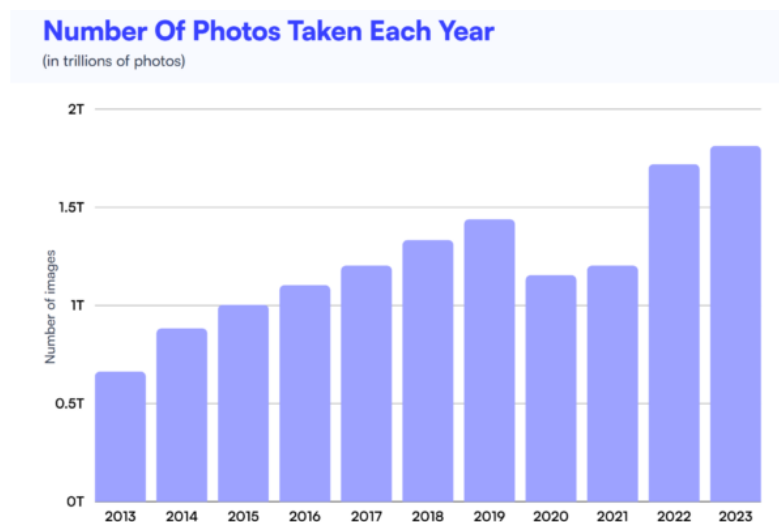


Figure 1 Number of photos taken each year taken from (Broz, 2022).

An abundance of websites allows users to upload images, for example Facebook, and this could be dangerous for several reasons. Of course, some websites will strip down the personal data but trusting these websites to not keep the data is a different question. Furthermore, these websites don't often make it obvious if they strip the metadata or not. Below is a small table, which shows some common websites and whether or not they strip metadata. The table also includes video metadata to include other types of files for comparison.

Does the website strip metadata?		
Website	Photo	Video
Discord	Yes	Yes
Github Comments	No	N/A
Google Chat	No	No
LinkedIn	Yes	Yes
Microsoft teams	No	No
Slack	Yes	No
Twitter	Yes	Yes
Wire and Yammer	No	No
Pinterest	No (Only GPSInfo)	N/A

Table 1: Websites and if they remove metadata adapted from (Steven, 2021).

Additionally, a useful fact about cloud storage reveals that 71% of data stored, is photos. Showing a large conveying area of images which could contain vulnerable data. Below is another graph to demonstrate the number of images available. (Armstrong, 2021)

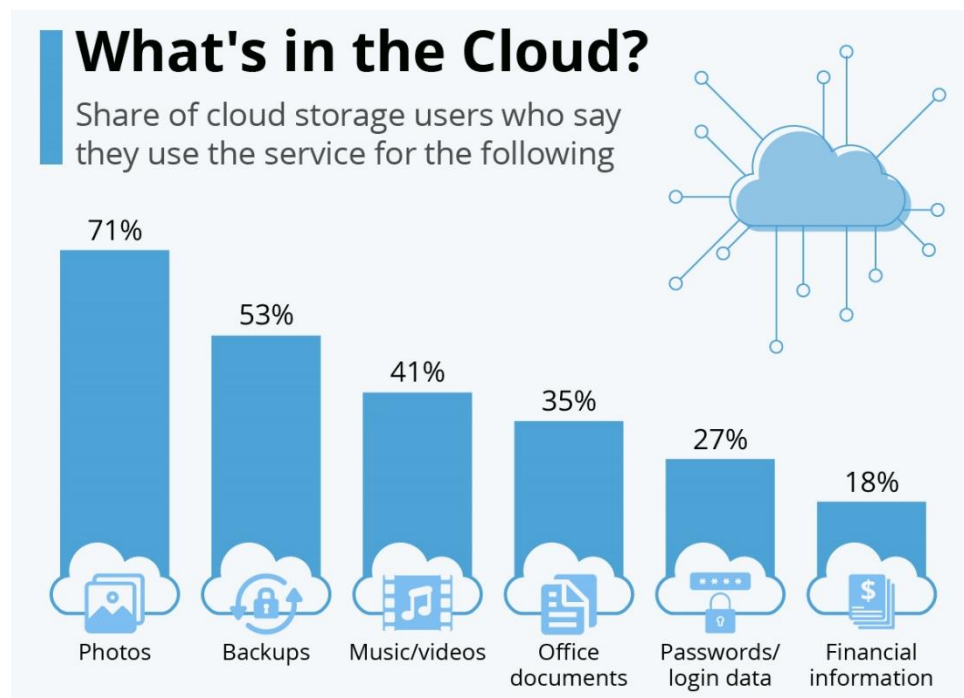


Figure 2 Cloud storage taken from (Armstrong, 2021).

This is the primary reason as why an application to analyses metadata is extremely useful. This will allow the user to strip the metadata within your local environment before sending and uploading anything. This ensures that the user's data is deleted before sharing. The application has further functions, for example geolocating. Which can be useful in a forensics investigation.

The application is built using python. This choice was made primary as the developer had more experience and was comfortable creating a tool using this language. A further reason is due to the accessible library's, which was utilized during this project.

1.2 AIM

The aims of this project are shown below:

- Create a metadata analyser.
 - Create a tool using python to analyse files.
 - Use appropriate libraries to reduce lines of code.
 - Use an appropriate structure to ensure code is reusable and include good practices, including try statements to prevent crashing.
 - Ensure tool is useable:
 - Allow users to select a single image or a folder of images.
 - Allow users to decide on several options listed below:
 - Deletion of metadata
 - Output table of data
 - Output a CSV and a HTML map if geo data is available.
 - Ensure the tool can take any file and determine file type before processing the Jpeg images, to prevent errors.
 - Ensure the tool can extract the data.
 - Ensure the tool can delete all the data except the image itself.
 - Ensure the tool can output the data to a CSV and create a HTML map.
 - This tool should be able to process image files providing different functions.
- Test the metadata analyser.
 - Test all areas of the tool to ensure comprehensive testing.
- Create a report on the project to further demonstrate the project.
 - Research information.
 - Report all findings.

1.3 ETHICAL USE OF SCRIPT

This script was created for ethical use. For example, to test and delete your own personal data from images you are uploading to the internet to increase online security. All test data will be taken from valid and from sources taken locally. This script is not intended to steal or use personal data from other images for any malicious purposes.

2 PROCEDURE

2.1 OVERVIEW OF REPORT

This section is a quick overview of what the report will cover and each section of the main sections of the report.

- Application background will disseminate information about the application and how the initial application is supposed to work.
- Application in use. This section will demonstrate the application in use and the variety of uses.
- During the Review of the code section, the report will examine the current code and how the process works with a clear narrative.
- After the review the report will evaluate several aspects of the tool, including performance.
- The final area of the report will discuss the overall advantages and limitations of the tool and future work which could be conducted. Followed by the References and Appendix's.

2.1.1 Tools and software used.

- Visual studio code to create and debug the application.
- Windows command prompt to test and run the application in a real-life environment.
- Python PIP was used to install all dependencies used within this project.
- Github was used to store and download project on different machines for testing.

2.2 APPLICATION BACKGROUND

The tool is called CMA.py for Casey's metadata analyser. The tool will need a file or a folder to work. Once the application has been initiated the tool will take the file. If a folder is provided the tool will go through each file individually and process the data individually. Each file will be opened, and all the metadata will be extracted and output to the terminal for the user to view.

Other flags can be used to perform other functions:

- -h : to output the help comments to help use the application.
- -d : to delete metadata of files.
- -o : to output to files, giving the user a CSV file of the data and if possible, a html file of a map with the coordinate of where the image was taken if not changed.
- --folder : to give a folder path.
- --file : to give a file path.
- -m : to modify data, currently not implemented see Future Work.

2.3 APPLICATION IN USE

To gain a better understanding of this application this section will demonstrate how to use the application. Below are some bullet points to explain how the report will test and show each possible output.

- Test output to the terminal on an image with metadata.
- Test output to the terminal on an image without metadata.
- Test output to a CSV and map file on an image with metadata.
- Test output to a CSV and map file on an image without metadata.
- Test deletion of metadata on an image, checking data before and after for comparison.

The above tests will be further performed on a folder to ensure that these tasks can be accomplished in a bulky environment.

2.3.1 Help Command “-h –help”.

This command: “python3 CMA.py -h” is for users who are not familiar with or need help with operating the tool. The tool will print to the terminal a series of instructions on how to operate the tool with some examples as well see Figure 3.

```
C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics>python3 CMA.py -h

  C A S E Y
    /  \  /  \
  M E T A D A T A
    /  \  /  \
  A N A L Y S E R

Please note this tool is for educational purposes

usage: [-h] (--file FILE | --folder FOLDER) [-d] [-o] [-m key value]

optional arguments:
  -h, --help            show this help message and exit
  --file FILE           The file path to the file to be analyzed, example: Desktop/image.png
  --folder FOLDER       The file path to the folder to be analyzed, example: Desktop/folder
  -d, --delete          WARNING ONLY: Use this flag to initiate the deletion metadata
  -o, --output          Use this flag to output results to a csv file and create a map, if applicable
  -m key value, --modify key value
                        Modify metadata with the specified key and value
                        Example: python your_script.py image.jpg -m Software ModifiedSoftware
```

Figure 3 CMA.py help command.

2.3.2 Test on image file with no other flags.

To call the script the on a single image file the command “python3 CMA.py –file Test1.jpg”. The image file is called Test1.jpg to provide a file the –file must be given before the file path.

The results were as expected. The script outputted to the terminal a neat table which revealed information about the image. The information held within the image was accurate based on a manual search, See Figure 31. Below is an image. As you can see a lot of valuable information is held about the image. Including Make, Model which is also accurate to the device. This was verified on an internet search. See Figure 30.

GPSInfo can also be seen. Although the individual number does not locate anyone this can be used to indicate that the image contains geo data. Further on the report will demonstrate how to extract and locate locational data from a jpg image. See Figure 4 for the results of the first test.

```
CASEY
METADATA
ANALYSER

Please note this tool is for educational purposes

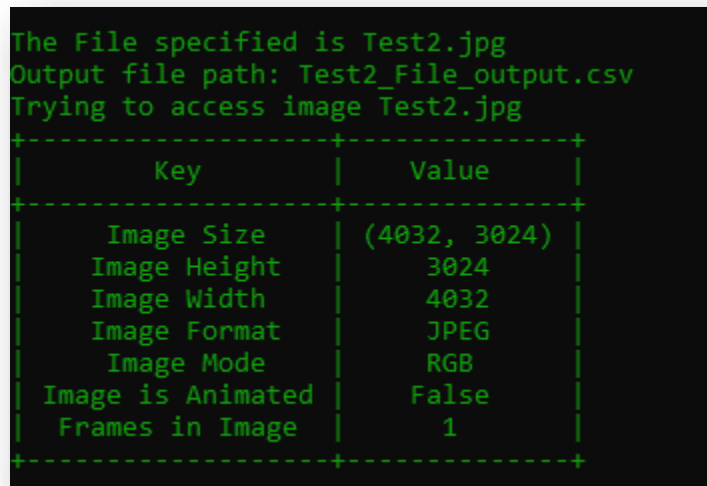
The File specified is Test1.jpg
Output file path: Test1_File_output.csv
Trying to access image Test1.jpg
+-----+-----+
| Key | Value |
+-----+-----+
| Image Size | (4000, 3000) |
| Image Height | 3000 |
| Image Width | 4000 |
| Image Format | JPEG |
| Image Mode | RGB |
| Image is Animated | False |
| Frames in Image | 1 |
| ImageWidth | 4000 |
| ImageLength | 3000 |
| GPSInfo | 696 |
| ResolutionUnit | 2 |
| ExifOffset | 238 |
| Make | samsung |
| Model | SM-N986B |
| Software | N986BXXU3FVC5 |
| Orientation | 6 |
| DateTime | 2022:04:21 16:08:02 |
| YCbCrPositioning | 1 |
| XResolution | 72.0 |
| YResolution | 72.0 |
+-----+-----+
```

Figure 4 Results from Test1.jpg

Please note that the banner “Casey’s metadata analyser” won’t be shown from now on to save space.

2.3.3 Test on image without metadata and other flags.

The command used was the same as previously used but the image used is called Test2.jpg as this file contains no metadata. Below is a screenshot of the results. See Figure 5.



```
The File specified is Test2.jpg
Output file path: Test2_File_output.csv
Trying to access image Test2.jpg
```

Key	Value
Image Size	(4032, 3024)
Image Height	3024
Image Width	4032
Image Format	JPEG
Image Mode	RGB
Image is Animated	False
Frames in Image	1

Figure 5 Results from Test2.jpg

Some data was still returned to a neat table despite no metadata. This is because the data returned is more mandatory for an image. This type of data does not reveal anything personal and will be used by applications to use the image correctly. The output was as expected. This reveals to the tester that the information is sanitized from personal data and the image contains no geo data.

This shows how much data is not needed when sending or sharing photos.

2.3.4 Test on image with metadata and output flag.

This test uses the command “python3 CMA.py -file Test3.jpg -o”. The -o is to output all results. The script will take the image and attempt to extract all metadata to a CSV file saving the file.

The script will also attempt to generate a map of where the image was taken saving the map as a HTML file. In this example the two files created are from the image file:

- Test3.jpg – Map.html
- Test3_File_output.txt (The format is a CSV, but user will need to open in an application that can read the file correctly)

Below is an image of all the results returned, See Figure 6. The results returned were accurate. The device was correct and the location of where the image was taken was also correct and extremely accurate. The CSV file was created and successfully named. The “,” found on each line of the CSV file is a formatting issue with CSV caused when opening a CSV file in a text editor instead of a spreadsheet.

The map is generated with the use of Folium and coordinates extracted from the image. The flag for output is shared to initiate the generation of the map. This could be separated to allow users more control. There are some messages which are outputted to help the users understand what is going on.

For example, the script will output “Outputting” to show the users that the -o flag has worked. Additionally, “map saved” or “no geo data” Will be outputted, depending on whether or not the image has geo data.

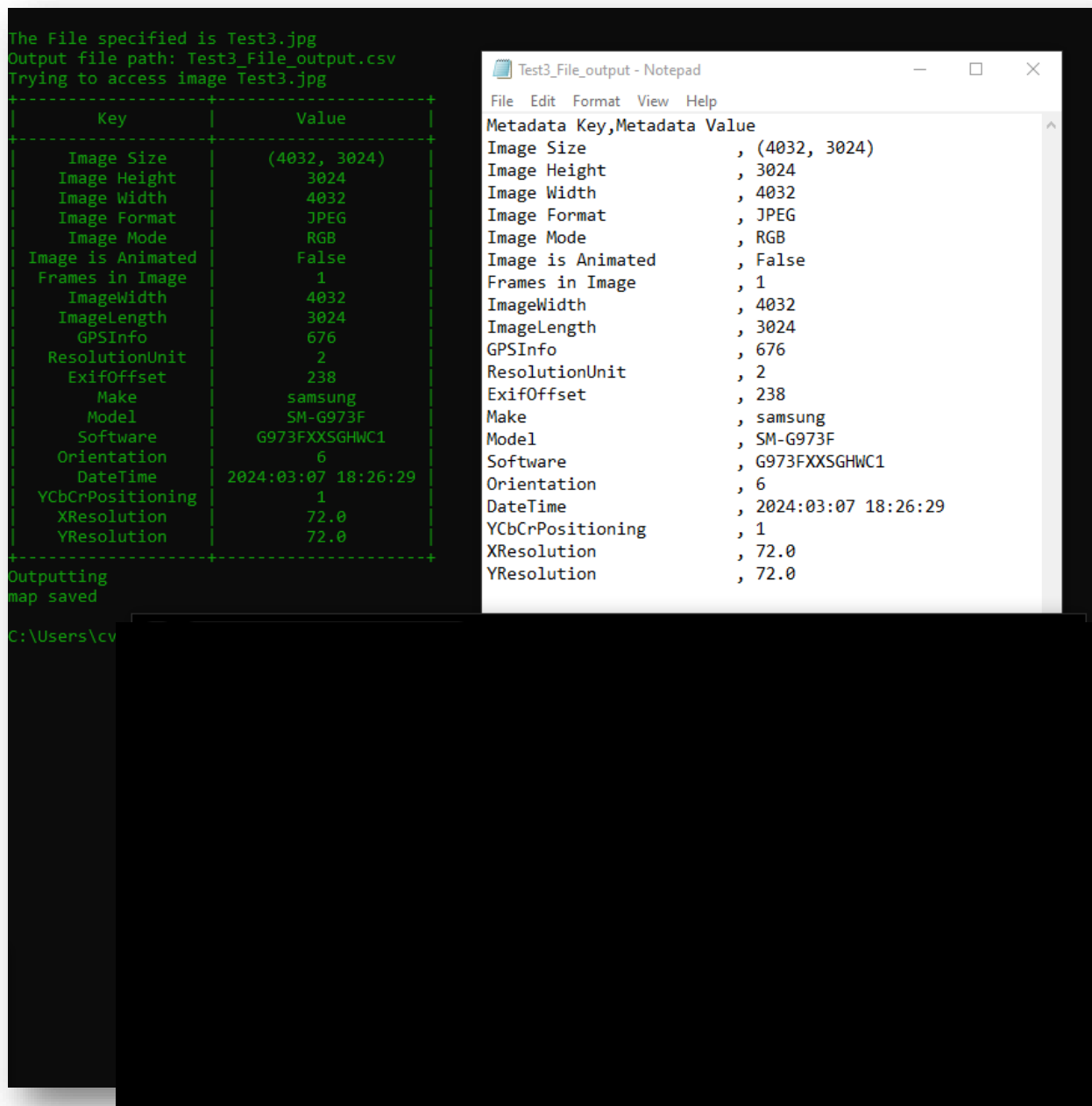
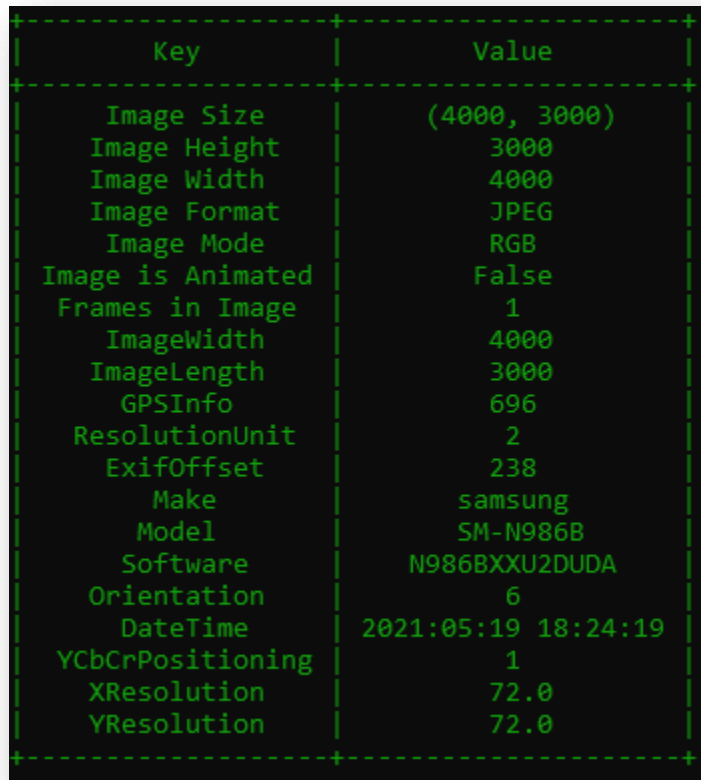


Figure 6 Results from Test3.jpg

2.3.5 Test deletion on image.

Before calling the deletion of an image, the original data was retrieved to compare the results. Below is the original metadata, See Figure 7. This reveals some personal data which could be dangerous sharing.

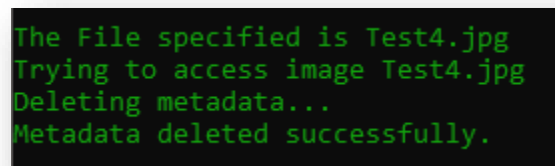
To initiate the deletion the -d flag must be provided. The command used is “python CMA.py –file Test4.jpg -d”.



Key	Value
Image Size	(4000, 3000)
Image Height	3000
Image Width	4000
Image Format	JPEG
Image Mode	RGB
Image is Animated	False
Frames in Image	1
ImageWidth	4000
ImageLength	3000
GPSInfo	696
ResolutionUnit	2
ExifOffset	238
Make	samsung
Model	SM-N986B
Software	N986BXXU2DUDA
Orientation	6
DateTime	2021:05:19 18:24:19
YCbCrPositioning	1
XResolution	72.0
YResolution	72.0

Figure 7 First results from Test4.jpg.

The results after the deletion only show whether or not the deletion was successful, See Figure 8.



```
The File specified is Test4.jpg
Trying to access image Test4.jpg
Deleting metadata...
Metadata deleted successfully.
```

Figure 8 Results from deletion of Test4.jpg.

To verify that the data was successfully deleted, and no personal data remains the user must now run the script again without the -d flag to analyze the results. As seen in Figure 9, the results were stripped down to the basic information. This has proved that the deletion function does work. Additionally printing the metadata after the deletion could prove useful.

```
The File specified is Test4.jpg
Trying to access image Test4.jpg
```

Key	Value
Image Size	(4000, 3000)
Image Height	3000
Image Width	4000
Image Format	JPEG
Image Mode	RGB
Image is Animated	False
Frames in Image	1

Figure 9 Second results from Test4.jpg.

2.3.6 Test script on folder of files.

The `-folder` flag can be used to run the script on the whole folder, individually analyzing each file. The command used was `"python3 CMA.py -folder tes"`. This process can take a lot longer depending on the size of the folder.

The `-o` for outputting the results does not need to be demonstrated as the results will be the same. The script will go through each file and perform the operations specified, delete, output, etc. Below are the results returned, Figure 10.

From Figure 10, its distinguishable to identify each image. The script will output stage with a path to the current file being processed. The results are as expected, the first file in the "tes" folder is a zip file which was flagged as an unknown file, See . This helps prevent the script crashing every time a non-jpg file is found. Having a quick look over each table's output can reveal which images have personal data.

To further use this application the `-d` flag can be used to go over every image and strip down all the metadata. The command used was `"python3 CMA.py -folder tes -d"` See Figure 32. This command will secure the images by removing personal data.

See Figure 33 to view the results after the deletion was performed. The differences can be seen in Figure 10 and Figure 33.


```

The Folder specified is tes
stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\test.zip
Unknown file type
stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder1.jpg
Trying to access image C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder1.jpg

```

Key	Value
Image Size	(4032, 3024)
Image Height	3024
Image Width	4032
Image Format	JPEG
Image Mode	RGB
Image is Animated	False
Frames in Image	1

```

stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder2.jpg
Trying to access image C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder2.jpg

```

Key	Value
Image Size	(4032, 3024)
Image Height	3024
Image Width	4032
Image Format	JPEG
Image Mode	RGB
Image is Animated	False
Frames in Image	1
ImageWidth	4032
ImageLength	3024
GPSInfo	696
ResolutionUnit	2
ExifOffset	238
Make	samsung
Model	SM-G973F
Software	G973FXXSGHWC1
Orientation	6
DateTime	2024:03:24 12:36:54
YCbCrPositioning	1
XResolution	72.0
YResolution	72.0

```

stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder3.jpg
Trying to access image C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder3.jpg

```

Key	Value
Image Size	(4032, 3024)
Image Height	3024
Image Width	4032
Image Format	JPEG
Image Mode	RGB
Image is Animated	False
Frames in Image	1

```

stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder4.jpg
Trying to access image C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder4.jpg

```

Key	Value
Image Size	(4032, 3024)
Image Height	3024
Image Width	4032
Image Format	JPEG
Image Mode	RGB
Image is Animated	False
Frames in Image	1

```

stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder5.jpg
Trying to access image C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder5.jpg

```

Key	Value
Image Size	(4032, 3024)
Image Height	3024
Image Width	4032
Image Format	JPEG
Image Mode	RGB
Image is Animated	False
Frames in Image	1
ImageWidth	4032
ImageLength	3024
GPSInfo	696
ResolutionUnit	2
ExifOffset	238
Make	samsung
Model	SM-G973F
Software	G973FXXSGHWC1
Orientation	6
DateTime	2024:03:20 11:55:24
YCbCrPositioning	1
XResolution	72.0
YResolution	72.0

Figure 10 First results from tes folder.

2.3.7 Additional file types

This project had intended to implement more file types. During the first implementation of Jpeg files, this revealed that the original goal was too large to complete to a high standard. Thus, the developer decided to only implement Jpeg files with a function to check for other file types, See Figure 11.

Only Jpeg and txt files are correctly implemented. The next goal is to implement PNG and PDF types. See Future Work 3.4. This section also prevents crashes as incorrect file types will be alerted to the terminal, but functionality won't change.

```
#check exstention and use an elif statement to direct to the correct function for analysis
if ex == "txt": print("Text files contain no metadata")
elif ex == "jpg" or ex == "jpeg":
    print(f"Trying to access image {filename}")
    image_metadata_extractor(filename=filename, delete_metadata=delete_metadata,
    modify_metadata_key=modify_metadata_key, modify_metadata_value=modify_metadata_value)
elif ex == "png" or ex == "PNG": print("png")
elif ex == "pdf": print("PDF file")
else:
    print("Unknown file type")

#other file types to be implemented
```

Figure 11 check file extension snippet.

2.4 REVIEW OF CODE AND PRACTICES

This script is programmed in python. This was suggested by the project brief. The script was also constructed using good programming practices (Thinkful, 2020).

2.4.1 Imports

A selection of imports was used for this project. All of these came in useful for reducing LOC(lines of code) or improving the scripts efficiency, reusability, or readability. Within the program are some comments of alternative libraries as well.

- Argparse
 - This import is to create a more useable tool as this will allow the user to parse arguments to the program before running the application.
- OS
 - This is used to help control the operating system, the script uses this import to get the file path of image files and where the output files will go.
- PIL, Image, ExifTags, TAGS
 - These imports are used to extract, read, and analyze the metadata with ease.
- Folium

- This import takes the latitude and longitude of the images coordinates to create and plot a map.
- CSV
 - This import is used to generate a CSV file of the metadata.
- Pretty Tables
 - Pretty table allows the script to generate a neat format to increase readability.

```
#Casey Donaldson
#2203162
#This application is to grab a file either image, docx and possibly video depending on the current stage.

#imports that will be used throughout the application
import argparse
import pandas
import os #used for file paths

#imports for image metadata (JPEG)
import PIL
from PIL import Image, ExifTags
import piexif #used in modified data
from PIL.ExifTags import TAGS

#Used for geo locating
from geopy.geocoders import Nominatim
#from gmplot import gmplot, alternative
import folium

#used for reporting
import csv
from prettytable import PrettyTable
```

Figure 12 Imports snippet.

2.4.2 Banner

At the top of the script a banner is shown, this is displayed each time the script is executed. This adds a certain level of personalization, See Figure 13. The banner is generated from an online website. (patorjk.com, n.d.)

```
#https://patorjk.com/software/taag/#p=display&f=Soft&t=Casey's%0AFootprinting%0AScanner
print(r"""
  _ _ _ _ _
 / \ / \ / \
 \ / \ / \ / \

| | | | | |
|_|_|_|_|_|
|_|_|_|_|_|

  _ _ _ _ _
 / \ / \ / \
 \ / \ / \ / \

Please note this tool is for educational purposes
""")
```

Figure 13 Banner.

2.4.3 Argument parser

The script takes certain arguments depending on what flags are raised. This was accomplished by the following snippet, See Figure 14.

The snippet shows a group being created to add arguments to. This then allows the script to detect and save the values as variables, to be used throughout the script. The script creates a special group for the `-file` and `-folder` which will demand only one be true but neither of them are required.

The modify value function works accordingly, the script detects a key and value before setting variables. The if statement at the bottom of the script detects if any values are missing, if true, then set both to none. The modify metadata function is currently non-operation but the arguments are correctly implemented.

```

#This is used to ensure the application is useable for new people by giving advice and help options
parser = argparse.ArgumentParser("", formatter_class=argparse.RawTextHelpFormatter)

group = parser.add_mutually_exclusive_group(required=True)
# Required arguments
group.add_argument("--file", help="The file path to the file to be analyzed, example: Desktop/image.png", required=False)
# Required arguments
group.add_argument("--folder", help="The file path to the folder to be analyzed, example: Desktop/folder", required=False)

# Optional args
parser.add_argument("-d", "--delete", help="WARNING ONLY: Use this flag to initiate the deletion metadata", action='store_true')
parser.add_argument("-o", "--output", help="Use this flag to output results to a csv file and create a map, if applicable", action='store_true')
parser.add_argument("-m", "--modify", nargs=2, metavar=('key', 'value'), help=
'''Modify metadata with the specified key and value
Example: python your_script.py image.jpg -m Software ModifiedSoftware
''')

#Now parse them.
args = parser.parse_args()

#Define variables.
File=str(args.file)
Folder=str(args.folder)
modify_metadata = args.modify
delete_metadata = args.delete
output_metadata = args.output

#If modify flag provided with both values raise function else define values as NONE
modify_metadata_key, modify_metadata_value = modify_metadata if modify_metadata else (None, None)

```

Figure 14 Argparse snippet.

2.4.4 Main section

The main section of the code only determines whether the path provided is a file or folder before calling the next function. From there the script uses a recursive technique to call further functions.

When the program runs, two variables are created. One for the file path, the other for the output used to output the file. Once all variables are set the file is sent to get the file extension for further processing.

Additionally, if a folder is provided the script will run through a loop, iterating through each file in the folder and setting a temporary file and output file variable before sending the individual file to the next function. If no file or folder is detected, then the script will tell the user to provide one of either. See Figure 15.

```

#if file provided
if args.file:

    #Print them out.
    print ("The File specified is", File)

    #create vari for file
    stage = args.file

    #create output file
    output_file = os.path.join(os.path.dirname(stage), f"{os.path.splitext(os.path.basename(stage))[0]}_File_output.csv")

    #execute on file only
    get_file_ex(filename=File)

elif args.folder:

    #Print them out.
    print ("The Folder specified is", Folder)

    #iterate through the folder
    for fileimage in os.listdir(Folder):

        #define vari for each image
        stage = os.path.abspath(os.path.join(Folder, fileimage))
        print(f"stage: {stage}")

        #define output file
        output_file = os.path.join(os.path.dirname(stage), f"{os.path.splitext(os.path.basename(stage))[0]}_File_output.csv")#

        get_file_ex(filename=stage)

else:
    print("Please provide either --file or --folder.")
    output_file = None

```

Figure 15 Main.

2.4.5 Get file extension function.

Each file is passed through this function to determine whether the file is processable, See Figure 16. Currently the only file that's allowed are JPEG's. Text files are picked up as not applicable as these files do not contain metadata. See Future Work for further information on implementing other file types.

The function determines the files by stripping the file down to the extension by using the "split" function in python and taking everything after the ".". To check whether the file type is valid the length is checked. Before deciding to process the data, the string is converted to lower case and compared to the current file types. If no matches are made the terminal will output an unknown file error message.

If a JPEG passes through the function to extract metadata will be called and all variables will be passed through. These include the file name, delete flag, and the modify data flags, the modify data flags do not alter the execution as they are not implemented, See Future Work 3.4.

```

#function to find what file is given
def get_file_ex(filename):

    #Split the file name and grab the exstention
    file_ex = filename.split('.')

    #file exstention exists
    if len(file_ex) > 1:
        #grab the exstention
        ex = file_ex[-1].lower() #convert to lower case

    #check exstention and use an elif statement to direct to the correct function for analysis
    if ex == "txt": print("Text files contain no metadata")
    elif ex == "jpg" or ex == "jpeg":
        print(f"Trying to access image {filename}")
        image_metadata_extractor(filename=filename, delete_metadata=delete_metadata,
                                modify_metadata_key=modify_metadata_key, modify_metadata_value=modify_metadata_value)
    elif ex == "png" or ex == "PNG": print("png")
    elif ex == "pdf": print("PDF file")
    else:
        print("Unknown file type")

    #other file types to be implemented

```

Figure 16 Get file extension function.

2.4.6 Image metadata extractor function.

Once the image metadata extractor function is called. The first instruction is to attempt to open the file in “read binary” mode, then open the file and save the file as a variable. All of this is held within a try statement to prevent the program crashing. This is a safety precaution in the event that a non-image file with a .jpg extension comes through. Once all the data is saved, the “.getexif()” function is called on the image and saved as a new variable called “exifdata”. See Figure 17.

```

#function to extract meta data from a jpg file
def image_metadata_extractor(filename, delete_metadata, modify_metadata_key=None, modify_metadata_value=None):

    try:
        #open the image and save as variable
        file = open(filename, "rb")
        image = Image.open(file)

    #exceptions that might cause a crash
    except FileNotFoundError:
        print(f"File not found - {filename}")
        return
    except PIL.UnidentifiedImageError as e:
        print(f"Error: Unable to identify image file - {filename}")
        return
    except Exception as x:
        print(f"Error - {x}")
        return #exit incase of error

    #get exif data for further analysis
    exifdata = image.getexif()

```

Figure 17 Image metadata function part 1.

The image data is then checked. If the data is “None”, then a message will print out alerting the user of the error, See Figure 18 and Figure 22.

2.4.7 Delete and modify metadata flags.

Once the image data is considered “not None”, the delete metadata flag is checked. If true, i.e., the -d flag has been provided. The delete will be attempted.

The script alerts the users before trying to convert all Exif data to none and overwriting the file. If the deletion is successful a success message, will be printed before returning to the function. If any errors occur, then the error exception will be printed to alert the user, See Figure 18.

At the bottom of Figure 18 another two flags are checked. If both these values are true (have a valid value) a message is printed out. This allows the user to know that the current implementation of the modify function is not functional, See Future Work.

Ideally this is where the modification function will be called before displaying any data, as shown by the comment see Figure 18.

```
#check data is not null
if exifdata is not None:

    #if delete data flag is raised
    if delete_metadata == True:
        #alert deleting data
        print("Deleting metadata...")
        try:
            # Remove exif data from the image
            image.info["exif"] = b""
            # Save the image with no data
            image.save(filename)
            print("Metadata deleted successfully.")
            return

        except Exception as e:
            print(f"Error deleting metadata: {e}")

    #if modified data raised then perform this
    if modify_metadata_key and modify_metadata_value:
        print("Modification is not implemented currently")
        #edit_metadata(filename, modify_metadata_key, modify_metadata_value)
```

Figure 18 Image metadata function part 2.

2.4.8 Extracting metadata.

Once the flags are completed the next step of the script is to grab the metadata. The script creates a dictionary filled with basic attributes which should be present within a Jpeg image, See Figure 19.


```
#grab basic information on most images
info_dict = {
    "Image Size": image.size,
    "Image Height": image.height,
    "Image Width": image.width,
    "Image Format": image.format,
    "Image Mode": image.mode,
    "Image is Animated": getattr(image, "is_animated", False),
    "Frames in Image": getattr(image, "n_frames", 1)
}
```

Figure 19 Image metadata function part 3.

The dictionary is then further populated. This was achieved using a for loop to iterate through each tag and if the tag is present the tag is saved as a variable and then decoded to a human readable format before the tag and data is updated at the end of the dictionary, See Figure 20.

```
#go through each tag and value
for tag_id in exifdata:
    tag = TAGS.get(tag_id, tag_id)
    data = exifdata.get(tag_id)

    #if there add to dict
    if isinstance(data, bytes):
        data = data.decode()

    #updating dict
    info_dict.update({tag : data})
```

Figure 20 Image metadata function part 4.

2.4.9 Displaying metadata.

The next step of the script is to report the data to the user. This is achieved by using 3 different methods. The first is displaying the data to the terminal. This is the default approach and happens on default. This was achieved by using the library Pretty Tables (Maurits, 2024).

The script creates an instance of the table and populates the field names, these are key and value. Once the table is created, a loop is then performed to iterate through each of the key value pairs within the dictionary adding them to the table using the “.add_row” function.

Once the For loop is finished and the table is populated, the table is printed. The neat format is created using the “.get_string” function, See Figure 21.

```
#create a table
table = PrettyTable()
#add feild names
table.field_names = ["Key", "Value"]

#iterate through the dict
for key, value in info_dict.items():
    #add each key value to table
    table.add_row([key, value])

#print the table to the terminal
print(table.get_string())
```

Figure 21 Image metadata function part 5.

The next two methods are outputting the data to a CSV file and creating a map with the location of where the image was taken. These will only execute if the output flag is given (-d). Once the output flag is given. The script will alert the user and pass the dictionary to the “dict_to_csv” function to create and format the file.

Additionally, if GPS data was found in the data extraction, then the “image_locator” function will be called on the file, See Figure 22.

```
#if output is wanted
if output_metadata == True:
    print("Outputting")
    #this is used to report to a csv file
    dict_to_csv(info_dict, csv_file_path=output_file)

    #attemp geo data as well
    if 'GPSInfo' in info_dict:
        image_locator(filename=filename)
else:
    print("Unable to read any exif data.")
```

Figure 22 Image metadata function part 6.

2.4.10 Dictionary to CSV function.

To output a CSV file the script alters the user and creates a file unless one is already present with the same name. The file is opened in write mode and the name is taken from the output file variable. This ensures the name is "ImageName_File_output.csv", creating a unique file for each image.

To populate the file the CSV library is used (Python Software Foundation, 2020).

The function creates an instance of the csv writer. The instance is then given the headers of the columns using the ".writerow" function giving the two values (key, value). This prints the first row as the headers.

A For loop is created to iterate through each key and value within the dictionary. Each key and value are formatted correctly to ensure the CSV file is handled correctly. Once the key and value variables are formatted the .write function is used to add the two columns to the CSV file.

After the For loop is completed, the function will end and close the file, returning to the previous function to check for GPS information, See Figure 23.

```
#dictionary to csv file function
def dict_to_csv(dict, csv_file):

    print(f"Output file path: {output_file}")
    #print(csv_file_path)
    #open new csv file
    with open(csv_file, 'w', newline='') as csvfile:

        #define the writer
        csv_writer = csv.writer(csvfile)

        #write the headers
        csv_writer.writerow(["Metadata Key", "Metadata Value"])

        #iterate through the dict
        for key, value in dict.items():

            #Go through each key value pair and format the key and value before writing to the csv file
            formatted_key = f"{key.strip()}"
            formatted_value = f"{str(value).strip()}"

            #write the csv file
            csvfile.write(f"{formatted_key:25}, {formatted_value}\n")
```

Figure 23 Dictionary to CSV file function.

2.4.11 Image locator function.

The full function is held within a try statement to avoid crashing as GPS data can be quite difficult. The function first opens the image and populates a dictionary with the Exif data. Before the script continues another check is performed to ensure GPS data is present. If a Value error occurs the exception will be printed, See Figure 24.

```

#function to get location of data if possible
def image_locator(filename):
    try:
        #re-open image
        img = Image.open(filename)

        #collect metadata and tags
        exifdata = {
            ExifTags.TAGS[key]: value
            for key, value in img._getexif().items()
            #check key/tags
            if key in ExifTags.TAGS
        }

        # Check if GPSInfo is present
        if 'GPSInfo' not in exifdata:
            raise ValueError("GPS information not found")

```

Figure 24 Image locator function part 1.

After the values are valid and present the script must then format the data before creating the map. The first step is to collect the actual coordinates. To accomplish this the script creates and saves two variables, North and East. These actually represent latitude and longitude, discussed later.

The Exif data is formatted like { "N or S" , "Coordinates" , "E or W" , "Coordinates" }, The script currently only needs the coordinates and not the direction. This is why the 2nd and 4th fields are saved, See Figure 25.

```

#declare that inside GPSinfo 2nd field is the north coordinates
north = exifdata['GPSInfo'][2]
#declare that inside GPSinfo 4th field is the east coordinates
east = exifdata["GPSInfo"][4]

```

Figure 25 Image locator function part 2.

The next step is to convert the coordinate to latitude and longitude format. This was achieved by following a guide on converting the coordinates ([lweb.cfa.harvard.edu](http://web.cfa.harvard.edu), n.d.). The math was then extrapolated into the below snippet saving both the latitude and longitude, See Figure 26.

```

#reform the format of the coordinates to lat and lng for processing
lat = (((north[0] * 60) + north[1]) * 60) + north[2]) / 60 / 60
lng = (((east[0] * 60) + east[1]) * 60) + east[2]) / 60 / 60

```

Figure 26 Image locator part 3.

The latitude and longitude will be checked with their corresponding field to determine whether the decimal number is positive or negative. For example, if the latitude has an S beside it the decimal number is negative. Thus, the latitude is changed to a negative number. This is similar to longitude, See Figure 27.

The longitude and latitude converter are defined by the Exif data provided in this format: { "N or S" , "Coordinates", "E or W" , "Coordinates"}. This shows that the first field is responsible for the Latitude and the 3rd for the longitude. An error occurred here which was proven difficult to fix as this snippet was missing and no errors occurred see Figure 27. The problem was logical and was solved with the use of another tool to check the location of longitude and latitude. To better understand this please see Reference list.

Once the latitude and longitude has been checked and converted if needed, they are then further saved as a floating-point integer to be read by Folium.

```
#Check if the value is positive or negative. north or south ect.
if exifdata.get('GPSInfo')[1] == 'S':
    lat = -lat
if exifdata.get('GPSInfo')[3] == 'W':
    lng = -lng

#print(lat, lng)

#change from fraction to floating points
lat, lng = float(lat), float(lng)
```

Figure 27 Image locator part 4.

To create the map of the library Folium was used (Story, 2024). This helps to plot the coordinates on a pre-made map and save it locally. This was achieved by defining a map aimed at the latitude and longitude with a default zoom set.

A marker was further added to pinpoint the location of the image, adding some text and color to the marker to add design. Once the map is set, the script saves the map as "image name – Map.html". See Figure 28.

```
#folium use create a instance of the map with the lat and lng including current zoom
fmap = folium.Map(location=[lat, lng], zoom_start=12)
#add marker to map
folium.Marker(location=[lat,lng], popup='Location', icon=folium.Icon(color='blue')).add_to(fmap)
#save the map
fmap.save(f'{filename} - Map.html')
print("map saved")
```

Figure 28 Image locator part 5.

If any error occurs during the function they will be printed out to the user to alert the user. This could be due to them deleting the data as the function continues to run after deleting the data so the exception will be handled as a success message for deleting the data. If an error occurs without the delete flag being raised then the error will be printed and the terminal will tell the user that No geo data exists, See Figure 29.

```
except Exception as e:
    if delete_metadata == True:
        print("Data correctly deleted")
    else:
        print(f"Error exception: {e}")
        print("No Geo Data")
```

Figure 29 Image locator part 6.

This concludes the review of the code. The last area of code which will be discussed is the modification of metadata see Future Work. To view the full code, see Appendix A.

2.5 EVALUATION

2.5.1 Coverage

The testing involved revealed that every Jpeg was correctly analysed, and data extracted. An unusual result did appear during the image locator function. This was during a test of an image and testing the function. The result encountered a “NaN” error exception due to the GPS information being unreadable and incorrectly formatted, See Figure 34. This issue was only found in one of several tests. This revealed that the tool is not 100% effective at creating maps. Additionally, this could be due to the data held in the image is incorrectly stored. This could happen when images are sent throughout the internet via several applications. Fortunately, the application caught the error and avoided crashing.

Unfortunately, this application covers only Jpeg in the current stage. This is highly useful as discussed previously in the introduction, a lot of files are image files. This still leaves a large area untapped.

Additionally, More specific metadata could be extracted, this would need further research and more implementation. This would result in more in-depth extraction.

2.5.2 Performance

The excessive try statements provide a semi-crash proof application. However, not full proof. The application could benefit with an additional accept statement before initiating the deletion as this could be fairly large and furthermore, the deletion could be accidental.

This application's performance is effective for several files. However, in forensics investigations thousands of files could be held. This could start to slow down the implementation of multi-threaded process is discussed further in Future Work. Overall, the performance is acceptable and does not take long to process the data whether It's deleting, outputting, or viewing the data.

Furthermore, the storage it could use to replicate all the files, for example one image file creates two extra files. This could be reduced by using multiple markers on a single map and updating a CSV file and maintaining one CSV file, see Future Work.

2.5.3 Useability

The script first requires the user to install some dependencies. However, once these are installed this script will work effectively on Windows and Linux without any issues. To expand onto MacOS the developer would need access to such an environment to test the script and implement any solutions to the software.

The developer implemented the Argument parser library to increase the overall useability of the application, to increase the speed the application can be used and to implement a help function. Additionally, the "-folder" flag was implemented to increase speed and allow the user to run the script on multiple files simultaneously.

3 DISCUSSION

3.1 GENERAL DISCUSSION

Overall, the current application has accomplished all of the current aims. The tool was implemented using the suggested python script. This provided useful with the python libraries found which supported the development of the script.

The tool was created following along with good practices to ensure the tool can be updated without overhead problems, for example organizing or fixing bad code. The tool has also shown to be useable allowing the user to:

- Select a single or multiple images.
- Select deletion, output, and map output.
- Ensure the script can handle the file type automatically.
- Ensure the script can analyze the data.
- Ensure the script is tested and ensure the script can correctly delete and output data.

Several other factors are to be considered to improve upon this tool. These are discussed below.

3.2 LIMITATIONS AND ADVANTAGES

3.2.1 Limitations

This application is highly effective and does the tasks which it was intended to. However, the limited file types can make this application obsolete to improve on this application further file types should be added.

This application could also suffer if given thousands of images to process. To solve this issue multi-threading would be the answer. Additionally, since each file creates a CSV and Map for each image this could take up a lot of storage space when outputting data, another issue could be the difficulty to keep track of each file, Discussed in Future Work.

The script does still leave some manual investigation and does not fully remove the manual inspection. This still hinders the process and could be further improved upon by using built-in alerts to speed up investigations. Discussed in Future Work.

Furthermore, this tool could still contain unknown errors. Mentioned previously the “NaN” error occurred. These could still occur, and this could hinder the tool. Fortunately, the tool handles these and the application won’t crash and will instead alert the user and continue to the best of the script’s ability.

3.2.2 Advantages

This application does contain several advantages.

The script uses try statements to prevent any errors. This is helpful to prevent crashing which could happen a lot when trying to render an image in a script. Additionally, several functions are in place to ensure the code is reusable.

Furthermore, This script saves the output files with unique names to increase the readability of smaller processes, whereas the image volume does not exceed 100 images. This usually sorts the files beside each other since the names are almost identical.

This script uses the Pretty Tables library to improve the format of the output to the terminal. During testing this improvement revealed a major change. The format has a clearer border with each key, value pair being recognized without any issues.

Even with the process slowing down due to several hundred images this could still prove better than outsourcing to online websites which may try and charge you for processing more than X number of times/images.

3.3 COUNTERMEASURES

This tool is more defensive than offensive, To prevent malicious hackers stealing personal metadata the best solution is to remove the metadata yourself.

As discussed previously some social media platforms will remove your metadata but not all of them. To increase your online safety, it's best to remove the data. This can be done in several ways, please be aware that removing metadata to hide malicious actions could still be considered illegal.

- Online Websites to remove metadata.
- CMA.py to remove your metadata from images.
- Other tools which are designed for removing metadata.
- A low technical solution would be using the snippet tool to snippet the image and save the snippet rather than the photo taken by the camera.
- Windows built in function if the user is on a Windows OS.

All of the above solutions have benefits and negative benefits. It is still more secure to remove your metadata, rather than relying on a social media platform to remove and delete the metadata.

The main benefits of using CMA.py are:

- Can use the application on your local machine.
- Can view your metadata.
- Allows the user to delete metadata without even uploading the image to the internet.
 - Could avoid man in the middle attacks.
- Can trust that the data is removed.
- Can trust the data is not stored elsewhere.
- Easy to use for low technical users.

The application can still undergo a lot of improvements as this is discussed below.

3.4 FUTURE WORK

3.4.1 Modify metadata function.

A function which is commented out can be seen in Appendix B – Modify Metadata function. This needs to be fixed. The issue appears to be due to the tag value being incorrectly formatted when performing a comparison. This function was not part of the original aims and could be seen as gold plating (Aldrige, 2021).

The idea behind this function is to allow the user to provide a key and a value to which will change the value of the metadata which relates to the same key. This was a simple concept but brought up a lot of other issues. Besides gold plating an ethical question which was raised. Would users who are allowed to alter metadata with ease use the tool unethically?

The work which will need to be conducted is an investigation on the ethical premises and the fix to correctly change the metadata.

3.4.2 Implementing more file types.

Discussed in the introduction, other file types are due to be implemented. This could be an extremely large process as extracting other metadata could require completely different functions. Fortunately, getting the file extension and the CSV file should be re-useable.

The image locator function will be unique to Jpeg files as these are the only files which contain GPSInfo. With more file types of the output could be increased, which as well could affect performance. Video files have been researched and mp3 and mp4 could contain valuable data.

This work could be integrated into a python library which removes image GPSInfo data or other personal data from files before using the file for other purposes.

3.4.3 Add alerts for unusual files.

To improve upon this tool, you could add alerts or other methods of notifying the user of certain incidents. This could help improve manual inspection.

Certain alerts could be:

- Image taken after today's date.
- Make and model do not match. iPhone Make with a Samsung Model.
- Ai could be used to recognize objects within the image if the file is an image file.
- Could check the binary data of the file to check if the file extension matches that of the actual extension.

More alerts could be implemented to increase manual inspection. This sort of work will start to aim the tool towards a certain niche. Another interesting idea could be trying to locate images without using GPSInfo. For example, Ai has been used to locate images based on the background and other factors. This could be integrated to locate where the image was taking without coordinates.

3.4.4 Improving useability

The useability of the application is acceptable some improvements which were identified at the end of implementation are:

- Allowing the user to specify a CSV, Map or both rather than using a single flag to output both files. This will give more flexibility to the user.
- The final improvement is improving the useability of the code. The terminal will output the dictionary to the terminal in a neat format. This could be improved by creating a separate function to print to the terminal. This method will allow implementing more file types a lot easier.

3.4.5 Improving performance.

The program works effectively and does not suffer from any performance issues currently. However, this could be due to the lack of testing on thousands of files. It's fair to predict that with thousands of images the performance will decline due to the volume of files.

Furthermore, the storage of the device could suffer. This is a small point but below is some solutions to implement which could improve the performance of the application.

3.4.5.1 *Multi-threading*

Multi-threading has been used for a long time now. This concept, in simple terms, is when you allow your computer to use more workers (threads) to complete more work sooner possibly by dividing the amount of work needing to be done.

This application could divide the amount of volume of images between a fixed % of the processor to process each image. This pattern follows the farming pattern as each worker will be working on a list of tasks.

Additionally, this could be further improved by creating two threads per image if the -o is given. This will allow each image to process the metadata with another thread working on preparing the map and CSV file. This pattern follows a folk pattern. To summarize, the application will process several images at once, each image will have two threads working on two tasks simultaneously.

These will improve the application when processing large amounts of data but with smaller folders this could actually slow down the process with all the overhead of creating the threads.

3.4.5.2 *One file instead of multiple*

The storage could suffer. To prevent this each file could add to an existing CSV and Map rather than creating new files for each image processed.

This will allow one map named after the folder which will have several markers each named after the appropriate image. This could also be used to improve manual inspection as visualization will be easier to see with all present markers on one map.

Similar with the CSV file, each image could have their own header with their own values. This would require more data formatting, but the result would be far superior.

If multi-threading is used then to implement this approach, mutexed will be used to ensure no data is missed or entered twice.

4 REFERENCE LIST

- Aldrige, E. (2021). *Gold Plating vs. Scope Creep*. [online] Project Management Academy Resources. Available at: <https://projectmanagementacademy.net/resources/blog/gold-plating-vs-scope-creep/> [Accessed 29 Mar. 2024].
- Armstrong, M. (2021). *Infographic: What's in the Cloud?* [online] Statista Infographics. Available at: <https://www.statista.com/chart/25896/gcs-cloud-storage-services-usage/>.
- Broz, M. (2022). *How Many Photos Are There? 50+ Photos Statistics (2022)*. [online] photutorial.com. Available at: <https://photutorial.com/photos-statistics/> [Accessed 20 Mar. 2024].
- Farrier, Ilie (2023). *What Is Metadata: Definition and Meaning*. [online] What Is Metadata: Definition and Meaning. Available at: <https://www.avast.com/c-what-is-metadata#:~:text=Metadata%20is%20a%20set%20of> [Accessed 31 Mar. 2024].
- lweb.cfa.harvard.edu. (n.d.). *How to Convert to Map Coordinates*. [online] Available at: https://lweb.cfa.harvard.edu/space_geodesy/ATLAS/cme_convert.html [Accessed 27 Mar. 2024].
- Maurits, L. (2024). *prettytable: A simple Python library for easily displaying tabular data in a visually appealing ASCII table format*. [online] PyPI. Available at: <https://pypi.org/project/prettytable/> [Accessed 27 Mar. 2024].
- patorjk.com. (n.d.). *Text to ASCII Art Generator (TAAG)*. [online] Available at: <https://patorjk.com/software/taag/#p=display&f=Graffiti&t=Type%20Something%20>.
- Python Software Foundation (2020). *csv — CSV File Reading and Writing — Python 3.8.1 documentation*. [online] Python.org. Available at: <https://docs.python.org/3/library/csv.html> [Accessed 27 Mar. 2024].
- Readthedocs.io. (2011). *Pillow — Pillow (PIL Fork) 6.2.1 documentation*. [online] Available at: <https://pillow.readthedocs.io/en/stable/> [Accessed 27 Mar. 2024].
- Steven (2021). *Which Social Media Networks Remove EXIF Data?* [online] Steven Woodhall. Available at: <https://stevenwoodhall.com/which-social-media-networks-remove-exif-data/> [Accessed 20 Mar. 2024].

Story, R. (2024). *folium 0.12.1.post1 : Make beautiful maps with Leaflet.js & Python*. [online] PyPI. Available at: <https://pypi.org/project/folium/> [Accessed 27 Mar. 2024].

Thinkful (2020). *Coding Best Practices*. [online] Thinkful. Available at: <https://www.thinkful.com/blog/coding-best-practices/>.

APPENDICES

APPENDIX A – FULL CODE

```
#Casey Donaldson
#2203162
#This application is to grab a file either image, docx and possibly video
depending on the current stage.

#imports that will be used throughout the application
import argparse
#import pandas
import os #used for file paths

#imports for image metadata (JPEG)
import PIL
from PIL import Image, ExifTags
import piexif #used in modified data
from PIL.ExifTags import TAGS

#Used for geo locating
from geopy.geocoders import Nominatim
#from gmplot import gmplot, alternative
import folium

#used for reporting
import csv
from prettytable import PrettyTable

#My banner, lovely.
#https://patorjk.com/software/taag/#p=display&f=Soft&t=Casey's%0AFootprinting%0AScanner
print(r"""

 _   _   _   _   \ /
/_ _ , / \ / _ | _ \ /
 \_ , / ~ \ . _ / | _ |

      | \ / | | _ | | \ / \ / \ /
      | | | _ | | / ~ \ / ~ \ / ~ \
```

```
  /\  | \ |  /\  |  \ /  \_  \_  \_
 /~~\ | \ | /~~\ |  \_  |  \_  |  \_
```

Please note this tool is for educational purposes

```
""")

#function to extract meta data from a jpg file
def image_metadata_extractor(filename, delete_metadata,
modify_metadata_key=None, modify_metadata_value=None):

    try:
        #open the image and save as variable
        file = open(filename, "rb")
        image = Image.open(file)

        #exceptions that might cause a crash
        except FileNotFoundError:
            print(f"File not found - {filename}")
            return
        except PIL.UnidentifiedImageError as e:
            print(f"Error: Unable to identify image file - {filename}")
            return
        except Exception as x:
            print(f"Error - {x}")
            return #exit incase of error

        #get exif data for further analysis
        exifdata = image.getexif()

        #check data is not null
        if exifdata is not None:

            #if delete data flag is raised
            if delete_metadata == True:
                #alert deleting data
                print("Deleting metadata...")
                try:
                    # Remove exif data from the image
                    image.info["exif"] = b""
```

```

        # Save the image with no data
        image.save(filename)
        print("Metadata deleted successfully.")
        return

    except Exception as e:
        print(f"Error deleting metadata: {e}")

# if modified data raised then perform this
if modify_metadata_key and modify_metadata_value:
    print("Modification is not implemented currently")
    # edit_metadata(filename, modify_metadata_key, modify_metadata_value)

# grab basic information on most images
info_dict = {
    "Image Size": image.size,
    "Image Height": image.height,
    "Image Width": image.width,
    "Image Format": image.format,
    "Image Mode": image.mode,
    "Image is Animated": getattr(image, "is_animated", False),
    "Frames in Image": getattr(image, "n_frames", 1)
}

# go through each tag and value
for tag_id in exifdata:
    tag = TAGS.get(tag_id, tag_id)
    data = exifdata.get(tag_id)

    # if there add to dict
    if isinstance(data, bytes):
        data = data.decode()

    # updating dict
    info_dict.update({tag : data})

# create a table
table = PrettyTable()
# add field names
table.field_names = ["Key", "Value"]

# iterate through the dict
for key, value in info_dict.items():
    # add each key value to table
    table.add_row([key, value])

```



```

        #print the table to the terminal
        print(table.get_string())

        #if output is wanted
        if output_metadata == True:
            print("Outputting")
            #this is used to report to a csv file
            dict_to_csv(info_dict, csv_file=output_file)

            #attemp geo data as well
            if 'GPSInfo' in info_dict:
                image_locator(filename=filename)
        else:
            print("Unable to read any exif data.")

#fuction to get location of data if possible
def image_locator(filename):
    try:
        #re-open image
        img = Image.open(filename)

        #collect metadata and tags
        exifdata = {
            ExifTags.TAGS[key]: value
            for key, value in img._getexif().items()
            #check key/tags
            if key in ExifTags.TAGS
        }

        # Check if GPSInfo is present
        if 'GPSInfo' not in exifdata:
            raise ValueError("GPS information not found")

        #declare that inside GPSinfo 2nd field is the north coordinates
        north = exifdata['GPSInfo'][2]
        #declare that inside GPSinfo 4th field is the east coordinates
        east = exifdata["GPSInfo"][4]

        #reform the format of the coordinates to lat and lng for processing
        lat = (((north[0] * 60) + north[1]) * 60) + north[2]) / 60 / 60
        lng = (((east[0] * 60) + east[1]) * 60) + east[2]) / 60 / 60

```

```

        #Check if the value is positive or negative. north or south ect.
        if exifdata.get('GPSInfo')[1] == 'S':
            lat = -lat
        if exifdata.get('GPSInfo')[3] == 'W':
            lng = -lng

        #print(lat, lng)

        #change from fraction to floating points
        lat, lng = float(lat), float(lng)

        #check lat and long
        #print(lat, lng)

        #folium use create a instance of the map with the lat and lng including
current zoom
        fmap = folium.Map(location=[lat, lng], zoom_start=12)
        #add marker to map
        folium.Marker(location=[lat,lng], popup='Location',
icon=folium.Icon(color='blue')).add_to(fmap)
        #save the map
        fmap.save(f'{filename} - Map.html')
        print("map saved")

    except Exception as e:
        if delete_metadata == True:
            print("Data correctly deleted")
        else:
            print(f"Error exception: {e}")
            print("No Geo Data")

#dictionary to csv file function
def dict_to_csv(dict, csv_file):

    print(f"Output file path: {output_file}")
    #print(csv_file_path)
    #open new csv file
    with open(csv_file, 'w', newline='') as csvfile:

        #define the writer
        csv_writer = csv.writer(csvfile)

        #write the headers

```

```

        csv_writer.writerow(["Metadata Key", "Metadata Value"])

    #iterate through the dict
    for key, value in dict.items():

        #Go through each key value pair and format the key and value before
        #writing to the csv file
        formatted_key = f"{key.strip()}"
        formatted_value = f"{str(value).strip()}"

        #write the CSV file
        csvfile.write(f"{formatted_key:25}, {formatted_value}\n")

#function to find what file is given
def get_file_ex(filename):

    #Split the file name and grab the exstention
    file_ex = filename.split('.')

    #file exstention exists
    if len(file_ex) > 1:
        #grab the exstention
        ex = file_ex[-1].lower() #convert to lower case

        #check exstention and use an elif statement to direct to the correct
        #function for analysis
        if ex == "txt": print("Text files contain no metadata")
        elif ex == "jpg" or ex == "jpeg":
            print(f"Trying to access image {filename}")
            image_metadata_extractor(filename=filename,
delete_metadata=delete_metadata,
            modify_metadata_key=modify_metadata_key,
modify_metadata_value=modify_metadata_value)
        elif ex == "png" or ex == "PNG": print("png")
        elif ex == "pdf": print("PDF file")
        else:
            print("Unknown file type")

    #other file types to be implemented

```

```

#This is used to ensure the application is useable for new people by giving
advice and help options
parser = argparse.ArgumentParser(" " ,
formatter_class=argparse.RawTextHelpFormatter)

group = parser.add_mutually_exclusive_group(required=True)
# Required arguments
group.add_argument("--file", help="The file path to the file to be analyzed,
example: Desktop/image.png", required=False)
# Required arguments
group.add_argument("--folder", help="The file path to the folder to be analyzed,
example: Desktop/folder", required=False)

# Optional args
parser.add_argument("-d", "--delete", help="WARNING ONLY: Use this flag to
initiate the deletion metadata", action='store_true')
parser.add_argument("-o", "--output", help="Use this flag to output results to a
csv file and create a map, if applicable", action='store_true')
parser.add_argument("-m", "--modify", nargs=2, metavar=('key', 'value'), help=
'''Modify metadata with the specified key and value
Example: python your_script.py image.jpg -m Software ModifiedSoftware
''')

#Now parse them.
args = parser.parse_args()

#Define variables.
File=str(args.file)
Folder=str(args.folder)
modify_metadata = args.modify
delete_metadata = args.delete
output_metadata = args.output

#If modify flag provided with both values raise function else define values as
NONE
modify_metadata_key, modify_metadata_value = modify_metadata if modify_metadata
else (None, None)
#If file provided
if args.file:

    #Print them out.
    print ("The File specified is", File)

```

```

#create vari for file
stage = args.file

#create output file
output_file = os.path.join(os.path.dirname(stage),
f"{os.path.splitext(os.path.basename(stage))[0]}_File_output.csv")

#execute on file only
get_file_ex(filename=File)

elif args.folder:

    #Print them out.
    print ("The Folder specified is", Folder)

    #iterate through the folder
    for fileimage in os.listdir(Folder):

        #define vari for each image
        stage = os.path.abspath(os.path.join(Folder, fileimage))
        print(f"stage: {stage}")

        #define output file
        output_file = os.path.join(os.path.dirname(stage),
f"{os.path.splitext(os.path.basename(stage))[0]}_File_output.csv")

        get_file_ex(filename=stage)

else:
    print("Please provide either --file or --folder.")
    output_file = None

```

APPENDIX B – MODIFY METADATA FUNCTION

```

#out of use Todo
def edit_metadata(filename, key, value):
    try:
        # Open the image file
        image = Image.open(filename)

        # Get the existing metadata
        exif_dict = piexif.load(image.info["exif"])

        # Iterate through data to find tag relating to key

```

```

for tag, tag_value in exif_dict.items():
    print(f"tag - {tag}")
    print(f"key - {key}")
    if tag == key:
        exif_dict[tag] = value
        print(tag, tag_value)
        break

# Save the image with updated metadata
exif_bytes = piexif.dump(exif_dict)
image.save(filename, exif=exif_bytes)

print("Metadata updated")

except Exception as x:
    print(f"Error: {x}")

```

APPENDIX C – FURTHER SCREEN SHOTS

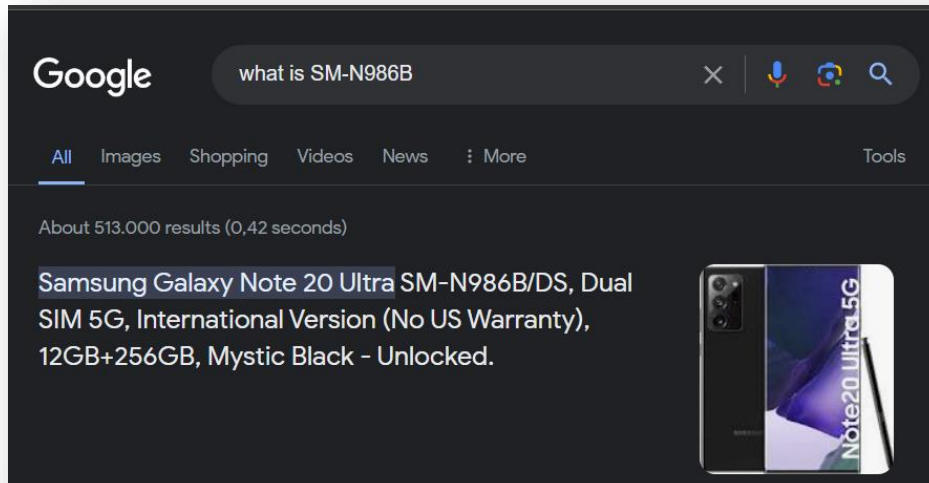


Figure 30 Internet search on Test1.jpg Image model.

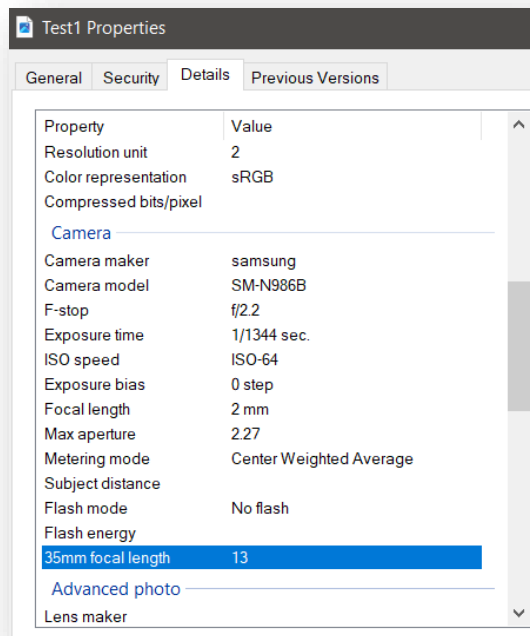


Figure 31 Test1.jpg verified details.

```

CASEY
  METADATA
  ANALYSER

Please note this tool is for educational purposes

The Folder specified is tes
stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\test.zip
Unknown file type
stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder1.jpg
Trying to access image C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder1.jpg
Deleting metadata...
Metadata deleted successfully.
stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder2.jpg
Trying to access image C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder2.jpg
Deleting metadata...
Metadata deleted successfully.
stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder3.jpg
Trying to access image C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder3.jpg
Deleting metadata...
Metadata deleted successfully.
stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder4.jpg
Trying to access image C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder4.jpg
Deleting metadata...
Metadata deleted successfully.
stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder5.jpg
Trying to access image C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder5.jpg
Deleting metadata...
Metadata deleted successfully.

```

Figure 32 Results from deleting metadata in tes folder.

CASEY

METADATA

ANALYSER

Please note this tool is for educational purposes

The Folder specified is tes
stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\test.zip
Unknown file type
stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder1.jpg
Trying to access image C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder1.jpg

Key	Value
Image Size	(4032, 3024)
Image Height	3024
Image Width	4032
Image Format	JPEG
Image Mode	RGB
Image is Animated	False
Frames in Image	1

stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder2.jpg
Trying to access image C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder2.jpg

Key	Value
Image Size	(4032, 3024)
Image Height	3024
Image Width	4032
Image Format	JPEG
Image Mode	RGB
Image is Animated	False
Frames in Image	1

stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder3.jpg
Trying to access image C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder3.jpg

Key	Value
Image Size	(4032, 3024)
Image Height	3024
Image Width	4032
Image Format	JPEG
Image Mode	RGB
Image is Animated	False
Frames in Image	1

stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder4.jpg
Trying to access image C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder4.jpg

Key	Value
Image Size	(4032, 3024)
Image Height	3024
Image Width	4032
Image Format	JPEG
Image Mode	RGB
Image is Animated	False
Frames in Image	1

stage: C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder5.jpg
Trying to access image C:\Users\cvdon\OneDrive\Documents\GitHub\Python-Scripts\Python-Scripts\A_Project\scripting_forensics\tes\Test_folder5.jpg

Key	Value
Image Size	(4032, 3024)
Image Height	3024
Image Width	4032
Image Format	JPEG
Image Mode	RGB
Image is Animated	False
Frames in Image	1

Figure 33 Results from tes folder after deletion.


```
Outputting
Output file path: Test6_File_output.csv
{1: '\x00', 2: (nan, nan, nan), 3: '\x00', 4: (nan, nan, nan), 5: 0, 6: nan}
Error exception: Location values cannot contain NaNs.
No Geo Data
```

Figure 34 NaN result.