

FILE TYPE IDENTIFICATION – A COMPUTATIONAL INTELLIGENCE APPROACH TO
DIGITAL FORENSICS

by

KARAMPIDIS KONSTANTINOS

Electrical Engineer, Technological Educational Institute of Crete, 1994

A THESIS

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF INFORMATICS ENGINEERING

SCHOOL OF APPLIED TECHNOLOGY

TECHNOLOGICAL EDUCATIONAL INSTITUTE OF CRETE

2015

Approved by:

Major Professor
Dr PAPAOURAKIS GEORGE

Copyright

KARAMPIDIS KONSTANTINOS

2015

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/4.0/>



Abstract

The rapid growth and use of digital devices (e.g. computers, android tablets and smartphones), made people vulnerable to cybercrimes. Dr. Debarati Halder and Dr. K. Jaishankar (2011) define cybercrimes as: "Offences that are committed against individuals or groups of individuals with a criminal motive to intentionally harm the reputation of the victim or cause physical or mental harm, or loss, to the victim directly or indirectly, using modern telecommunication networks such as Internet (Chat rooms, emails, notice boards and groups) and mobile phones (SMS/MMS)" [1]. For instance, one major and loathsome crime is child pornography. A child predator may try to hide evidence in a computer or any other digital device, by changing the file type. This could be easily done by altering the file extension, file signature or both. A digital forensic examiner on the other hand, uses forensic software to accurately identify the file types in order to determine which files may contain potential evidence. Nevertheless, current type recognition mechanisms are vulnerable to simple deceptions and even the most widely used commercial forensic software suites may not predict correctly an intentionally altered file. For instance, if someone changes file extension from .jpg to .doc, the forensic software will identify that the file type is changed. Nevertheless, if the file signature is changed as well in order to be related to a .doc file, the forensic software detection algorithm may show poor results. Another important field where file type identification must be quick and accurate is spam e-mail. Every day massive amount of spam e-mails are received and lot of time is spent to delete them. Unfortunately this is not the only disadvantage. Network bandwidth is taken, e-mail servers are slowing down and eventually an unexperienced end user may not be able to identify if the e-mail hides malicious content. These are only a few paradigms of the possible damage caused by an unsuccessful file type recognition. This master's thesis will

try to examine all possible practices of identifying a file type and propose a new method – in a digital forensics perspective - to identify a file type with high accuracy.

Table of Contents

Copyright	ii
Abstract	iii
List of Figures	vii
List of Tables	ix
Acknowledgements.....	x
Dedication	xi
Preface.....	xii
Chapter 1 - Introduction to Digital Forensics	1
1.1 Forensic Process	2
1.1.1 Data Collection	3
1.1.2 Data examination	4
1.1.3 Data Analysis	5
1.1.4 Report.....	6
1.2 Forensic Tools.....	6
Chapter 2 - File Type Identification.....	9
2.1 File Format.....	9
2.2 Extension based method	10
2.3 Magic bytes method.....	12
2.4 Content based method.....	22
Chapter 3 - Computational Intelligence to Digital Forensics	26
3.1 Statement of the problem.....	26
3.2 Delimitations – Data Mining Software	26
3.3 The dataset	32
3.4 Feature extraction	33
3.5 Feature selection	35
3.5.1 Correlation based Feature Selection (CFS).....	37
3.5.2 Best first as a search method.....	38
3.5.3 Genetic Algorithm as a search method	39
3.6 The Classifier – Multi Layer Perceptron (MLP)	41

3.7 Cross Validation of the training data	43
Chapter 4 - Results	45
4.1 Results using Best First as search method	45
4.2 Results using Genetic Algorithm as search method	46
4.3 Comparison of the two search methods	47
4.4 Restatement of the problem	50
4.5 Results on the new testing datasets	51
4.6 Comparison of the proposed method to the literature.....	56
Chapter 5 - Conclusions & Future Work	58
5.1 Conclusions.....	58
5.2 Future Work	59
References.....	60
Appendix A - Weka Implementation.....	65

List of Figures

Figure 1.1: Digital Forensic Areas of Interest	1
Figure 1.2: Phases of the forensic process	2
Figure 1.3: BackBox Linux distribution	8
Figure 1.4: Computer Aided Investigative Environment (CAINE) Linux distribution.....	8
Figure 2.1: Unhide extension of known file types in Windows	10
Figure 2.2: The file to be renamed.....	11
Figure 2.3: The renamed file.....	11
Figure 2.4: Extension mismatch and correct identification on Autopsy forensic software	12
Figure 2.5: The file signature of a .doc file. The magic bytes are in the rectangular box	13
Figure 2.6: File signature of a png image	15
Figure 2.7: Altering the file signature through a hex editor	15
Figure 2.8: Extension mismatch	16
Figure 2.9: Autopsy can't recognize the change.....	16
Figure 2.10: Autopsy can't see an extension mismatch.....	17
Figure 2.11: TrID	17
Figure 2.12: Analyze It!	18
Figure 2.13: Analyze file header and content	18
Figure 2.14: Toolsley online identifier	19
Figure 2.15: DROID	20
Figure 2.16: Exiftool.....	20
Figure 2.17: Falstaff correct identification	21
Figure 2.18: Falstaff wrong identification	22
Figure 3.1: GIF format details and header	27
Figure 3.2: PNG file structure.....	28
Figure 3.3 JPEG format details.....	29
Figure 3.4: The file structure of a pdf file.....	31
Figure 3.5: Byte Frequency Distributions for two jpg images.....	34
Figure 3.6: Byte Frequency Distributions for two png images.....	34
Figure 3.7: Byte Frequency Distributions for two gif images	34

Figure 3.8: Feature selection flowchart – Best First	39
Figure 3.9: Feature selection flowchart – Genetic Algorithm	40
Figure 3.10: A multilayer perceptron with two hidden layers	41
Figure 3.11: Flowchart of the proposed method	44
Figure 4.1: Comparison of search methods TP Rate	48
Figure 4.2: Comparison of search methods – Precision	49
Figure 4.3: Comparison of search methods – Recall	49
Figure 4.4: Accuracy comparison of the proposed method in altered images	55
Figure A-1: Importing training set and preprocess in Weka.....	65
Figure A-2: Selecting parameters for the proposed method	65
Figure A-3: Another way of selecting attributes using k-fold cross validation.....	66
Figure A-4: Classifier results	67
Figure A-5: Classifier results in forged jpg images	68
Figure A-6: Classifier results in forged png images	69
Figure A-7: Classifier results in forged gif images.....	70

List of Tables

Table 2.1: A list of some widely used file types and their file signatures	14
Table 3.1: Signature of a JPEG image	28
Table 3.2: Options for the fourth byte in jpeg header.....	29
Table 3.3: The most common JPEG markers, https://en.wikipedia.org/wiki/JPEG	30
Table 3.4: Caltech 101 Dataset	32
Table 3.5: Our Dataset	33
Table 3.6: Parameters of the Genetic Algorithm	40
Table 3.7: Parameters of the multilayer perceptron.....	42
Table 4.1: Remaining features after selection with Best First search method and CFS.....	45
Table 4.2: Confusion matrix – Best First - CFS – Training time 500 epochs	45
Table 4.3: Confusion matrix – Best First - CFS – Training time 1000 epochs	46
Table 4.4: Remaining features after selection with GA as search method and CFS	46
Table 4.5: Confusion matrix – Genetic Algorithm - CFS – Training time 1000 epochs.....	46
Table 4.6: Detailed Accuracy for Best First	47
Table 4.7: Detailed Accuracy for Genetic Algorithm.....	47
Table 4.8: The new dataset	50
Table 4.9: Confusion matrix – Identifying forged jpg images.....	51
Table 4.10: “Misclassified” pdf instances (jpg actual type)	51
Table 4.11: Actual confusion matrix – jpeg images	52
Table 4.12: Detailed Accuracy By Class – Our proposed method (in forged jpg images).....	52
Table 4.13: Confusion matrix – Identifying forged png images.....	52
Table 4.14: “Misclassified” pdf instances (png actual type)	53
Table 4.15: Actual confusion matrix – png images	53
Table 4.16: Confusion Matrix – Identifying forged gif images.....	54
Table 4.17: “Misclassified pdf” instances (gif actual type).....	54
Table 4.18: Actual confusion matrix – gif images.....	54
Table 4.19: Final Confusion Matrix of the proposed method.....	55
Table 4.20: The proposed method compared to the literature	56

Acknowledgements

I would like to thank the faculty and administrative staff of the Department of Informatics Engineering in Technological Institute of Heraklion, for their dedication and assistance during the postgraduate program “Informatics and Multimedia” through past year. I would specifically like to thank my thesis advisor, Dr. George Papadourakis for his valuable guidance and advices and thesis committee members Dr. Spiros Panagiotakis and Dr. Athanasios Malamos for their valuable input and assistance

Dedication

This master's thesis and the attendance of the postgraduate program would be inevitable if i haven't had the full support, understanding and inducement of my wife. Rightfully this thesis is 100% dedicated to her.

Thank you Argyro

Preface

Digital Forensics is a relatively new field in Computer Science. Although most people think that only a computer might be a cyber “weapon” this is not true. All electronics devices may hide possible evidence. One of the most important steps to Digital Forensics is the correct identification of a file type. Many times suspects try to hide evidence by changing the file type. In Chapter one, a small introduction to Digital Forensics is made and the standard forensic procedures, tools and software used by forensic examiners are presented. In Chapter two we present all possible methods of identifying a file, give examples by using well-known software tools and refer to the literature for other scientific proposals. In Chapter three we propose a new method of file type identification. Our method uses evolutionary algorithms such as Genetic Algorithms for feature extraction and a multilayer perceptron for classification. In Chapter four we present the results of this method and finally in Chapter 5 there are the conclusions of this thesis along with thoughts of future work in the specific scientific area.

Chapter 1 - Introduction to Digital Forensics

Computers have had increasing roles in all aspects of human life. Especially when they became small and cheap enough to be in everybody's house, mainly from the early 80s. Unfortunately they became also a convenient tool for criminal acts as well. This development has led to the rise of digital forensics, the uncovering and examination of evidence located on all electronics with digital storage, including computers, cell phones, and networks. Digital forensics can be divided into four main areas of interest:

- Computer Forensics
- Network Forensics
- Mobile device Forensics
- Database Forensics

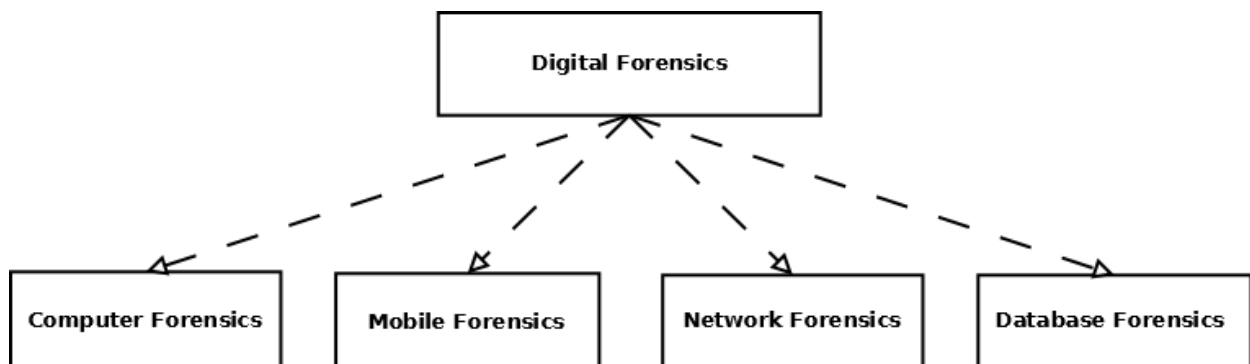


Figure 1.1: Digital Forensic Areas of Interest

1.1 Forensic Process

Due to the risk of losing potential evidence there is the need to respect a specific procedure when trying to discover hidden evidence in electronic devices. According to National Institute of Standards and Technology (NIST) [2], the forensic process has four major phases:

- Data Collection
- Examination
- Analysis
- Report

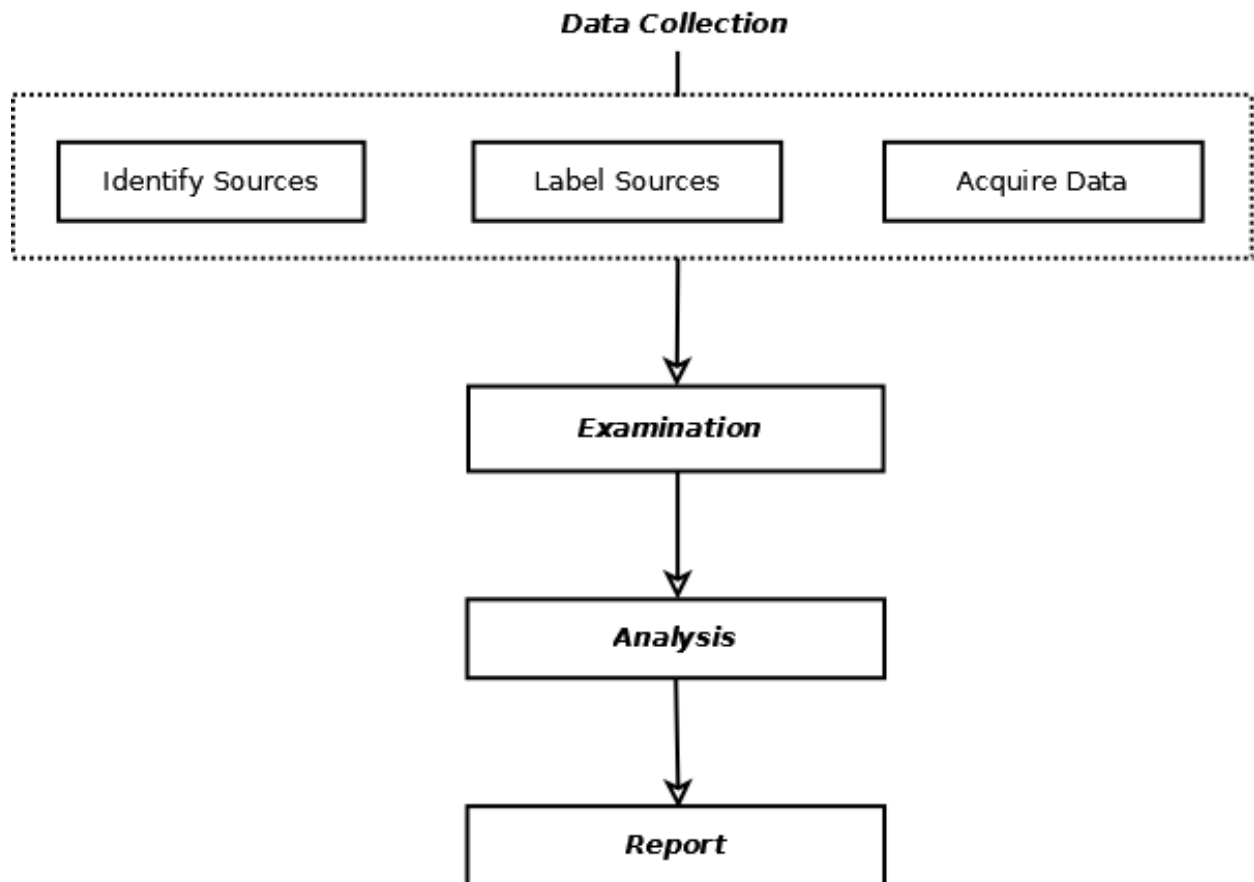


Figure 1.2: Phases of the forensic process

1.1.1 Data Collection

Digital sources which may hide potential evidence are numerous due to the increasingly use of technology for professional or amusement purposes. Potential evidence could be found on desktop computers, laptops, servers or network storage devices. These systems have internal drives such as hard disks (HDD) and ports like Universal Serial Bus (USB) to which external data storage media and devices can be attached. These external media could be an external hard drive, a USB flash drive, memory cards, optical discs etc. We must also take into consideration that evidence may hide into RAM, clipboard or network connection (volatile data) and for this reason a system shut down or reboot, may lead to their extinction. Furthermore, besides of computers or computer-related devices, data may be in portable devices like cell phones or digital cameras. A forensic investigator must have access to the crime scene, examine the area and identify all possible sources of data although occasionally it is not possible to collect data from a primary data source e.g. the past network activity of a device. This information may be in ISP's log files and a certain court order may needed in order to have access to these data. Moreover we must be very careful when handling and examining devices due to strict law in privacy matters.

Every time a potential source is identified, it must be uniquely labeled, recorded, and collected. It is essential to begin collecting data from volatile sources. As stated earlier volatile data could be the content of the RAM or the network activity etc. There are a lot of live forensic tools (open source or proprietary) which can retrieve these information. In general, the data gathered could be information about running processes, loaded libraries, used resources, logged on users, network connection status, open ports etc.

After capturing the volatile data, we can proceed with the non-volatile ones. A copy of the whole content of the device must be extracted. We must make clear that we just not only copy the

contents of a device as its metadata i.e. hashes and timestamps (modification, access, and creation times) would be lost. Instead we must use forensic imaging tools or commands in order to maintain these crucial information. Forensic imaging is done with special forensic tools. The forensic investigator applies a write blocker (hardware or software) in order to avoid modification of the data and takes a forensic image of the device while keeping the metadata and compressing the all the empty blocks. For example two commands -which require only minimal resources to run- used for this reason are dd or dcfldd (used mainly on Linux operating systems). For reasons of integrity and authenticity, every time a raw image of the data is acquired its message digest is calculated both for the original and forensically copied data, then by comparing the digests we make sure that they are the same and not tampered.

It is essential to say that, the forensic examiner should make a master copy and a working copy of the files. The examiner then, will work with the working copy without concerned of wrong handlings or alteration.

1.1.2 Data examination

After data has been collected, the next phase is to examine them. A raw image of a hard drive has many gigabytes or terabytes of files. The task of identifying the files that contain information of interest –potential evidence- is a difficult task. Furthermore, potential evidence may contain unnecessary information that should be filtered. This is done by using forensic tools and techniques in order to reduce the amount of data that has to be examined thoroughly. We apply text and pattern searches to identify relevant data and try to determine the type of contents of each data file. Knowledge of data file types is used to exclude files that are of no interest to the

investigation and to focus only to these that may have information to reveal. Windows registry is another worth looking place for extracting evidence as it can reveal information about the system, the users, and the software installed or accessed.

Besides the huge amount of data, a forensic investigator sometimes have to deal with encrypted data as well. Users might encrypt individual files, folders, or partitions so that no other can access their contents without the use of a decryption key. It is very easy to identify an encrypted file, but it is very difficult to bypass the encryption without having the encryption key. For this reason, the examiner must look carefully to find encryption tools that are installed in the device, identify the encryption method and finally see if the encryption key is stored somewhere in the raw image.

1.1.3 Data Analysis

Afterwards the data examination, the subsequent step is to perform analysis of the remaining extracted data. There are many tools available that analyze different types of data. Forensic examiners must be aware of the value of using system times and file times. If the examiner knows when an incident happened or when a file was created or modified, it can be critical to forensic analysis. In other words the examiner is able then to reform a timeline of actions taken place. In the case that multiple tools are used to evaluate the data, the analyst should fully understand how each tool works and how it extracts and displays file metadata (file creation time – MAC). As already said, write-blockers (hardware or software) must be used to prevent these tools from altering the creation times. However, write-blockers cannot prevent the operating system from caching the changes in memory. As a result of this the operating system might report

the cached creation times instead of the actual times. For all these reasons, the forensic examiner should carefully choose a MAC viewing method and rely on special tools that can generate forensic timelines based on event data, through a graphical interface for event visualization and analysis. Forensic data analysis also involves data from other sources, such as the network traffic, network monitoring or applications.

1.1.4 Report

The final phase of the forensic process is reporting. The report is often written and sometimes – when comes to a court room – verbal. The report contains all the information about the examiner, date and time the data were collected, the tools and the methods were used to evaluate the data and last the conclusions. The forensic examiner must be accurate when describing an event, give a structured justification of the conclusions he/she came up to and leave no margin of doubt. If an event has more than one possible explanations, each should be specified in detail in the report. Finally the forensic examiner must be able to accurately justify his/hers scientific findings, while being clear and comprehensible when presenting the facts to an unskilled audience e.g. a court room.

1.2 Forensic Tools

A lot of forensic tools have been created over the last years. There are small programs that deal with specific forensic actions or whole forensic packages with which a forensic examiner can work with and deal with the most difficult cases. In this paragraph we will present the most used ones. First, we have to mention that there are Linux live distributions offering open source forensic tools. The most major are:

- Kali Linux : a Debian-based distribution with a collection of security and forensics tools [3].
- CAINE (Computer Aided INvestigative Environment) is an Ubuntu-based GNU/Linux live distribution. It offers a complete forensic environment [4].
- DEFT (Digital Evidence & Forensic Toolkit) is a customised distribution of the Ubuntu live Linux CD. It offers some of the best open-source applications dedicated to incident response and computer forensics [5].
- BackBox is a Debian-based security distribution designed for penetration testing and forensic investigations [6].
- NetSecL is a security-focused distribution and live DVD based on openSUSE [7].
- Parrot Security OS is a security oriented operating system designed for penetration testing, computer forensics, cryptography, steganography etc. The distribution is based on Debian [8].

Besides Linux distributions there are a lot (proprietary or open source) standalone forensic suites that help a forensic examiner. The most known and used are:

- Encase by Guidance Software [9]
- Sleuth Kit - Autopsy [10].
- FTK Access Data [11].
- Oxygen Forensics [12].

Of course there are lot more of other forensic software, some of which will be referred and used in the next chapter.



Figure 1.3: BackBox Linux distribution



Figure 1.4: Computer Aided Investigative Environment (CAINE) Linux distribution

Chapter 2 - File Type Identification

2.1 File Format

A file format is the blueprint of a file. It tells the processing device (e.g. a computer) how data within a file are organized and specifies the way the information is encoded in a digital storage medium. File formats may be either proprietary e.g. .dwg for an Autocad file, free which is not burdened by any copyrights, patents or other restrictions, or open which anyone can read and study but it may be burdened by restrictions on use. One popular method used by many operating systems, including Windows –which is the most popular operating system among computer end users- is to determine the format of a file based on the end of its name, the letters following the final period. This is known as the filename extension. For example, text documents are identified by names that end with .doc (or .docx), and PNG images by .png. In the original FAT filesystem, file names were limited to an eight-character identifier and a three-character extension, known as an 8.3 filename (also called a short filename or SFN). Many formats still use three-character extensions even though modern operating systems and applications no longer have this constraint. Some file formats are designed for very particular types of data e.g. doc or docx stands for document files, jpg declares a compressed picture etc., while png extension relates to images using lossless data compression. Nevertheless, other file formats are intended for storage of several different types of data: the flash video (flv, f4v) format can act as a container for video and audio from Adobe Systems. There are thousands of file formats and the list is getting bigger day by day. Since there is no standard list of extensions and given the fact that more than one format can use the same extension, this could lead to confuse both the operating system and end users. From a user's perspective this confusion might be just ignorance or could hide deceit.

This master's thesis will endeavor to find out which methods of file type identification were suggested by the scientific community and to propose a new technique of correctly identifying hidden images.

2.2 Extension based method

This method is the simplest one but it is also the most easy to be spoofed. All files have an extension (in Windows operating system) and this extension associates the file with the appropriate software. For example .doc or docx extension stands for Microsoft's Word, .pdf stands for Adobe's Reader etc. By default -for security reasons in Windows operating system- extensions are hidden but this can easily change from control panel (figure 1).

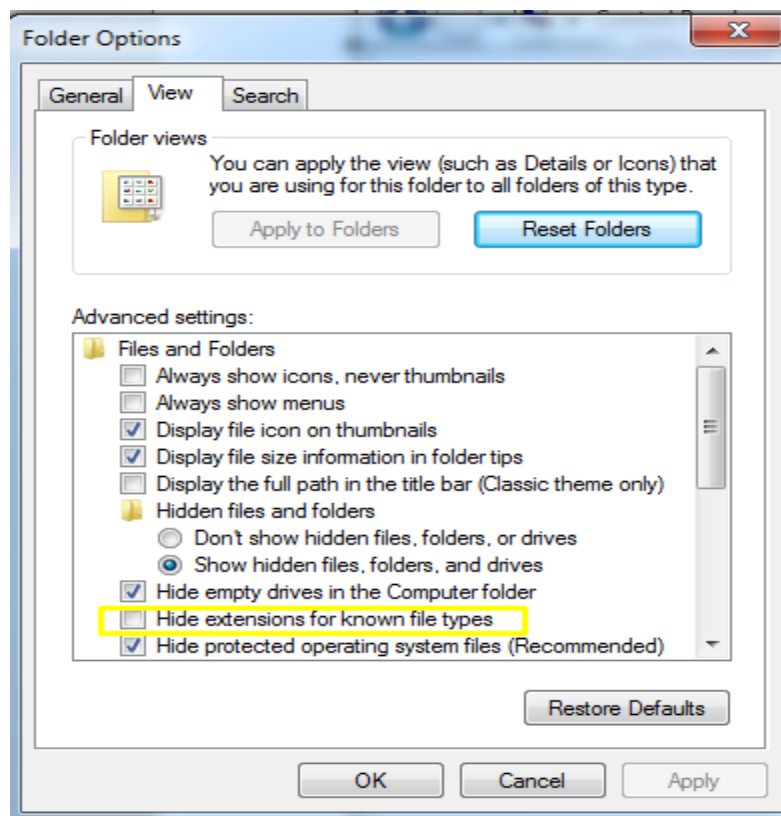
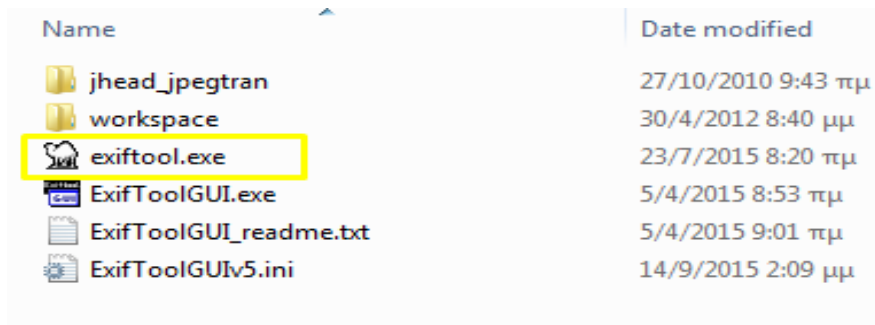


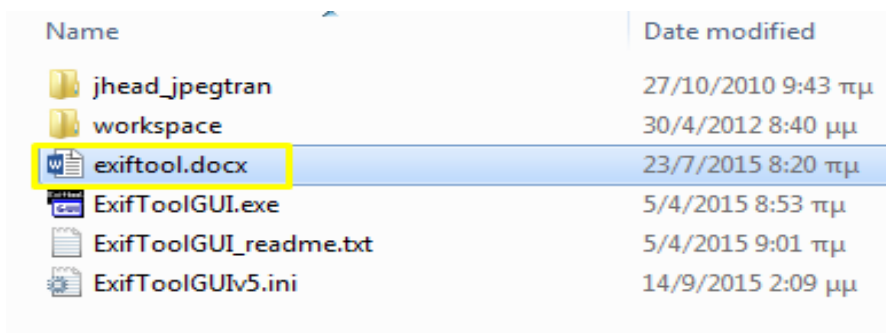
Figure 2.1: Unhide extension of known file types in Windows

The problem is that anyone can change the extension of the file by a simple renaming and this results to the change of the file type. For example let us consider an executable file e.g. exiftool.exe (figure 2.2) and try to change its extension by renaming it to exiftool.docx (figure 2.3)



Name	Date modified
jhead_jpegtran	27/10/2010 9:43 πμ
workspace	30/4/2012 8:40 μμ
exiftool.exe	23/7/2015 8:20 πμ
ExifToolGUI.exe	5/4/2015 8:53 πμ
ExifToolGUI_readme.txt	5/4/2015 9:01 πμ
ExifToolGUIv5.ini	14/9/2015 2:09 μμ

Figure 2.2: The file to be renamed



Name	Date modified
jhead_jpegtran	27/10/2010 9:43 πμ
workspace	30/4/2012 8:40 μμ
exiftool.docx	23/7/2015 8:20 πμ
ExifToolGUI.exe	5/4/2015 8:53 πμ
ExifToolGUI_readme.txt	5/4/2015 9:01 πμ
ExifToolGUIv5.ini	14/9/2015 2:09 μμ

Figure 2.3: The renamed file

As we can see it is very easy for someone to intentionally change the file's extension and try to fool forensic examiners, in order to hide possible evidence. On the other hand a forensic examiner cannot rely on the information a file extension gives. This particular spoofing method is very easy to be detected by forensic software such as Encase or Autopsy.

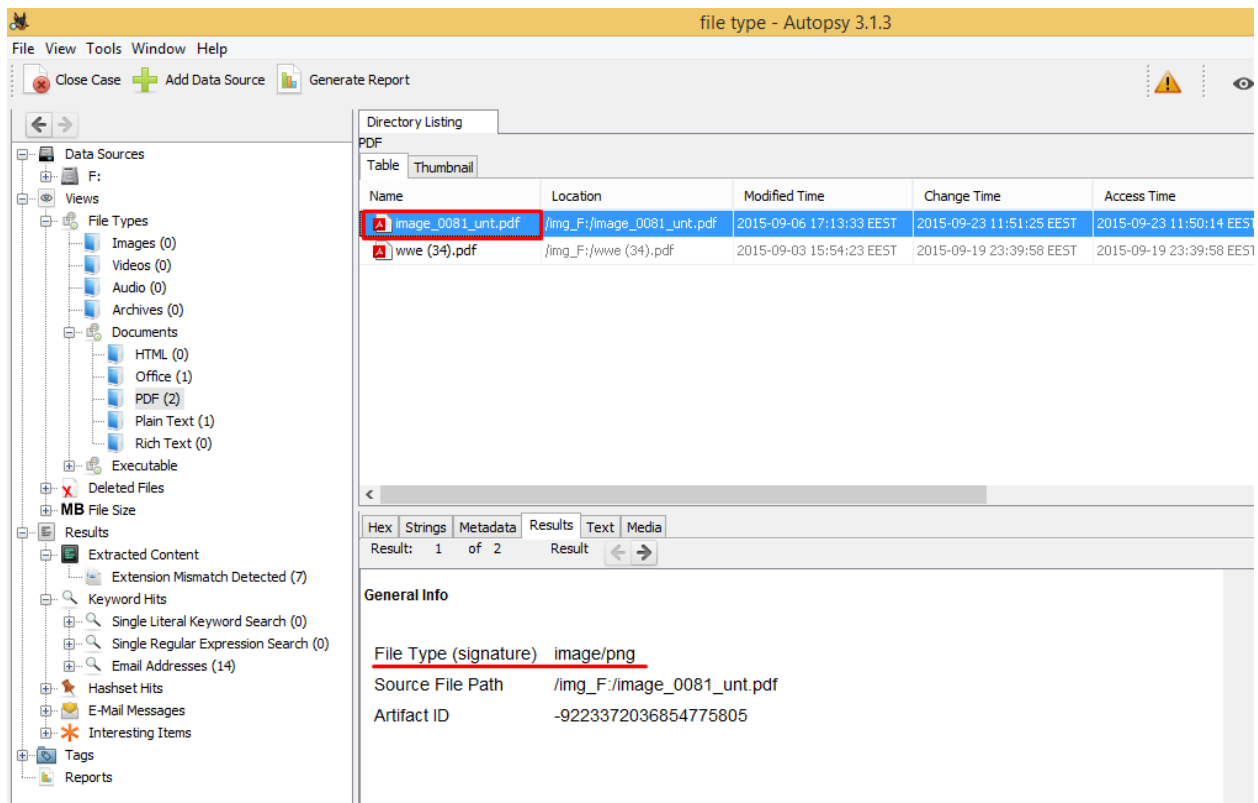


Figure 2.4: Extension mismatch and correct identification on Autopsy forensic software

2.3 Magic bytes method

The second method of file type identification is based on the magic bytes. These are some predefined signatures and they can be found on file's header. A file header is the first portion of a computer file that contains metadata. Metadata may enclose information about the content, quality and condition of the file. The file header also contains necessary information for the corresponding application to recognize and understand the file. Magic bytes may also include some extra information regarding the tool and the tool's version that is used to produce the file.

Table 2.1: A list of some widely used file types and their file signatures

File Type	Signature
DOC	D0 CF 11 E0 A1 B1 1A E1
FLV	46 4C 56 01
PDF	25 50 44 46
JFIF, JPE, JPEG, JPG	FF D8 FF E0 xx xx 4A 46 49 46 00
MP3 audio file	49 44 33
PNG	89 50 4E 47 0D 0A 1A 0A
RAR (v5) compressed archive file	52 61 72 21 1A 07 01 00
MS Windows/DOS Executable File (EXE)	4D 5A
GIF87a	47 49 46 38 37 61
GIF89a	47 49 46 38 39 61

There are several thousands of file types for which magic bytes are defined and there are multiple lists of magic bytes that are not completely consistent. Since there is not any standard for what a file may contain, the creators of a new file type usually include something to uniquely identify their file type. It is common that some programs or their developers may never put any magic bytes at the beginning of the file header. This approach can be also deceived. Altering the magic bytes of a file is a much harder way to defeat the true file type detection than the extension renaming, but the result is the same, i.e. the file type is not accurately recognized. In figure 2.6 there is a png image opened in a hex editor and we can see the magic bytes in the red rectangular box.

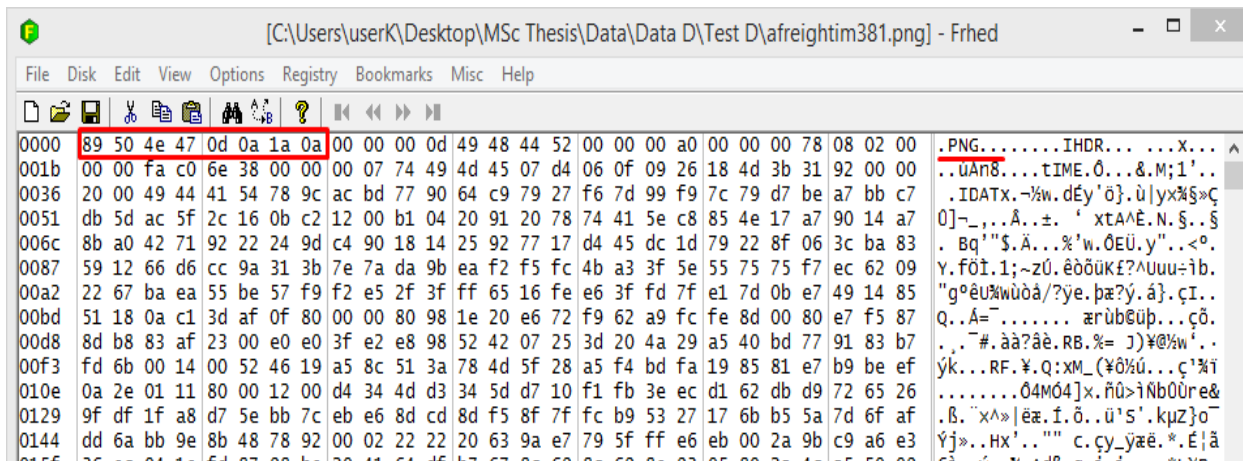


Figure 2.6: File signature of a png image

If we change the first bytes to FF D8 FF E8 xx xx 4A 46 49 46 00, the file from a png image will change to a jpeg image.

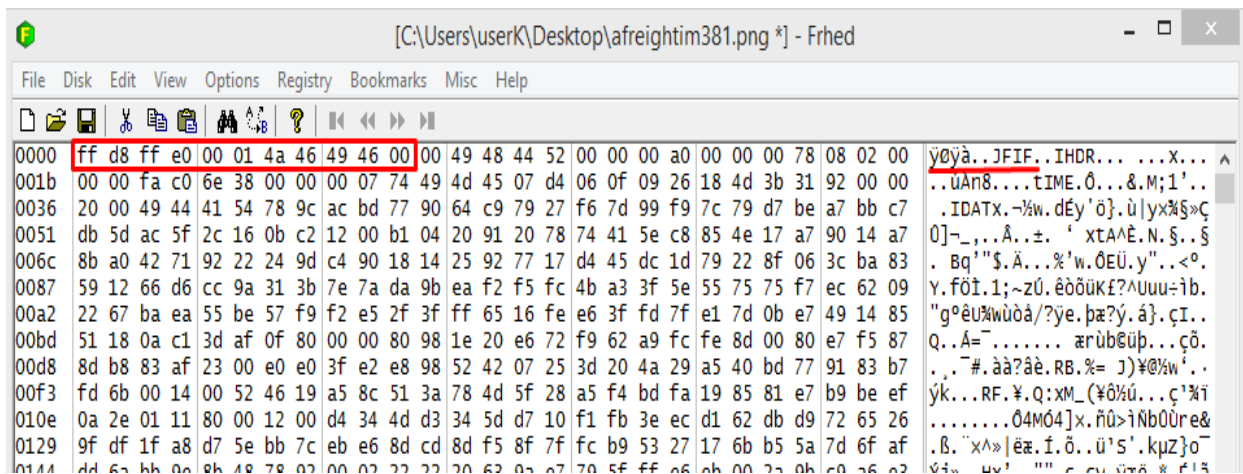


Figure 2.7: Altering the file signature through a hex editor

If we only change the file signature and keep the correct extension, the forensic software will highlight the file as a mismatch between extension and signature.

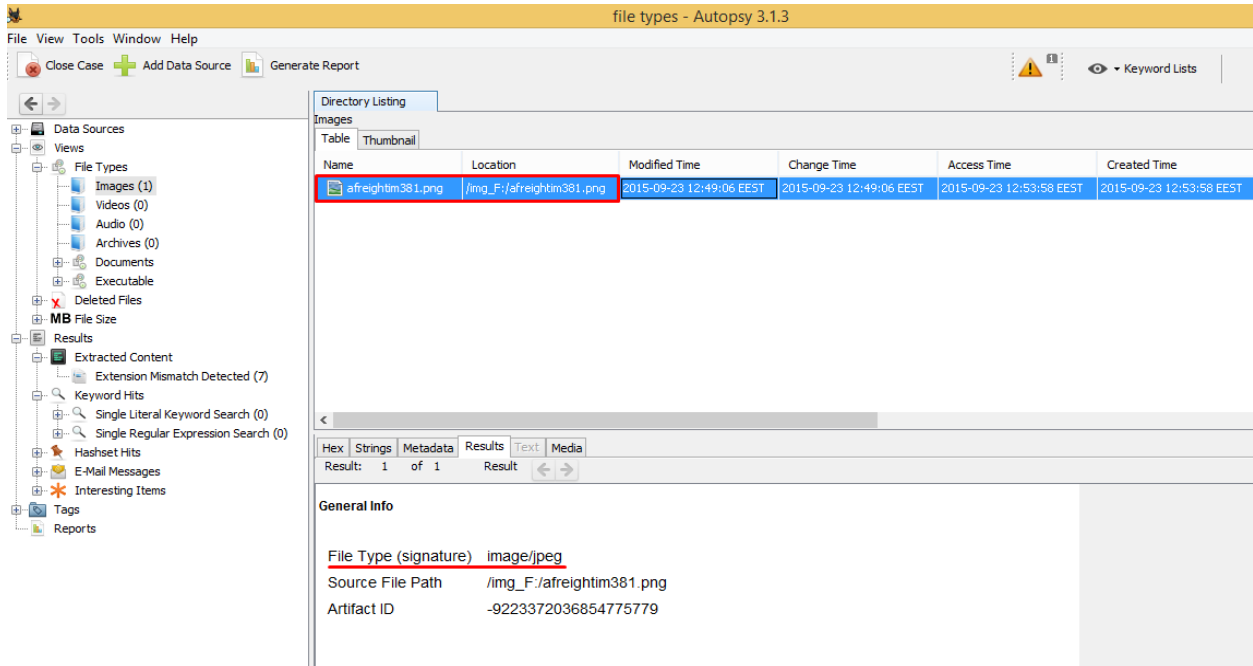


Figure 2.8: Extension mismatch

Subsequently if we change both extension and signature, Autopsy cannot recognize the deception.

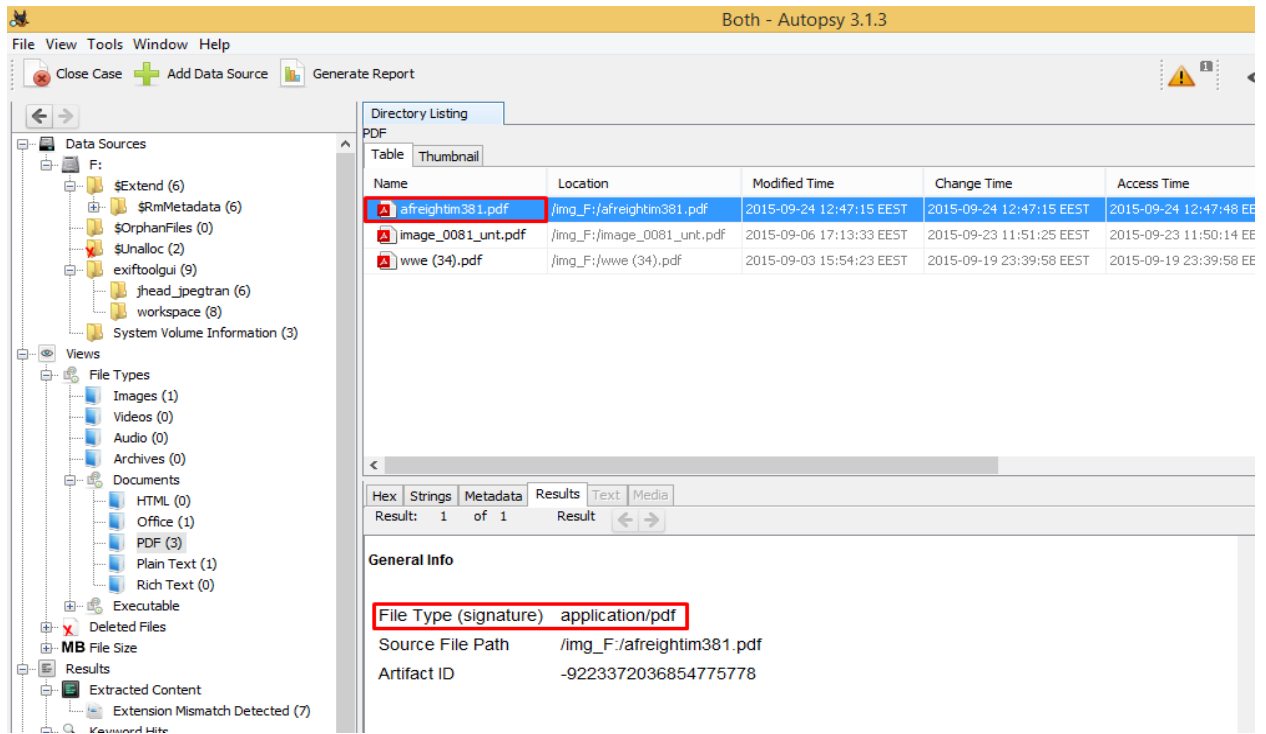


Figure 2.9: Autopsy can't recognize the change

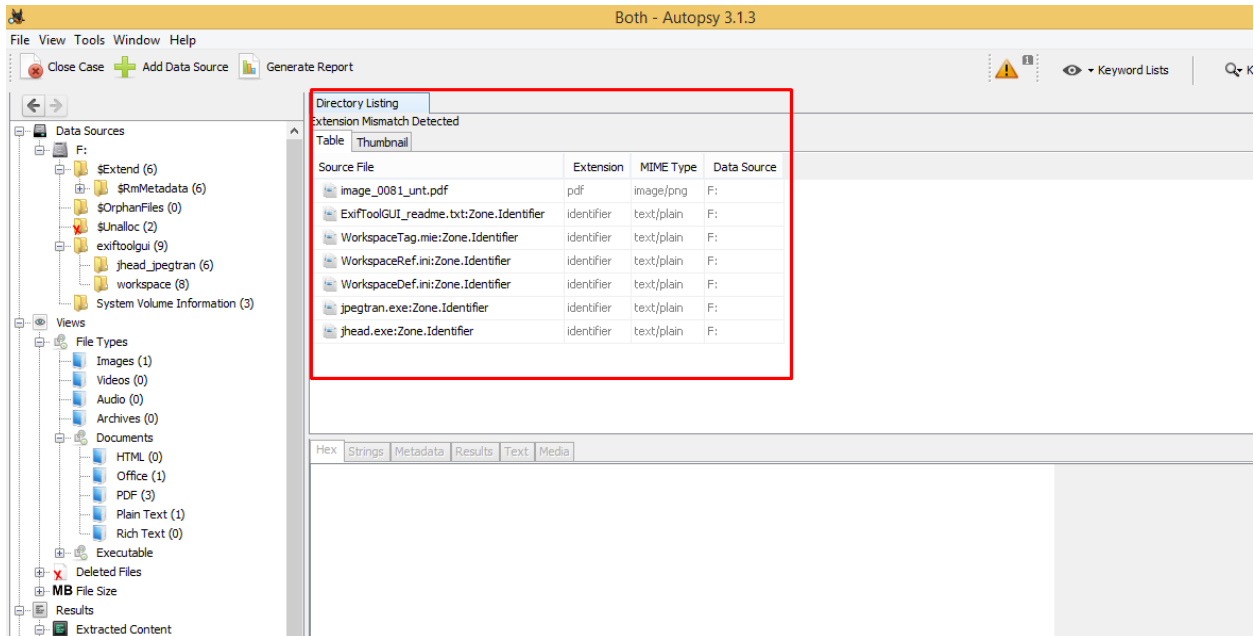


Figure 2.10: Autopsy can't see an extension mismatch

Besides Autopsy and Encase which are specialized forensic software, there are a lot of open source tools like TRiD [14], AnalyzeIt [15], ExifTool [16], Toolsley [17] (an online identifier) and DROID [18]. We have created a document named holiday.doc and then changed both the file extension and magic numbers to a jpeg image. If we check its file type with Trid the result is:

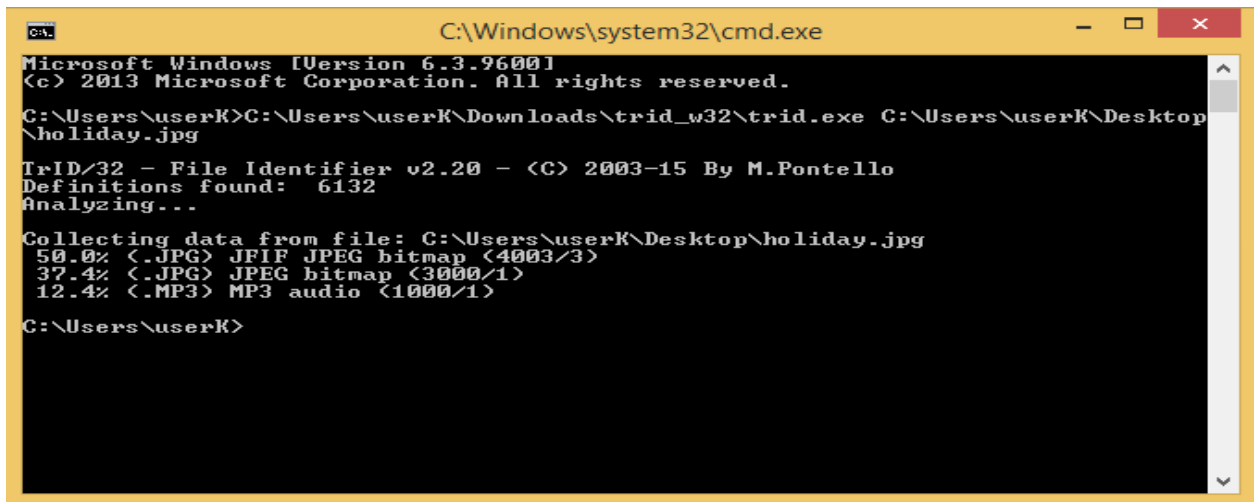


Figure 2.11: TrID

The result for the same file in AnalyzeIt is:

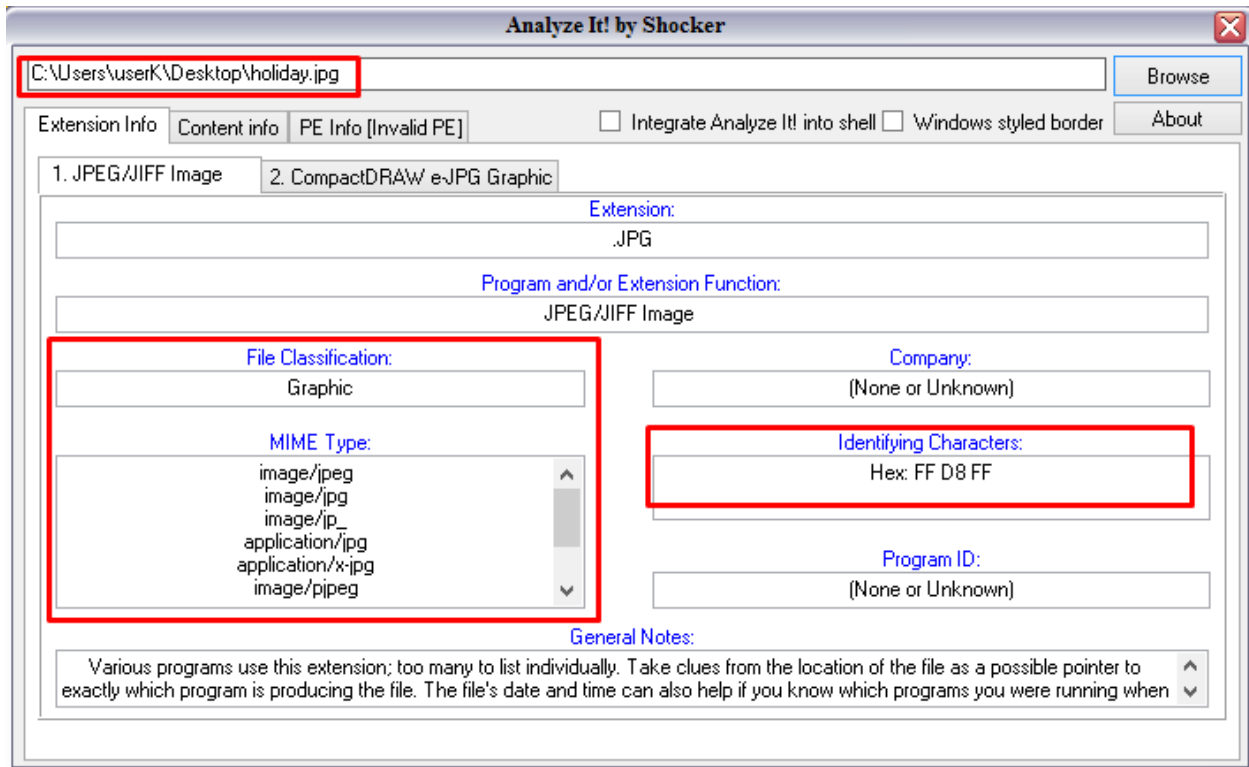


Figure 2.12: Analyze It!

In this software there is an option to check file type with a content based method and as we see the program show that we have an image file, which is wrong.

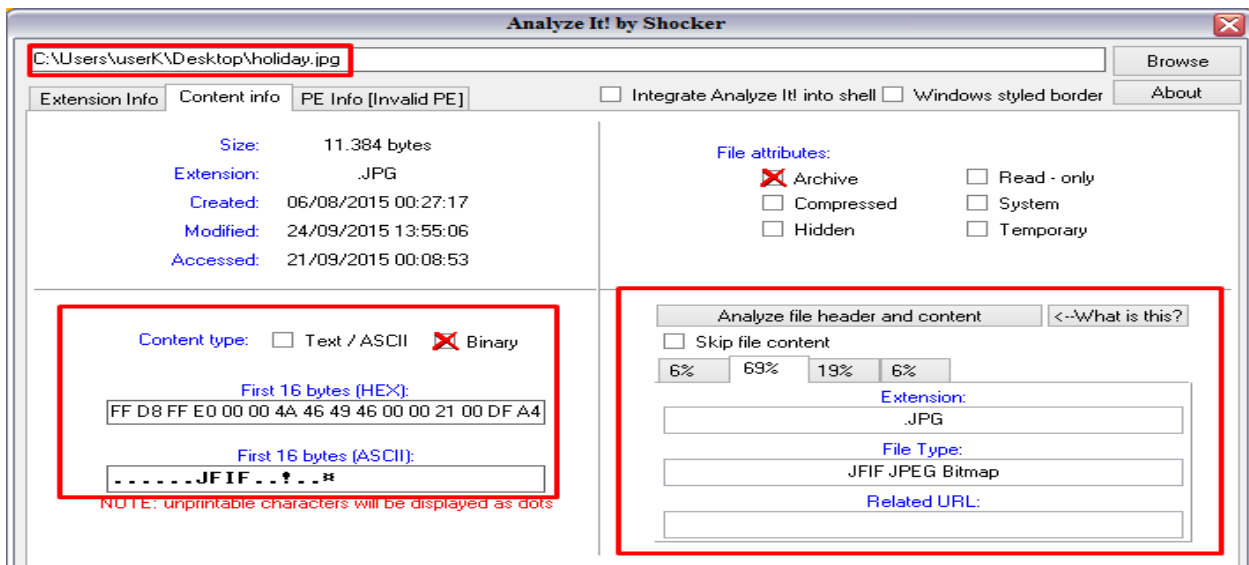


Figure 2.13: Analyze file header and content

Toolsley also recognizes the file as a jpeg image:

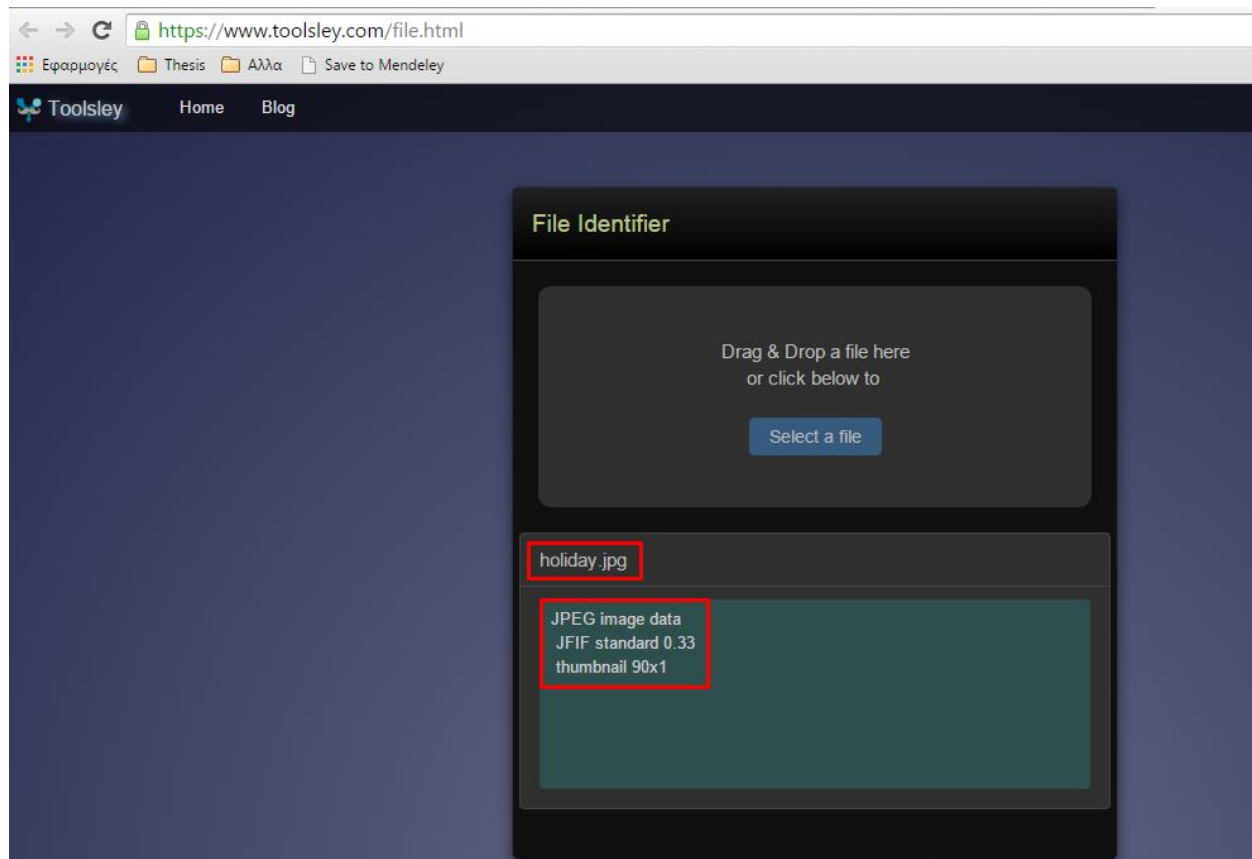


Figure 2.14: Toolsley online identifier

DROID (Digital Record Object Identification) is a software from the UK National Archives, which relies on PRONOM, the National's Archive Registry of file format information. The test on the same file – holiday.jpg – showed:

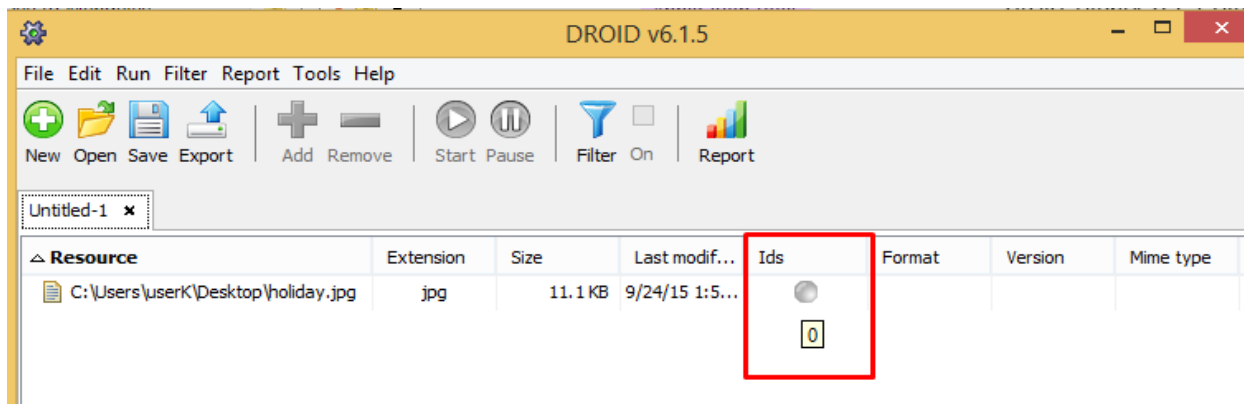


Figure 2.15: DROID

The program tells us that the file has an extension of .jpg but the grey dot under IDS column shows that it can't identify which format the file is.

Exiftool shows that there is a jpeg format error, but it does recognizes the file as a jpeg image when it's not.

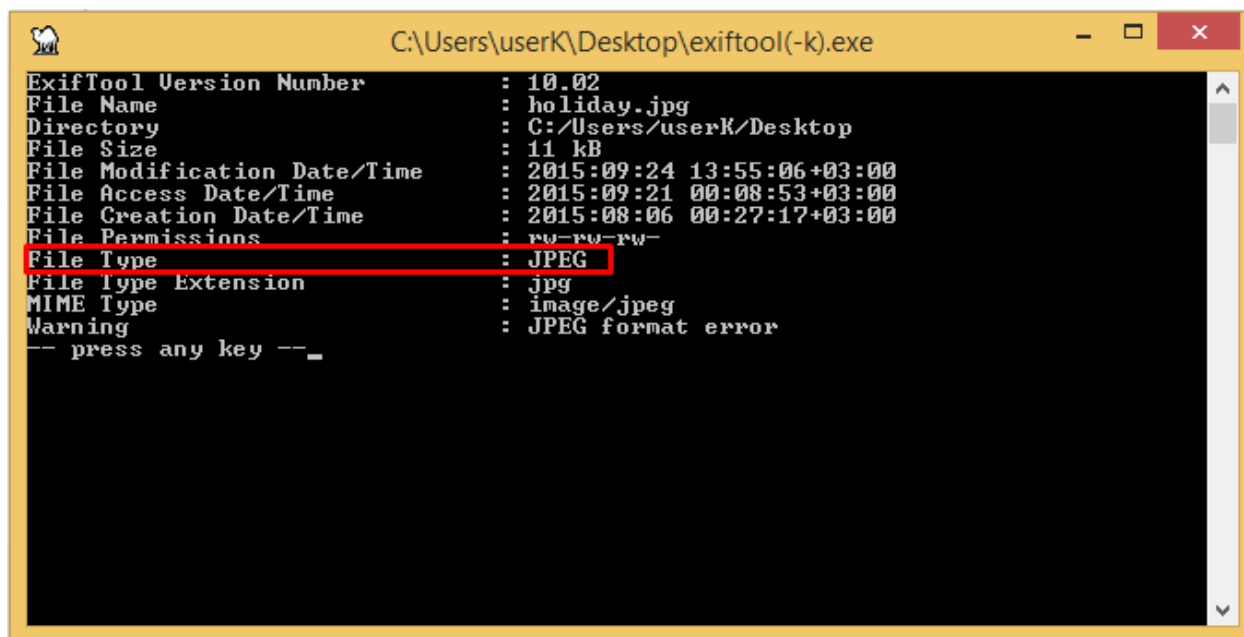


Figure 2.16: Exiftool

In all above cases there is a wrong classification of the file or the software understands that there is something wrong but it cannot determine the correct file format. If this file was potential evidence it would be likely lost.

Only Falstaff [19], recognizes the file correctly but there are two disadvantages

- the probability is 97% (in this case but in other paradigms it does not work well)
- it is an online tool which is not convenient for a forensic examiner.

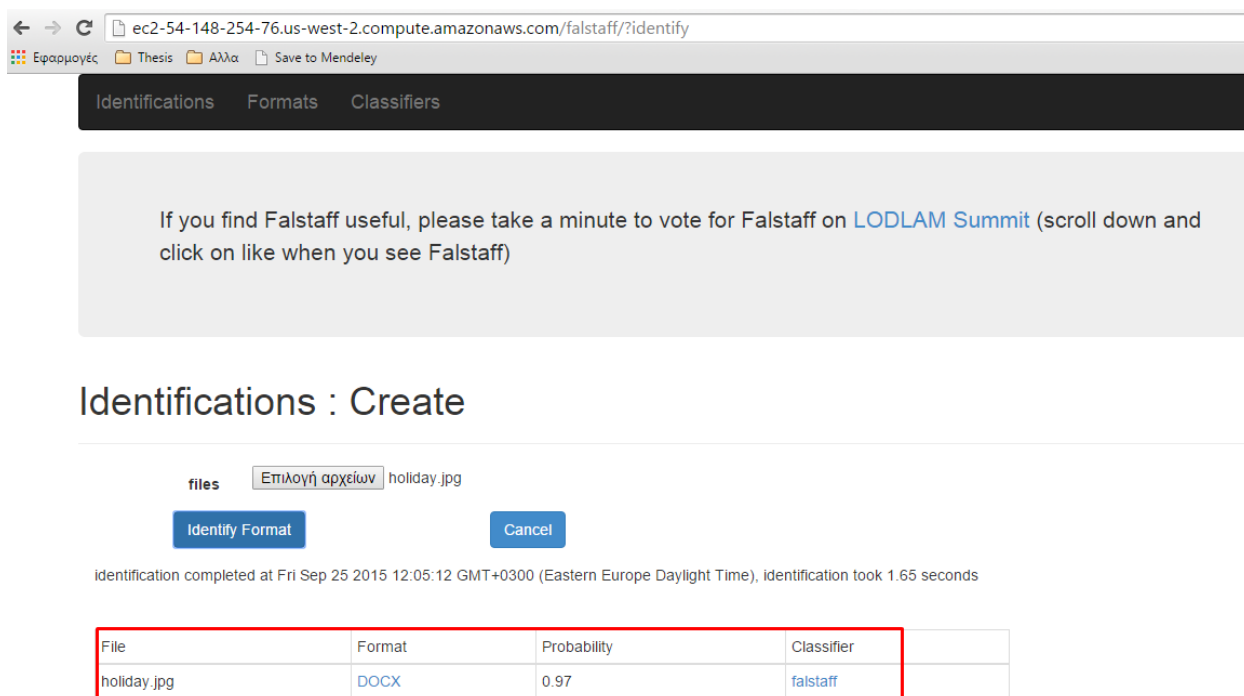


Figure 2.17: Falstaff correct identification

In order to test this tool again we tried another file which has been changed from png image to pdf but the results were poor.

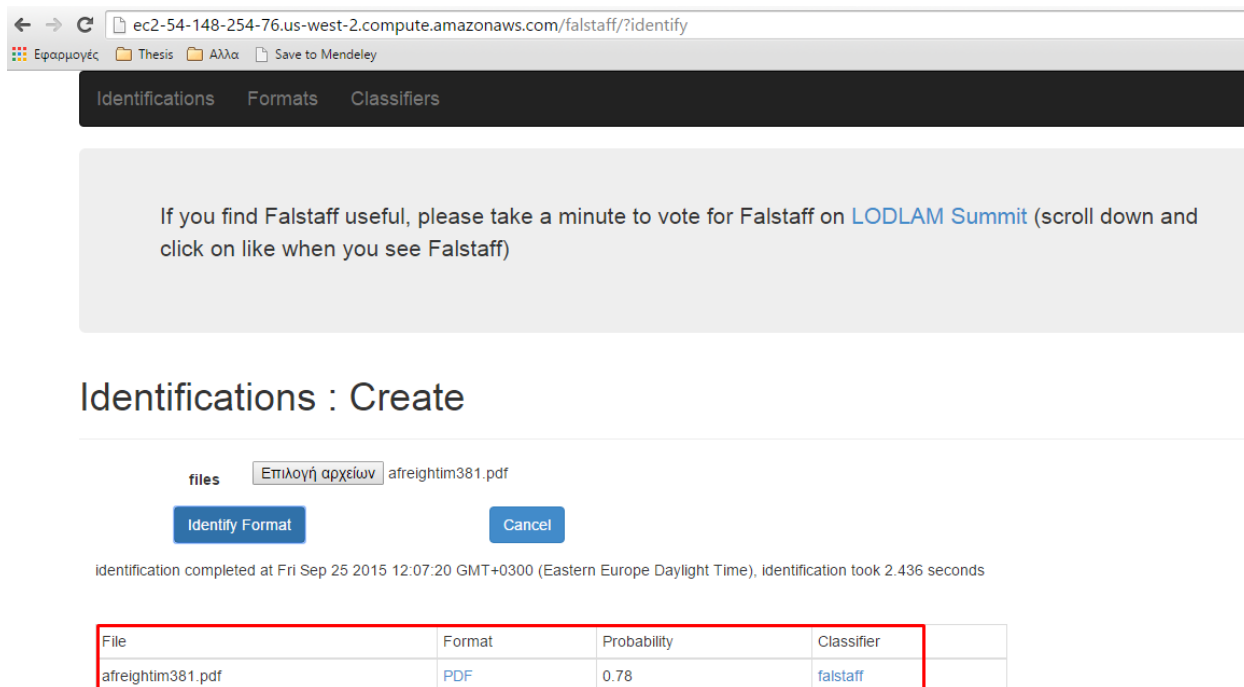


Figure 2.18: Falstaff wrong identification

We have to mention that this online tool uses machine learning techniques, multiple file features and novel signatures computed from file format samples.

2.4 Content based method

The third method of file type detection is to deliberate the file contents and use statistical modeling techniques. It is a new and promising research area and it is probably the only way to determine the forged file types with good results. It can reveal the malicious file types that their contents do not match with their claimed types. The contents of a file are a sequence of bytes and a byte has 256 unique permutations (0~255). Thus, counting the occurrence of byte patterns that is often referred as byte frequency distribution gives distinguishable patterns to identify file types.

There are many content-based file type identification schemes that use byte frequency distribution to build the representative models for file type and use any statistical and data mining techniques to identify file types.

McDaniel and Heydari were the first who actually proposed a way for content-based file type detection [20], [21]. They proposed three different algorithms for the content-based file type detection: Byte frequency Analysis (BFA), Byte Frequency Cross-correlation (BFC), and File Header/Trailer (FHT) analysis. These algorithms were used to produce a “fingerprint” of each file. Since every file type has a similar “fingerprint” with another file of the same type, the produced “fingerprint” is compared to the known one and find the true file type. The accuracy varied from 23% to 96% depending upon which algorithm was used.

Li et al. [22] made a few changes on the McDaniel and Heydari's method, in order to improve its accuracy. They stated that it is very difficult to produce one single descriptive model that accurately represents all members of a single file type class. Instead they proposed to compute a set of centroid models and use clustering to find a minimal set of centroids with good performance while the use of more pattern data is necessary. This approach resulted to 82% accuracy (one centroid), 89.5% accuracy (multi-centroid) and 93.8% accuracy (more exemplar files).

Dunham et al. [23] used neural networks to classify 10 different file types from a dataset of 760 files and achieved 91.3% accuracy. Karresand and Shahmehri [24] proposed a method based on data fragments. In general they used Byte Frequency Distribution (BFD) and especially the mean and standard deviation to model the file types. Like et al. [25] used the BFD along with a Manhattan distance comparison to detect whether the examined file is executable or not. Moody and Erbacher [26] used Statistical Analysis for Data type Identification (SADI) which included

average, distribution of averages, standard deviation, distribution of the standard deviations, kurtosis and distribution of byte values. They used fragments of 200 files as a dataset of 8 known file types, which resulted to a 74.2% accuracy.

Calhoun and Coles [27] used also a statistical method and specifically Fisher's linear discriminant to a dataset of 100 fragments of 2 different file types and achieved an accuracy of 60.3 – 86% (depending which sequence of bytes was examined). Amirani et al. [28] used the Principal Component Analysis (PCA) and unsupervised neural networks for the automatic feature extraction. The classifier they used was a five layer perceptron (MLP), achieving an accuracy of 98.33% which was the best so far.

Cao et al. [29] used Gram Frequency Distribution and vector space model with results of 90.34% accuracy. Ahmed et al. [30] proposed two very interesting methods. Primary they used the cosine distance as a similarity metric when comparing the file content. Subsequent they decomposed the identification procedure into two steps by taking the divide and conquer: in the first step, the similar files in terms of byte pattern frequencies were grouped into several clusters. In the next step, the cluster which contained different file types was fed to the neural network in order for improved classification. They used 2000 files of 10 file types as a dataset and achieved an accuracy of 90.19%. Ahmed et al. [31] also proposed two new techniques to reduce the classification time. The first method is a feature selection technique and the K-nearest neighbor (KNN) classifier was used. The second method is the content sampling technique, which uses a small portion of a file to obtain its byte-frequency distribution.

Amirani et al. [32] proposed an improved version of their first approach by using an SVM classifier and finally succeeded to raise the accuracy of the method up to 99.16% for a whole file.

Finally, Evensen et al. [33] used an n-gram analysis with naïve Bayes classifier to a large dataset of 60000 files (6 file types) with very good results of 99.51% topmost.

Chapter 3 - Computational Intelligence to Digital Forensics

3.1 Statement of the problem

As mentioned in the previous two chapters the problem relies on the modification of file's signature and its extension. In this case forensic tools or other software cannot always identify correctly the true file type, which would be crucial if these files were potential evidence in a court room. We will propose a method using computational intelligence techniques which will:

- train a classifier to identify the correct file type
- reveal the correct type if the file is altered

3.2 Delimitations – Data Mining Software

It is necessary to say that due to thousands of known file types, this research is inevitable to cover all file types comprehensively. It is also important to declare that we have focused only in images and portable documents, due to their significance to Digital Forensics. More specific, this thesis only included JPEG, PNG, GIF (not animated) and PDF files. Furthermore, we examined only whole files and not fragments of files.

Graphics Interchange Format (GIF) is a creation of CompuServe and is used to store multiple bitmap images in a single file for exchange between platforms and systems. Due to Lempel-Ziv-Welch (LZW) data compression, the format became very popular as LZW could reduce the image size without degrading the visual quality.

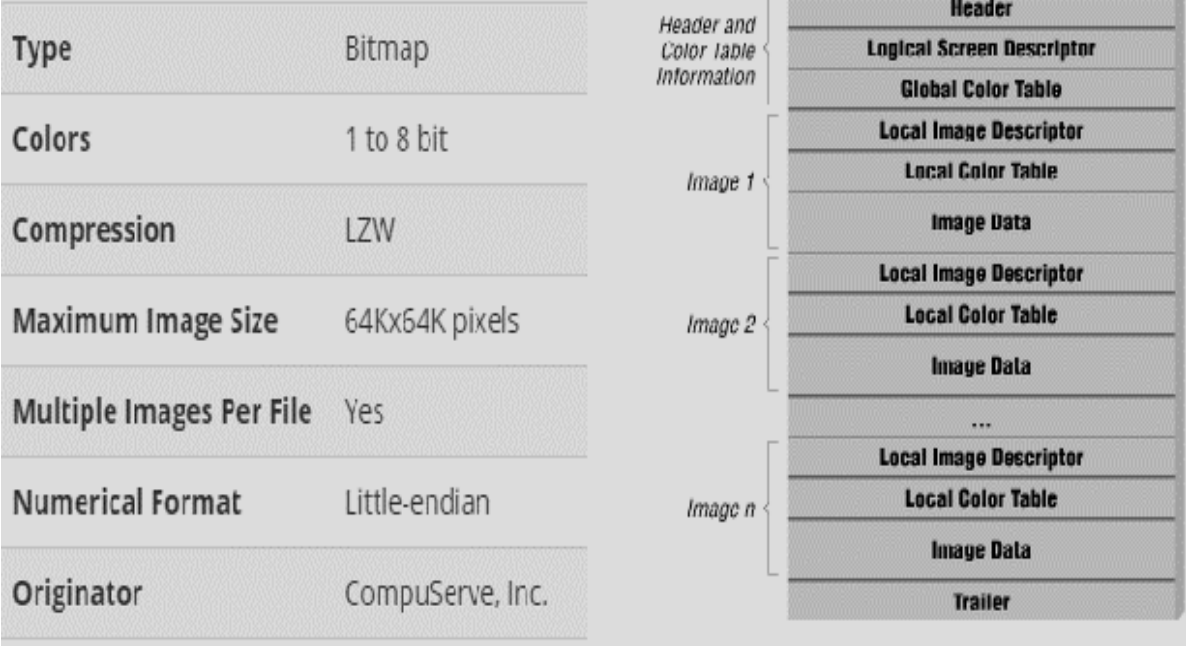


Figure 3.1: GIF format details and header

Portable Network Graphics (PNG) was designed to be the successor to GIF format, when CompuServe announced that programs implementing GIF would require royalties because of patent on LZW compression method used in GIF. The PNG datastream consists of a PNG signature (first 8 bytes) followed by a sequence of chunks. There are 18 chunk types defined in the International Standard, but the critical chunks which must be in every PNG file are: one IHDR (image header), one or more IDAT (image data) and one IEND (image trailer). Each chunk consists of three or four fields: Length, Chunk Type, Chunk Data and CRC. There are also variations of png file format such as MNG (Multiple image Network Graphics) with support of animation as animated GIF and APNG (Animated Portable Network Graphics) originally published by Mozilla developers but widely used for thumbnails on Sony’s Playstation.

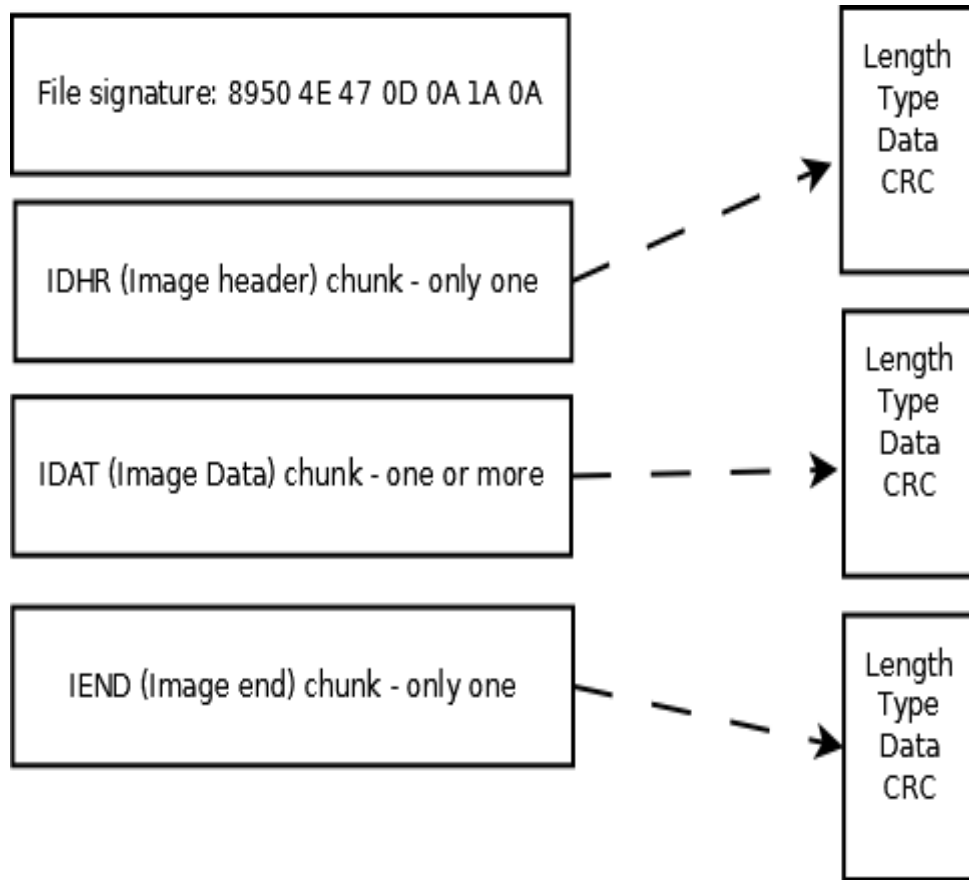


Figure 3.2: PNG file structure

Joint Photographic Experts Group (JPEG) is the most common image format used by digital cameras. It uses a loss compression method and typically a 10:1 compression with no particular loss quality in the image is achieved. A jpeg file has a signature of:

Table 3.1: Signature of a JPEG image

FF	D8	FF	E1	xx	xx	45	78	69	66	00
----	----	----	----	----	----	----	----	----	----	----

,where the fourth byte is indicative of the jpeg content. The options for the fourth byte are:

Table 3.2: Options for the fourth byte in jpeg header

DB	Samsung D807 JPEG file
E0	Standard JPEG/JFIF file
E1	Standard JPEG/EXIF file
E2	Canon EOS-1D JPEG file
E3	Samsung D500 JPEG file
E8	Still Picture Interchange File Format (SPIFF)

The file details are:

Type	Bitmap
Colors	Up to 24-bit
Compression	JPEG
Maximum Image Size	64Kx64K pixels
Numerical Format	Big-endian
Multiple Images Per File	No
Originator	C-Cube Microsystems

Figure 3.3 JPEG format details

A JPEG image consists of a sequence of segments, each beginning with a marker, each of which begins with a 0xFF byte followed by a byte indicating what kind of marker it is. The most common used markers are:

Table 3.3: The most common JPEG markers, <https://en.wikipedia.org/wiki/JPEG>

Short name	Bytes	Payload	Name and Comments
SOI	0xFF, 0xD8	None	Start Of Image
SOF0	0xFF, 0xC0	Variable size	Start Of Frame (Baseline DCT) Indicates that this is a baseline DCT-based JPEG, and specifies the width, height, number of components, and component subsampling
SOF2	0xFF, 0xC2	Variable size	Start Of Frame (Progressive DCT) Indicates that this is a progressive DCT-based JPEG, and specifies the width, height, number of components, and component subsampling
DHT	0xFF, 0xC4	Variable size	Define Huffman Table(s)
DQT	0xFF, 0xDB	Variable size	Define Quantization Table(s)
DRI	0xFF, 0xDD	2 bytes	Define Restart Interval Specifies the interval between RSTn markers, in macroblocks. This marker is followed by two bytes indicating the fixed size so it can be treated like any other variable size segment.
SOS	0xFF, 0xDA	Variable size	Start Of Scan Begins a top-to-bottom scan of the image. In baseline DCT JPEG images, there is generally a single scan. Progressive DCT JPEG images usually contain multiple scans. This marker specifies which slice of data it will contain, and is immediately followed by entropy-coded data.
RSTn	0xFF, 0xDn n(n=0..7)	None	Restart Inserted every r macroblocks, where r is the restart interval set by a DRI marker. Not used if there was no DRI marker. The low 3 bits of the marker code cycle in value from 0 to 7.
APPn	0xFF, 0xEn	Variable size	Application-specific For example, an Exif JPEG file uses an APP1 marker to store metadata, laid out in a structure based closely on TIFF.
COM	0xFF, 0xFE	Variable size	Comment
EOI	0xFF, 0xD9	None	End Of Image

Finally, Portable Document Format (PDF) is a file format used to present documents in a manner independent of application software, hardware and operating systems. Its file signature is 25 50 44 46 (hexadecimal) and the file structure is:

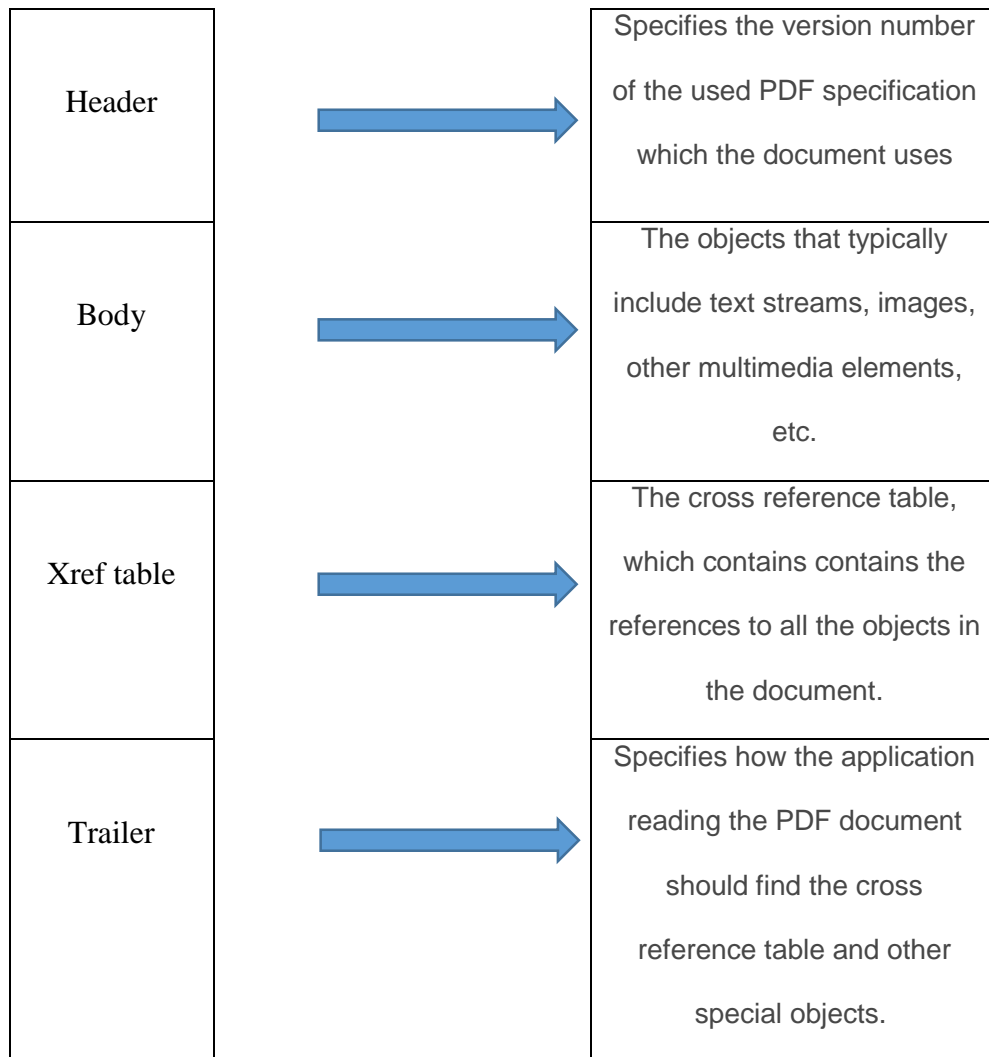


Figure 3.4: The file structure of a pdf file

We must bear in mind that all PDF readers must start reading a pdf file from its end.

Concerning the data mining software, Waikato Environment for Knowledge Analysis (Weka) [34] was used as it offers excellent tools for data preprocessing, classification etc. It is an open source

software developed by the University of Waikato in New Zealand written in Java and the version used in this master’s thesis was 3.6.

3.3 The dataset

Caltech 101 [35] was used as dataset. It is a dataset made by Caltech University and it is available online for free download. It contains images from 101 categories and the total number of images included in this dataset is 9.144. These images come in many subfolders and each subfolder contains images with the same name as other ones. All these images are in jpeg format, so we had to convert some of them to other formats such as gif and png. After this conversion for convenience in identification, we renamed the images from image 0001 to image 9144. Then we divided the dataset to training and test set.

Table 3.4: Caltech 101 Dataset

Caltech 101 Dataset						
Type	Total	Image Number	Training Set		Testing Set	
			Number of images	Image Number	Number of images	Image Number
jpeg	1840	0001-1840	1288	0001–1288	552	1289-1840
png	1840	1841-3680	1288	1841–3128	552	3129-3680
gif	1839	3681-5519	1287	3681-4967	552	4968-5519
Total	5519		3863		1656	

In addition to the Caltech dataset, we added 1840 pdf files which are undergraduate thesis found online to the library of Technological Institute of Heraklion [36] under the search term: ‘a’ in many departments. All files are open access to the public and anyone can download them. Therefore, the final dataset we used is as follows:

Table 3.5: Our Dataset

Dataset			
	Total files	Training	Testing
jpeg	1840	1288	552
png	1840	1288	552
gif	1839	1287	552
pdf	1840	1288	552
Total	7359	5151	2208

3.4 Feature extraction

We will use Byte Frequency Distribution (BFD) as feature extraction method. In order to create the byte frequency distribution, we must count the number of occurrences of each byte value for a single input file. We generate and use an array with elements from 0 to 255, and initialize all values to zero. Each byte in the input file is then looped through. For each byte, the value is extracted and the appropriate element of the array is incremented by one. For example, if the next byte in the file contained the ASCII value 21, then array element 21 would be incremented by one. Once the number of occurrences of each byte value is obtained, each element in the array is divided

by the number of occurrences of the most frequent byte value. This normalizes the array to frequencies in the range of 0 to 1.

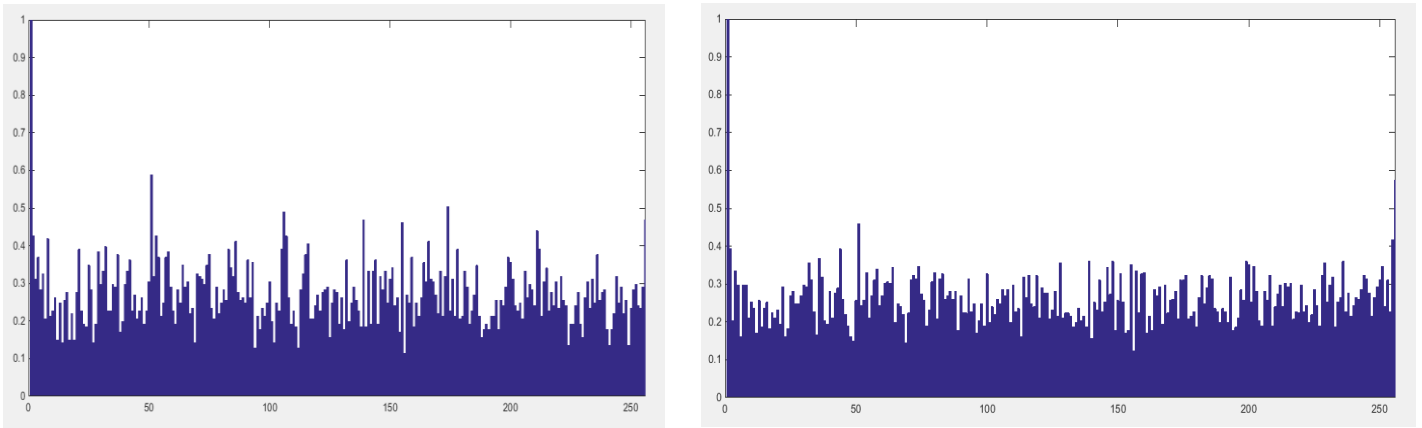


Figure 3.5: Byte Frequency Distributions for two jpg images

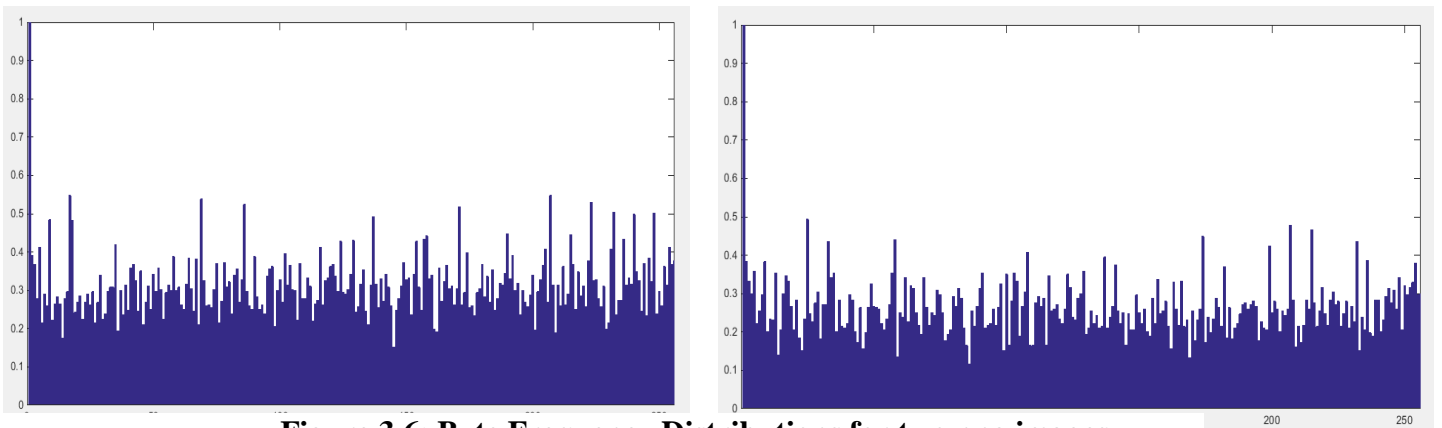


Figure 3.6: Byte Frequency Distributions for two png images

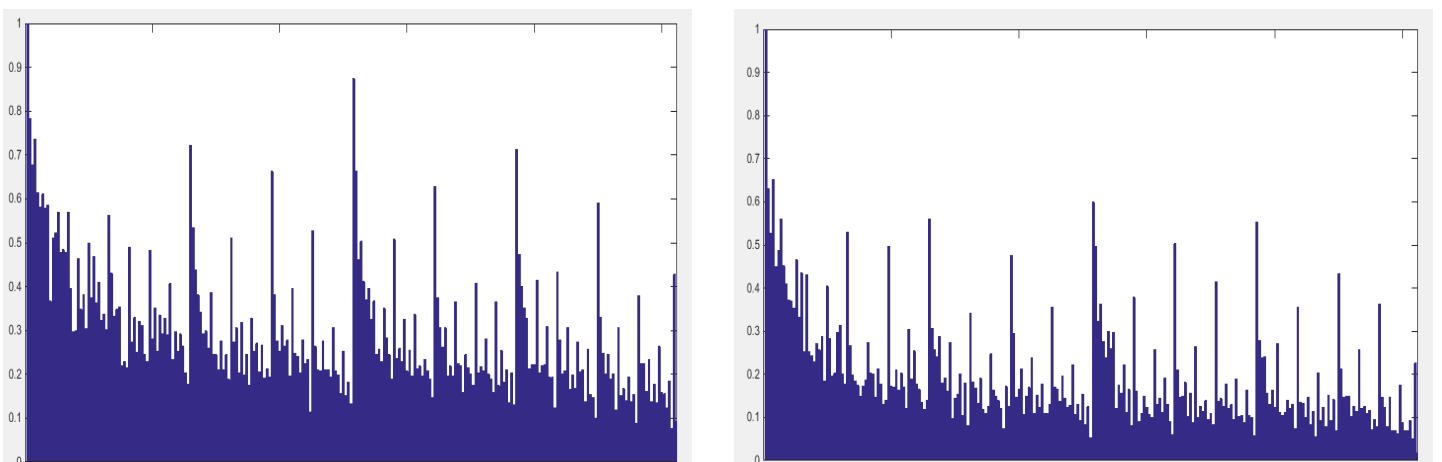


Figure 3.7: Byte Frequency Distributions for two gif images

It is obvious if we look carefully the above images that the BFD between the same file types are about the same. The main problem is that the number of features extracted from this method is 256 which means we have 2^{256} subset of features, therefore we must use feature selection in order to train our classifier correctly and decrease training time. A script was made to Matlab which extracts BFD for both training and test set. This script creates a comma separated value (csv) file which contains the 256 values of the file and also adds as 257th feature the instance's actual class (according to instance's extension). It must be said that the script can extract BFD of any file type and not only the four ones which this thesis will deal with.

3.5 Feature selection

Feature selection is the procedure of finding and selecting the minimum number of the most informative relevant features, in order to capture the patterns on our data whilst having the best results. It is a step prior to applying machine learning algorithms and while the size of data used becomes larger, it turns out to be an important and essential step too. Feature selection works by removing features that are not relevant or are redundant. The noteworthy benefits of performing feature selection on our data are:

- **Reduces Overfitting:** Less redundant data means less opportunity to make decisions based on noise.
- **Improves Accuracy:** Less misleading data means that model's accuracy increases.
- **Reduces Training Time:** Less data means that machine learning algorithms run faster.
- **Simplifies the Models:** More simple models are easier to be deployed or analyzed by the researchers.

It is also important to state that feature selection is different from dimensionality reduction. Although both methods try to reduce the number of attributes in the dataset, the dimensionality reduction method works in a different way as the resulting features are transformations of the original feature set, whereas feature selection methods include and exclude attributes already present in the data without changing them. One widely used dimensionality reduction method is Principal Component Analysis (PCA). Two are the main approaches for feature selection:

1. Filter Feature Selection Methods
2. Wrapper Feature Selection Methods

Filter feature selection methods make use of statistical measures in order to evaluate how relevant a feature is. This is done by obtaining the merit for each feature of the subset, the features are then ranked by their score and either selected to be kept or removed from the dataset. It is a pre-processing step and the subset of features selected is independent of the machine learning algorithm. This approach is faster than the wrapper method but the criterion used to evaluate the merit of a feature must be carefully chosen, otherwise this could lead to a machine learning model with poor results. The method is independent from the classifier, we select features only once and then we are able to use and evaluate different classifiers. The method is often univariate and considers the feature independently which means that the possibility of feature dependencies cannot be taken under consideration. A lot of techniques were proposed such as Correlation based Feature Selection (CFS), Gain Ration (GR), Chi squared, Information gain etc.

In wrapper methods the subsets of features are evaluated by the machine learning algorithm itself. Every subset is given a score by the algorithm and evaluated comparing to other subsets. The main advantage is that there is an interaction between feature subset search and also this

method takes into consideration possible feature dependencies. On the contrary it is obvious that this method is computationally inefficient due to large computation time, especially when the number of the extracted features is high. Furthermore the risk of overfitting is higher than the filter selection methods.

3.5.1 Correlation based Feature Selection (CFS)

CFS [37] is a filter feature selection method which gives high scores to subsets that include features that are highly correlated to the class attribute but have low correlation to each other. As Hall said:

“Good feature subsets contain features highly correlated with (predictive of) the class, yet uncorrelated with (not predictive of) each other.”

The implementation of CFS in Weka, allows the user to decide which heuristic search strategy will be applied. It is essential to report that CFS works well both for numerical and nominal types of data.

Let S be a feature subset consisting of k features. The merit of each subset is calculated as:

$$Merit_{S_k} = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k - 1)\overline{r_{ff}}}}$$

where:

$\overline{r_{cf}}$ is the average value of all feature-classification correlations and

$\overline{r_{ff}}$ is the average value of all feature-feature correlations

Finally, the criterion for the CFS algorithm is:

$$\text{CFS} = \max_{S_k} \left[\frac{r_{cf1} + r_{cf2} + \dots + r_{cfk}}{\sqrt{k + 2(r_{f_1f_2} + \dots + r_{f_if_j} + \dots + r_{f_kf_1})}} \right].$$

In our case, CFS will be used as the evaluation method for subsets coming out from a search method. Forward selection, backward elimination, and best first are a few search strategies among others in Weka. In forward selection initially there are no features. Afterwards, features are added to the subset until no higher evaluation of the subset is observed. Conversely in backward elimination there is a full feature set and as long as the evaluation of the subset does not worsen, one feature at a time is removed.

3.5.2 Best first as a search method

In best first we can choose to start either with no features or all the features. In the first case the search is like forward selection by adding single features, while in the second case the search is like backward selection by deleting single features. In order to avoid exploring the whole feature subset search space, a stopping criterion is obligatory. The search will stop if five sequential fully expanded subsets have less merit (score) compared to the current best subset. The flowchart of the method is shown in the next figure.

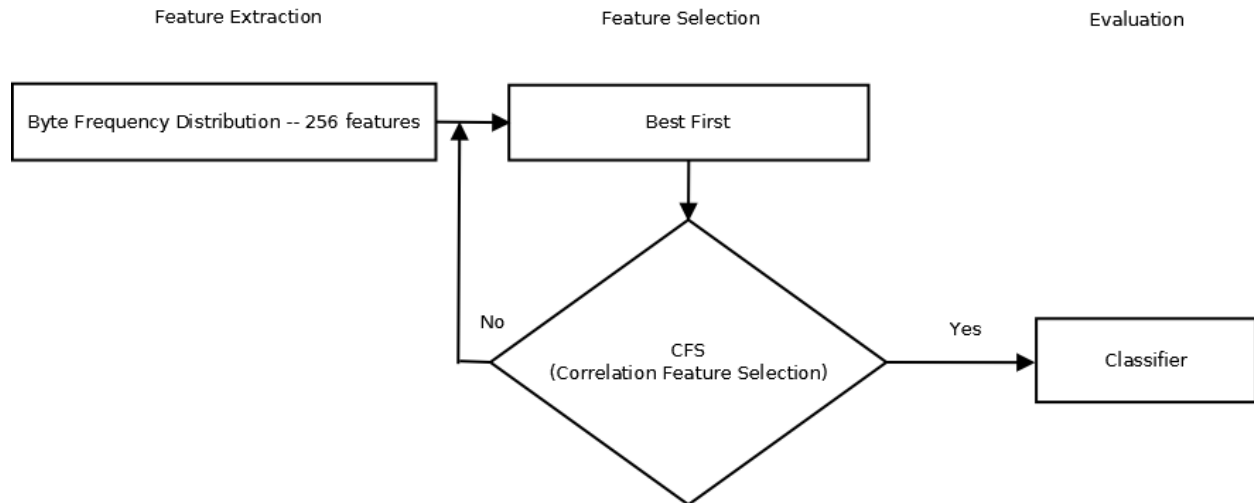


Figure 3.8: Feature selection flowchart – Best First

3.5.3 Genetic Algorithm as a search method

The idea of using a genetic algorithm for feature extraction is not new [38], [39], [40]. Genetic Algorithms (GAs) are evolutionary algorithms inspired by Darwin’s evolution and natural selection. It is an intelligent way to perform a “random” search in order to solve optimization problems. GAs comprise a subset of these evolution-based optimization techniques focusing on the application of selection, mutation, and recombination to a population of competing problem solutions. GAs are parallel iterative optimizers, and have been successfully applied to a broad spectrum of optimization problems, including many pattern recognition and classification tasks. In feature selection problems, each individual would represent a feature subset. Since the total number of features extracted in our case is 256, each chromosome is represented by a feature vector of dimension 256. If a bit’s value is zero (0) it means that the respective feature is not selected, and if the bit’s value is one (1) means that the feature is selected.

The score of each candidate solution can be evaluated using a fitness function, with respect to some criteria of interest. Weka uses Goldberg’s Genetic Algorithm [41]. In our case CFS will be the fitness function, Roulette wheel selection is used to probabilistically select individuals and Single-point crossover operator is used.

Table 3.6: Parameters of the Genetic Algorithm

Parameter	Value
Population size	256
Number of generations	100
Crossover	0.8
Mutation probability	0.033

The flowchart then of the selected method is shown on the next figure:

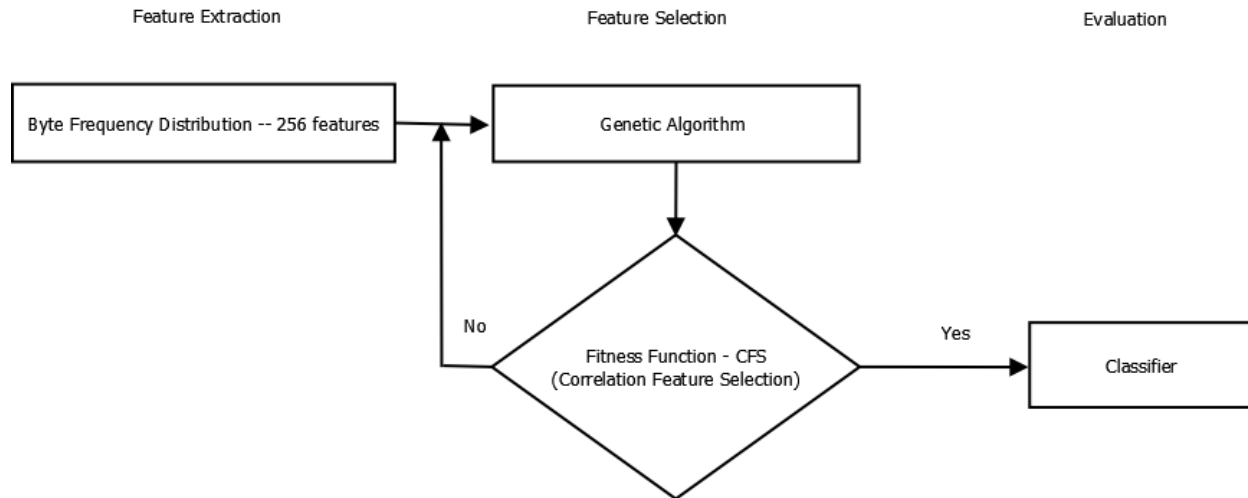


Figure 3.9: Feature selection flowchart – Genetic Algorithm

3.6 The Classifier – Multi Layer Perceptron (MLP)

As a classifier, we will use a feed forward backpropagation multilayer perceptron (MLP). A multilayer perceptron is used among others to classification or regression problems and typically the topology of a MLP includes the input layer, the hidden layer (or layers) and the output layer. A MLP with one hidden layer was used by Harris [42] in order to identify file types too.

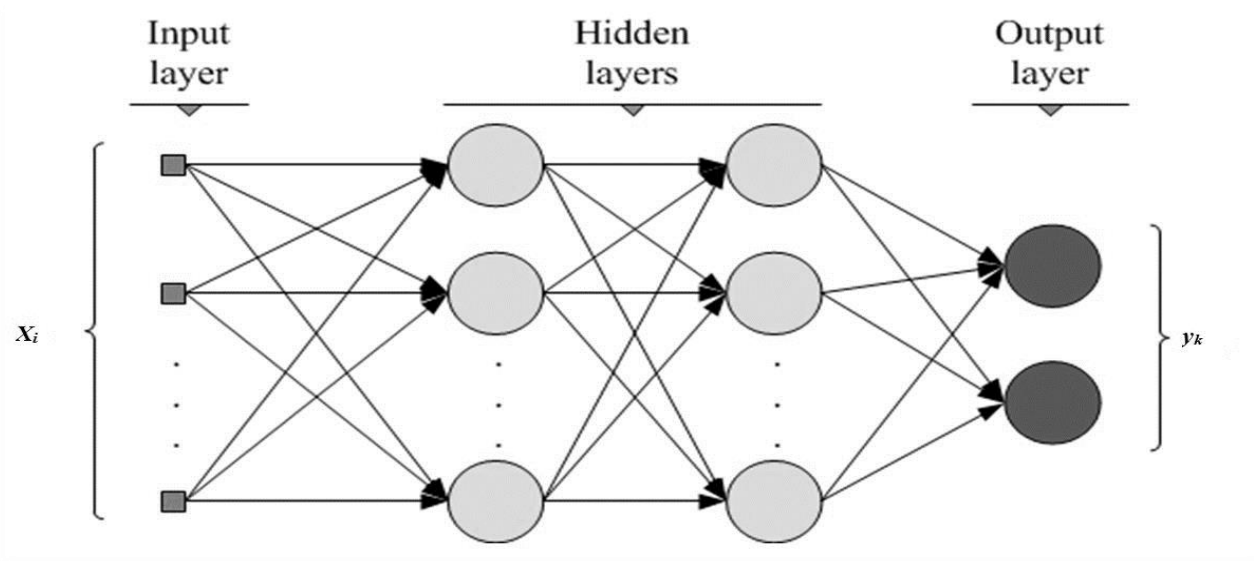


Figure 3.10: A multilayer perceptron with two hidden layers

Each layer is fully connected with each other and each node in one layer connects with a certain weight w_{ij} to every node in the following layer. When an input pattern is presented to the input layer, the weighted sum of the input to the j^{th} node in the hidden layer is given by:

$$\text{Net}_j = \sum w_{ij} x_j + \theta_j$$

,where:

x_j is the j^{th} input

w_{ij} is the weight (random value at first) and

θ_j is the bias of the neuron.

The bias is a "pseudo input" to each neuron in the hidden layer and the output layer, and it is used to surpass cases where the values of an input instance are zero. The neuron would 'fire' if the output value of the activation function (sigmoid in our case) overcomes a threshold and this value becomes an input to the neurons of the next layer connected to it. This is done until the output of the network is calculated. The calculated output of the network is then compared to the anticipated output, and an error signal is computed for each of the output nodes. This error is then backpropagated to the neural network and it is used to adjust the weights in order to decrease the error in every iteration until the neural network ideally derives the preferred output. In general, the backpropagation algorithm looks to converge the minimum value of the error function, by using a technique called the delta rule. This process is known as "training" and it is iteratively continued until the training time (number of epochs) is reached or another stopping criterion (e.g. mean square error) is met. In our case we will use a MLP in Weka with the following parameters:

Table 3.7: Parameters of the multilayer perceptron

Hidden layers	3
Learning rate	0.3
Momentum	0.2
Training time (epochs)	500

The number of nodes used in hidden layers are:

$$\frac{features+classes}{2} = \frac{44+4}{2} = 24$$

We have used a momentum in order to avoid local minimum and to accelerate the learning process.

3.7 Cross Validation of the training data

A dataset usually is divided into training and test set. A typical split for the dataset is 70% for the training and 30% for the test set. This is called the holdout method. Although this is a fast validation method, the main disadvantage is that if we have a small or non-balanced dataset, instances in training or test set may not be representative. This means that there might be none or few instances related to a class, which will result to a less accurate classification model.

For this reason, we will perform repeated stratified cross validation. In particular, we will use stratified 10 fold cross validation, which is found to be the best choice [43] to get an accurate estimate. We divide the training set into ten parts (folds). For each fold i ($i=1-10$), the classifier is trained by the instances that do not belong to fold i . Then the test fold i is applied and the error rate or the classifier is computed. This is done for 10 times and the total classifier error is:

$$\text{error} = \frac{\sum_{i=1}^k n_i}{m}$$

where: n_i is the the number of examples in Fold i that were wrongly classified and

m is the total number of instances.

Finally, stratified means that every fold has the right fraction of each class value. In Weka when we implement a k -fold cross validation, the algorithm which trains the classifier (backpropagation in this case) runs once more (11th time) using 100% of the training data and this finally results to a classification model. Then we can present unseen instances (i.e test sets) to the model and predict their class.

Therefore the final flowchart of our proposed method is:

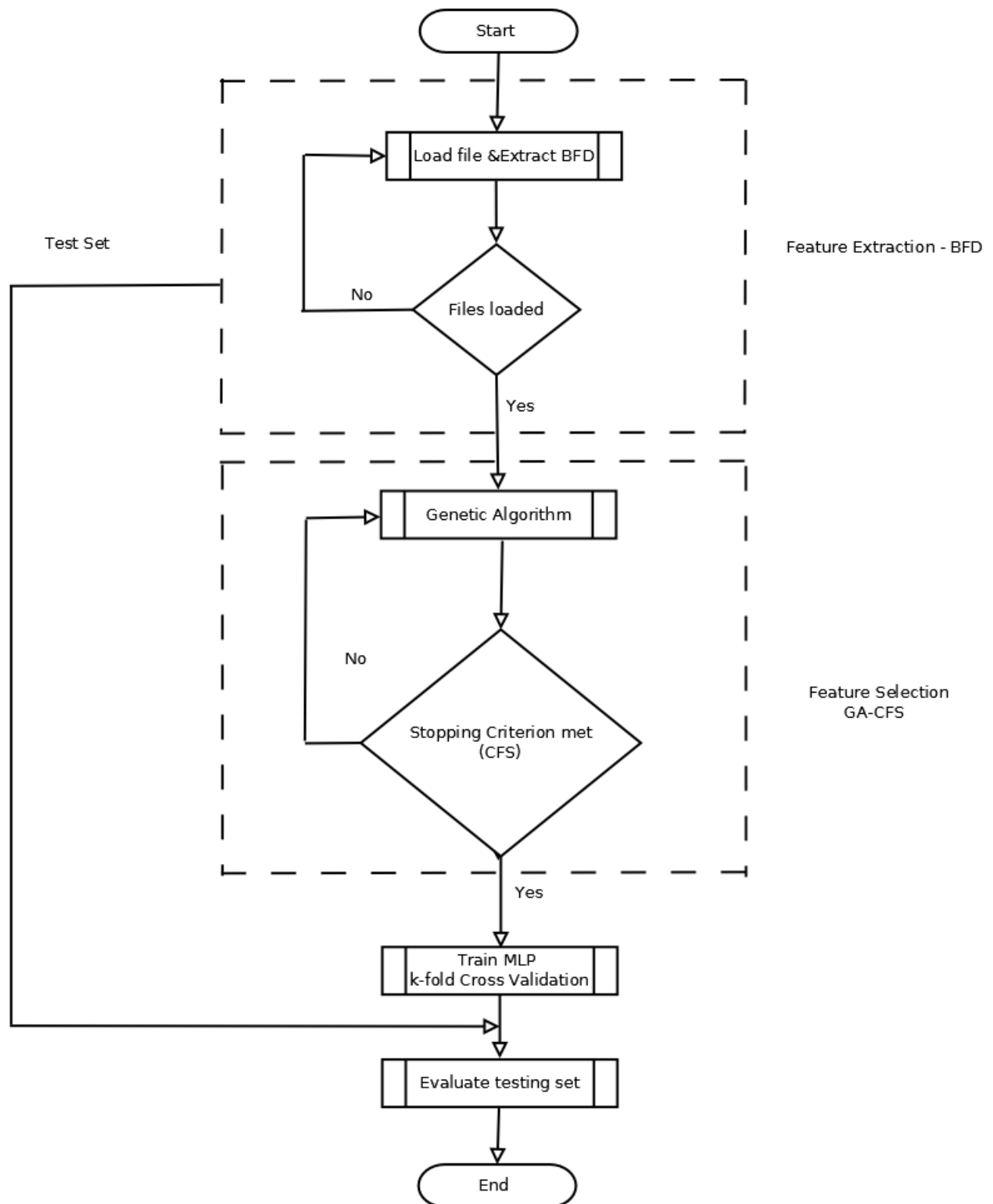


Figure 3.11: Flowchart of the proposed method

Chapter 4 - Results

4.1 Results using Best First as search method

Weka was used for feature selection and classification. First we used Best First as feature selection method. This resulted to 13 features out of 256 i.e 94.92% reduction. The most informative features were:

Table 4.1: Remaining features after selection with Best First search method and CFS

1	2	3	9	33	48	49	51	65	67	128	133	224
---	---	---	---	----	----	----	----	----	----	-----	-----	-----

Then the remaining features were used to train the classifier, a multilayer perceptron run the backpropagation algorithm for 500 epochs and the resulted confusion matrix was:

Table 4.2: Confusion matrix – Best First - CFS – Training time 500 epochs

<i>=== Confusion Matrix ===</i>				
a	b	c	d	← classified as
552	0	0	0	a = .jpg
15	528	6	3	b = .pdf
5	3	538	6	c = .png
1	0	5	546	d = .gif

We then tried to examine the classifier' behavior by increasing the number of epochs to 1000. The confusion matrix was:

Table 4.3: Confusion matrix – Best First - CFS – Training time 1000 epochs

<i>=== Confusion Matrix ===</i>				
a	b	c	d	← classified as
552	0	0	0	a = .jpg
21	522	5	4	b = .pdf
5	3	539	5	c = .png
1	0	6	545	d = .gif

It is obvious, if we compare the two above confusion matrices that there is no improvement to classification results. On the contrary, there was a small decrease to classification rates especially to pdf files.

4.2 Results using Genetic Algorithm as search method

Afterwards the search method for the candidate subset, changed to a Genetic Algorithm. This resulted to the selection of 44 features i.e. 82.81% reduction, which were:

Table 4.4: Remaining features after selection with GA as search method and CFS

1,2,3,4,5,6,8,10,11,13,14,15,17,20,21,24,26,33,37,38,41,46,48,49,54,65,69,79,81,105,109,113,130,132,133,144,168,176,194,210,222,244,250,254

We then trained again the same classifier for 500 epochs and the results are:

Table 4.5: Confusion matrix – Genetic Algorithm - CFS – Training time 1000 epochs

<i>=== Confusion Matrix ===</i>				
a	b	c	d	← classified as
552	0	0	0	a = .jpg
2	547	2	1	b = .pdf
4	3	542	3	c = .png
0	0	8	544	d = .gif

Once more, if we increased the number of epochs to 1000 the results were not better. As a matter of fact, we noticed the same results we had with 500 epochs as training time.

4.3 Comparison of the two search methods

The detailed accuracy for the first search method (Best First) is:

Table 4.6: Detailed Accuracy for Best First

Detailed Accuracy By Class						
TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
1	0.013	0.963	1	0.981	1	.jpg
0.957	0.002	0.994	0.957	0.975	0.993	.pdf
0.975	0.007	0.980	0.975	0.977	0.993	.png
0.989	0.005	0.984	0.989	0.986	0.997	.gif
Weighted Avg.	0.98	0.007	0.980	0.980	0.98	0.996

The detailed accuracy for the second search method (GA) is:

Table 4.7: Detailed Accuracy for Genetic Algorithm

Detailed Accuracy By Class						
TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
1	0.004	0.989	1	0.995	1	.jpg
0.991	0.002	0.995	0.991	0.993	0.998	.pdf
0.982	0.006	0.982	0.982	0.982	0.998	.png
0.986	0.002	0.993	0.986	0.989	1	.gif
Weighted Avg.	0.990	0.003	0.990	0.990	0.990	0.999

Where:

- **TP Rate:** True Positives Rate (instances correctly classified as a given class)
- **FP Rate:** False Positives Rate (instances falsely classified as a given class)
- **Precision:** fraction of instances that are truly of a class divided by the total instances classified as that class
- **Recall:** fraction of instances classified as a given class divided by the actual total in that class (equivalent to TP rate)
- **F-Measure:** A combined measure for precision and recall calculated

$$\text{as: } 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Comparing the two methods (concerning TP Rate, Precision, Recall):

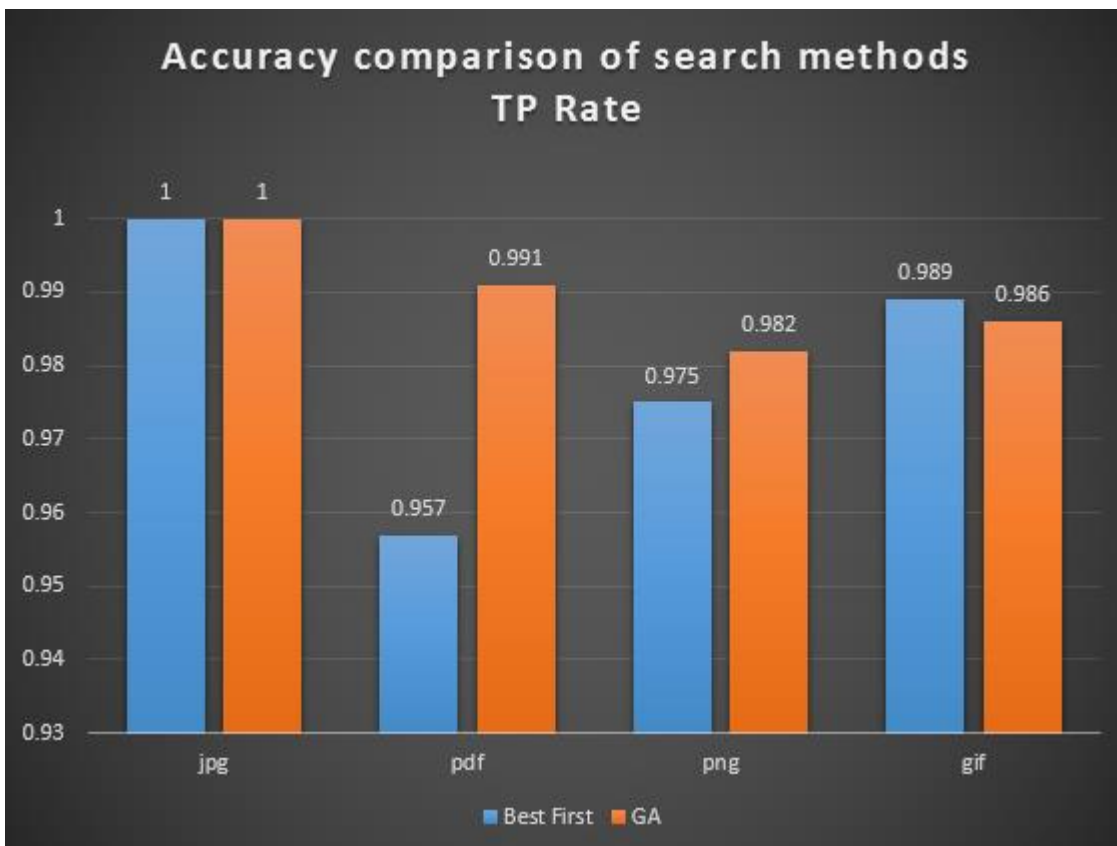


Figure 4.1: Comparison of search methods TP Rate

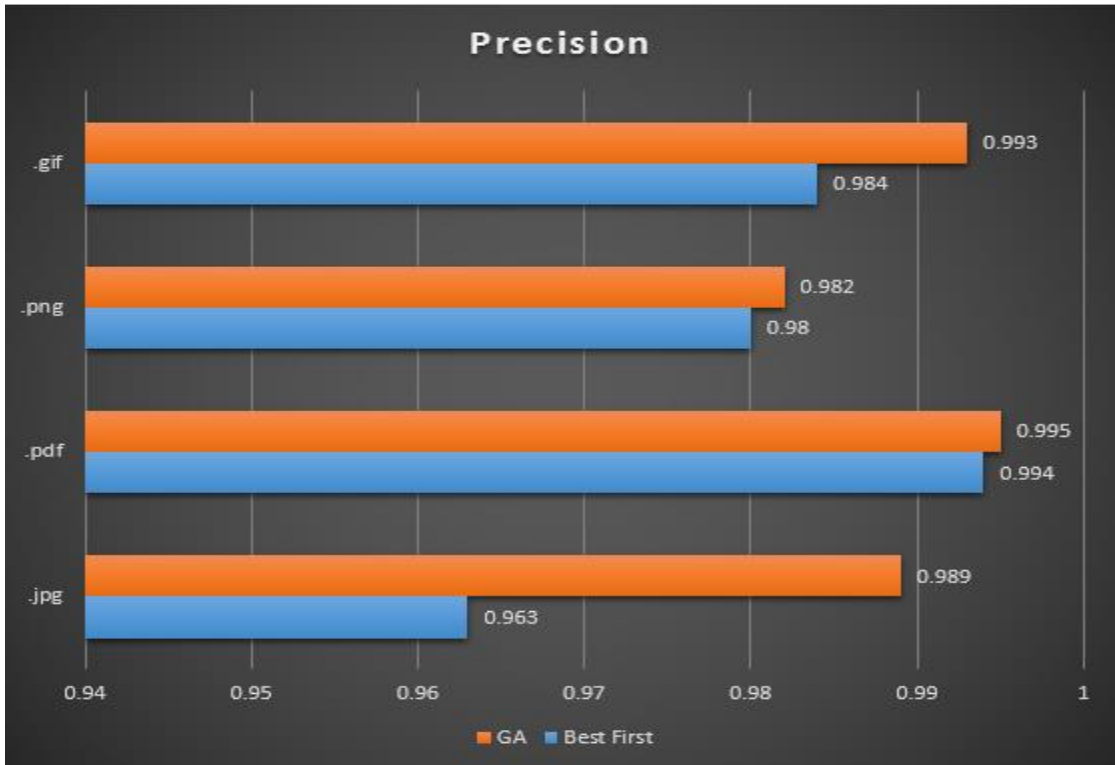


Figure 4.2: Comparison of search methods – Precision

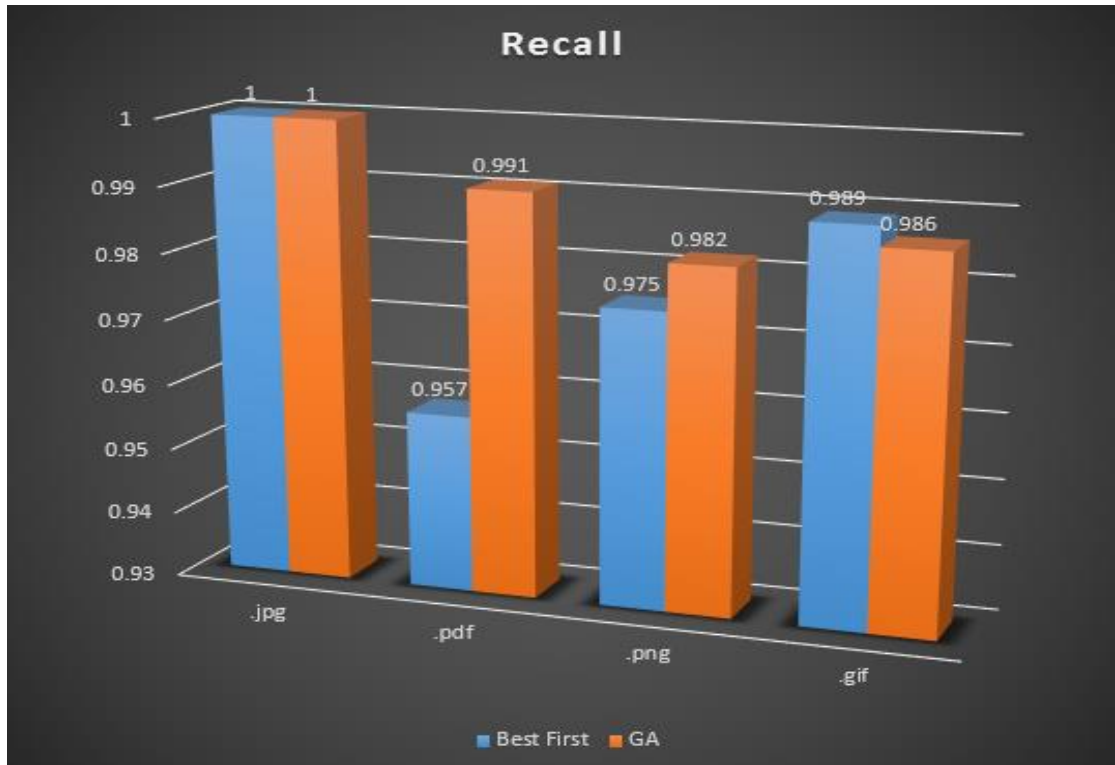


Figure 4.3: Comparison of search methods – Recall

4.4 Restatement of the problem

Our method worked well and had very good results. However, will this proposed method work evenly well - by digital forensics perspective - if someone alter image files (both extension and signature) and transform them to pdf?

In order to examine this, we made a new dataset. This time the difference is that 30% - approx. - of the 552 pdf files in test set i.e. 168 files, was in fact images intentionally changed (extension & signature) to pdf files.

Table 4.8: The new dataset

Dataset			
	Total files	Training	Testing
jpeg	1840	1288	552
png	1840	1288	552
gif	1839	1287	552
pdf	1840	1288	552
Total	7359	5151	2208

In order to identify easily these 168 altered ‘pdf’ files, we renamed them from forged_XX_(1).pdf to forged_XX_(168).pdf, where XX is the actual image type e.g. forged_jpg_(1).pdf means that the actual type of the file is a jpg image and forged_png_(1).pdf means that the actual type of the file is a png image. Therefore, the classification model was deployed for three times and every time the 168 forged ‘pdf’ files in the dataset were changed.

4.5 Results on the new testing datasets

In all three cases a Genetic Algorithm was selected as a search method for the candidate features (with CFS as a fitness function) and a multilayer perceptron was used as a classifier.

1. *Altering jpg images to pdf files*

The resulted confusion matrix is:

Table 4.9: Confusion matrix – Identifying forged jpg images

=== Confusion Matrix ===				
a	b	c	d	← classified as
552	0	0	0	a = .jpg
170	379	2	1	b = .pdf
4	3	542	3	c = .png
0	0	8	544	d = .gif

This time the accuracy for pdf files seemed to worsen. Recall that there are 168 pdf files which their actual type is jpg image. By comparing the output predictions in weka and the testing dataset we found that the misclassified files were in fact the altered jpg images.

Table 4.10: “Misclassified” pdf instances (jpg actual type)

Instance Number	Instance Name	Actual Type	Predicted Type
1778-1945	forged_jpg_(1).pdf - forged_jpg_(168).pdf	jpg	jpg

From the above table we concluded that every file we transformed (from jpg to pdf) was accurately predicted. Therefore, the actual confusion matrix in our case is:

Table 4.11: Actual confusion matrix – jpeg images

=== Confusion Matrix ===				
a	b	c	d	← classified as
552	0	0	0	a = .jpg
2	547 (379+168)	2	1	b = .pdf (with the forged images)
4	3	542	3	c = .png
0	0	8	544	d = .gif

The accuracy of predicting correctly the actual class of the altered files is 100%.

Table 4.12: Detailed Accuracy By Class – Our proposed method (in forged jpeg images)

Detailed Accuracy By Class						
TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
1	0.004	0.989	1	0.995	1	.jpg
0.991	0.002	0.995	0.991	0.993	0.998	.pdf
0.982	0.006	0.982	0.982	0.982	0.998	.png
0.986	0.002	0.993	0.986	0.989	1	.gif
Weighted Avg.	0.990	0.003	0.990	0.990	0.999	

2. Altering png images to pdf files

The resulted confusion matrix was:

Table 4.13: Confusion matrix – Identifying forged png images

=== Confusion Matrix ===				
a	b	c	d	← classified as
552	0	0	0	a = .jpg
3	385	162	2	b = .pdf
4	3	542	3	c = .png
0	0	8	544	d = .gif

By comparing the output predictions in Weka and the testing dataset we found that the misclassified files were:

Table 4.14: “Misclassified” pdf instances (png actual type)

Instance Number	Instance Name	Actual Type	Predicted Type
1778-1877	forged_png_(1).pdf - forged_png_(100).pdf	png	png
1879-1901	forged_png_(102).pdf - forged_png_(124).pdf	png	png
1903-1918	forged_png_(126).pdf - forged_png_(141).pdf	png	png
1920	forged_png_(143).pdf	png	png
1923-1929	forged_png_(146).pdf - forged_png_(152).pdf	png	png
1931-1945	forged_png_(154).pdf - forged_png_(168).pdf	png	png

From the above table we concluded that only 6 out of 168 png altered files were not predicted correctly. This gives a 96.43% accuracy for png altered images. Thus, the final confusion matrix is:

Table 4.15: Actual confusion matrix – png images

=== Confusion Matrix ===				
a	b	c	d	← classified as
552	0	0	0	a = .jpg
3	547 (385+162)	0	2	b = .pdf (with the forged images)
4	3	542	3	c = .png
0	0	8	544	d = .gif

3. Altering gif images to pdf files

The resulted confusion matrix was:

Table 4.16: Confusion Matrix – Identifying forged gif images

=== Confusion Matrix ===				
a	b	c	d	← classified as
552	0	0	0	a = .jpg
2	379	2	169	b = .pdf
4	3	542	3	c = .png
0	0	8	544	d = .gif

By comparing the output predictions in Weka and the testing dataset we found that the misclassified files were:

Table 4.17: “Misclassified pdf” instances (gif actual type)

Instance Number	Instance Name	Actual Type	Predicted Type
1778-1945	forged_gif_(1).pdf - forged_gif_(168).pdf	gif	gif

Again the accuracy of the model to the altered images is 100%. The actual confusion matrix then is:

Table 4.18: Actual confusion matrix – gif images

=== Confusion Matrix ===				
a	b	c	d	← classified as
552	0	0	0	a = .jpg
2	547 (379+168)	2	1	b = .pdf
4	3	542	3	c = .png
0	0	8	544	d = .gif

Combining the above accuracy results for the altered images (jpg, gif & png) in the three test datasets, we have:

Table 4.19: Final Confusion Matrix of the proposed method

=== Confusion Matrix of the forged files===				
a	b	c	d	← classified as
168	0	0	0	a = .jpg
0	0	0	0	b = .pdf
0	6	162	0	c = .png
0	0	0	168	d = .gif

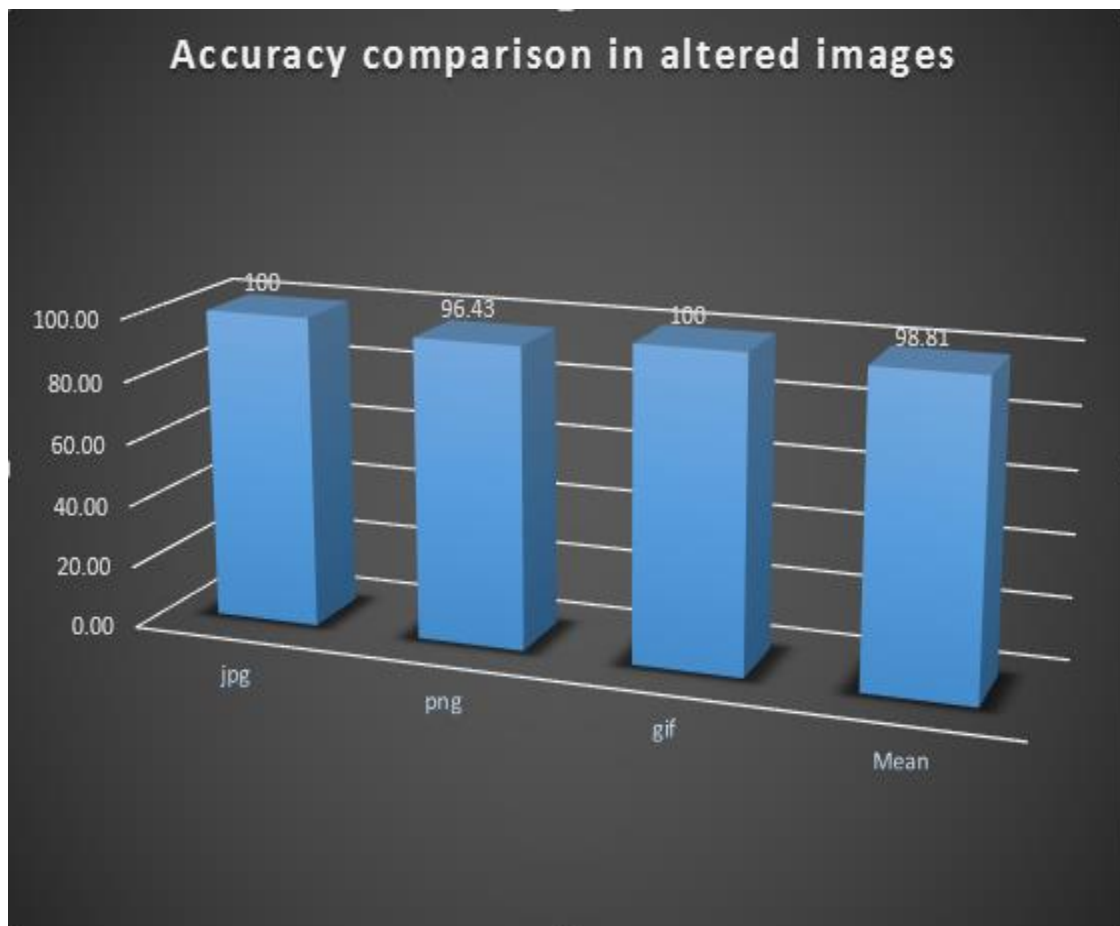


Figure 4.4: Accuracy comparison of the proposed method in altered images

4.6 Comparison of the proposed method to the literature

In this point it is wise to summarize in a table, the most promising methods proposed by other researchers along with ours. This is already done in previous chapters but for reasons of practice the following table is given.

Table 4.20: The proposed method compared to the literature

<i>Researchers</i>	<i>Year</i>	<i>Proposed method</i>	<i>File Types</i>	<i>Number of files</i>	<i>Accuracy</i>
McDaniel and Heydari	2003	BFA, BFC, FHT analysis	30	120	27.5, 45.83, 95.83
Li et al.	2005	Manhattan distance, Mahalanobis distance, Multi-centroid	8 (5)	800	82 (One-Centroid) 89.5 (Multi-Centroid), 93.8 (Exemplar files)
Dunham et al.	2005	Neural Networks	10	760	91.3
Amirani et al.	2010	PCA + Neural networks feature extraction. MLP Classifier	6	720	98.33
Cao et al.	2010	Gram Frequency Distribution, Vector space model	4	1000	90.34 (2-gram + 256 grams as type signature)
Ahmed et al.	2010	Cosine similarity, divide and conquer, MLP Classifier	10	2000	90.19
Ahmed et al.	2011	Feature Selection, Content Sampling, KNN Classifier	10	5000	90.5 (40% of features), 88.45 (20% of features)
Amirani et al.	2013	PCA + Neural Networks feature extraction SVM Classifier	6	1200	99.16 (Whole files), 85.5 (1500 bytes fragments), 82 (1000 bytes fragments)

Evensen et al.	2014	n-gram analysis with naïve Bayes classifier	6	60000	99.51 (Whole files), 99.08 (8192 bytes fragments 5 types), 98.34 (1024 bytes fragments, 5 types)
Our method	2015	CFS+Genetic Algorithm feature extraction, MLP classifier	4	7359	98.96% (Whole file) 98.81% (Digital Forensics perspective)

In addition to the above mentioned methods, others were suggested too but we included only those ones which dealt with whole files, in order to make the comparison easier.

Chapter 5 - Conclusions & Future Work

5.1 Conclusions

In this master thesis we tried to examine the problem of altering and identifying files by a digital forensics viewpoint. In the beginning a small introduction to Digital Forensics was made, in order to help the reader to fully understand the significance of file type identification. All possible ways of altering a file were enumerated, along with the most widespread software for correct forensic identification. We must take into consideration that there is no official standard for file types and this made the problem even harder. File Type Identification turned out to be a very demanding problem as a lot of parameters had to be examined in order to have optimal results. For example, one major step prior to classification was feature extraction and feature selection. Especially the right choice of an algorithm in order to remove irrelevant and redundant features, was a critical step as Byte Frequency Distribution (BFD) -which used for feature extraction- produced a large number of features. The idea of using a Genetic Algorithm along with CFS as its fitness function worked well and reduced the number of features. The selected features then – after 10-fold cross validation of the data- were used to train a multilayer perceptron and the classification results were very promising. Furthermore this method was tested as a forensic tool and gave excellent results as well. Along with the proposed method, a literature review was made [44] and presented and finally our proposed method of file type identification was compared to the literature. The proposed method identifies four (4) types of files (jpeg, png, gif & pdf), which happen to be the most common file types in anyone's computer or other electronic device.

5.2 Future Work

The results taken from this proposed method were very good and very promising. As mentioned we tried and managed to identify types of whole files. It should be very interesting to deploy our model in fragments of files and examine its behavior. During our research we had strong evidence that the proposed model would work well too, although modifications and changes have to be made to the model.

One other aspect of the problem is to try to identify more file types. Since our script which extracts BFD can easily find the features of any file type and not the specific four file types, this could be an extension to this research. Another possible future study is to examine if this model works also well in stego-images. It should be very interesting to find out if a stego-image should be recognized and furthermore -if we wanted to expand our research- to extract the hidden ‘information’ from the stego-image. Video triage and examination would be another domain of expanding the proposed method. We could also make new classification models using different classifiers and examine which one has better results.

The most promising area of future work is file fragments identification. If a classification model is created and manages to identify accurately fragments of files, this – after expanding it to identify as many file types as possible- might become an excellent tool to the hands of forensics examiners in order to fight digital crime.

References

- [1] D. Halder and K. Jaishankar, Cyber crime and the Victimization of Women: Laws, Rights, and Regulations., Hershey, PA, USA: IGI Global, 2011.
- [2] NIST, "<http://csrc.nist.gov/>," August 2006. [Online]. Available: <http://csrc.nist.gov/publications/nistpubs/800-86/SP800-86.pdf>. [Accessed 26 October 2015].
- [3] "Kali Linux," [Online]. Available: <https://www.kali.org/>. [Accessed 26 October 2015].
- [4] "Caine," [Online]. Available: <http://www.caine-live.net/>. [Accessed 26 October 2015].
- [5] "DEFT," DEFT Association, [Online]. Available: <http://www.deflinux.net/>. [Accessed 26 October 2015].
- [6] "Back Box Linux," [Online]. Available: <https://www.backbox.org/>. [Accessed 26 October 2015].
- [7] "NetSEcL The Linux Networking," [Online]. Available: <http://netsecl.com/>. [Accessed 26 October 2015].
- [8] "Parrot Security OS," [Online]. Available: <http://www.parrotsec.org/>. [Accessed 26 October 2015].
- [9] G. S. Inc., "Computer Forensic Software- Encase Forensic," Guidance Software Inc., [Online]. Available: <https://www.guidancesoftware.com/products/Pages/encase-forensic/overview.aspx>. [Accessed 26 October 2015].
- [10] "The Sleuth Kit (TSK) & Autopsy: Open Source Digital Forensic Tools," [Online]. Available: <http://www.sleuthkit.org/autopsy/>. [Accessed 26 October 2015].
- [11] A. Group, "E-Discovery & Computer Forensics | AccessData," AccessData Group, [Online]. Available: <http://accessdata.com/>. [Accessed 26 October 2015].
- [12] O. Forensics, "Oxygen Forensic Suite," Oxygen Forensics, [Online]. Available: <http://www.oxygen-forensic.com/>. [Accessed 26 October 2015].
- [13] G. Kessler, "File Signatures," [Online]. Available: http://www.garykessler.net/library/file_sigs.html.

- [14] M. Pontello, "Marco Pontello's Home - Software - TrID," [Online]. Available: <http://mark0.net/soft-trid-e.html>. [Accessed 27 October 2015].
- [15] ShockingSoft, "Analyze It!! v2.0 by Shocker, ShockingSoft," [Online]. Available: <http://www.shockingsoft.com/AnalyzeIt.html>. [Accessed 27 October 2015].
- [16] P. Harvey, "Exiftool by Phil Harvey," [Online]. Available: <https://www.sno.phy.queensu.ca/~phil/exiftool/>. [Accessed 27 October 2015].
- [17] "File Identifier- Identify unknown files instantly," [Online]. Available: <https://www.toolsley.com/file.html>. [Accessed 27 October 2015].
- [18] "Download DROID: file format identification tool - The National Archives," The National Archives, [Online]. Available: <http://www.nationalarchives.gov.uk/information-management/manage-information/preserving-digital-records/droid/>. [Accessed 27 October 2015].
- [19] S. Zevin, "Identifications : Create," [Online]. Available: <http://ec2-54-148-254-76.us-west-2.compute.amazonaws.com/falstaff/>. [Accessed 27 October 2015].
- [20] M. McDaniel, "Automatic File Type Detection Algorithm," Masters Thesis, James Madison University, 2001.
- [21] M. McDaniel and M. H. Heydari, "Content based file type detection algorithms," in *Proceedings of the 36th IEEE Annual Hawaii International Conference on System Science (HICSS'03)*, 2003.
- [22] W. Li, K. Wang, S. J. Stolfo and B. Herzog, "Fileprints: Identifying file types by n-gram Analysis," in *Proceedings of the 6th IEEE Systems, Man and Cybernetics Information Assurance Workshop*, West Point , New York, 2005.
- [23] J. G. Dunham, M. T. Sun and J. Tseng, "Classifying File Type of Stream Ciphers in Depth Using Neural Networks," in *The 3rd ACS/IEEE International Conference on Computer Systems and Applications*, 2005.
- [24] M. Karresand and N. Shahmehri, "File Type Identification of Data Fragments by Their Binary Structure," in *Proceedings of the IEEE Workshop on Information Assurance*, 2006.
- [25] Z. Like and G. B. White, "An Approach to Detect Executable Content for Anomaly Based Network Intrusion Detection," in *IPDPS 2007*, Long Beach, California, 2007.

- [26] R. F. Erbacher and S. J. Moody, "Sadi-statistical analysis for data type identification," in *Systematic Approaches to Digital Forensic Engineering, 2008. SADFE'08. Third International Workshop on*, 2008.
- [27] W. Calhoun and D. Coles, "Predicting the types of file fragments," *Journal Digital Investigation: The International Journal of Digital Forensics & Incident*, vol. 5, pp. 14-20, 2008.
- [28] M. C. Amirani, M. Toorani and A. Beheshti, "A new approach to content-based file type detection," in *Proceedings of the 13th IEEE Symposium on Computers and Communications (ISCC'08)*, 2008.
- [29] D. Cao, J. Luo, M. Yin and H. Yang, "Feature selection based file type identification algorithm," in *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference*, 2010.
- [30] I. Ahmed, K. Lhee, H. Shin and M. Hong, "Content-based file-type identification using cosine similarity and a divide-and-conquer approach," *IETE Technical Review*, vol. 27, no. 6, pp. 465-477, 2010.
- [31] I. Ahmed, K. Lhee, H.-J. Shin and M.-P. Hong, "Fast Content-Based File Type Identification," in *Advances in Digital Forensics VII*, Orlando, FL, USA, Springer Berlin Heidelberg, 2011, pp. 65-75.
- [32] M. C. Amirani, M. Toorani and S. Mihandoost, "Feature-based type identification of file fragments," *Security and Communication Networks*, vol. 6, no. 1, pp. 115-128, 2013.
- [33] J. Evensen, S. Lindahl and M. Goodwin, "File-type Detection Using Naïve Bayes and n-gram Analysis," in *Norwegian Information Security Conference, NISK 2014*, Fredrikstad, 2014.
- [34] U. o. Waikato, "Weka 3 - Data Mining with Open Source Machine Learning Software in Java," University of Waikato, 2015. [Online]. Available: <http://www.cs.waikato.ac.nz/ml/weka/>. [Accessed 29 October 2015].
- [35] L. Fei-Fei, R. Fergus and P. Perona, "Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories," in *IEEE. CVPR 2004, Workshop on Generative-Model Based Vision*, 2004.
- [36] T. I. o. Crete, "E-Thesis," [Online]. Available: <http://nefeli.lib.teicrete.gr/search/>. [Accessed 27 October 2015].

- [37] M. A. Hall, "Correlation-based Feature Selection for Machine Learning," April 1999. [Online]. Available: <http://www.cs.waikato.ac.nz/~mhall/thesis.pdf>. [Accessed 27 October 2015].
- [38] H. Vafaie and K. De Jong, "Genetic algorithms as a tool for feature selection in machine learning," in *Fourth International Conference on Tools with Artificial Intelligence, TAI '92*, Arlington, VA, 1992.
- [39] . Z. Li, Z. Jing, W. Fang, . L. Xia, A. Bin and Q. Junping, "A genetic algorithm based wrapper feature selection method for classification of hyper spectral data using support vector maching," *Geographical Research*, vol. 27, no. 3, pp. 493-501, 2008.
- [40] L. Jourdan, C. Dhaenens and E.-G. Talbi, "A Genetic Algorithm for Feature Selection in Data-Mining for Genetics," in *MIC'2001 - 4th Metaheuristics International Conference*, Porto, 2001.
- [41] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Boston: Addison-Wesley Longman Publishing Co., Inc., 1989.
- [42] R. M. Harris, "Using Artificial Neural Networks for Forensic File Type Identification," Purdue University, West Lafayette, Indiana, 2007.
- [43] R. Kohavi, "A study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," in *Fourteenth International Joint Conference on Artificial Intelligence*, Quebec Canada , 1995.
- [44] K. Karampidis, G. Papadourakis and I. Deligiannis, "File Type Identification - A Literature Review," in *9th International Conference on New Horizons in Industry, Business and Education, NHIBE 2015*, Skiathos, 2015.

This page was intentionally left blank

Appendix A - Weka Implementation

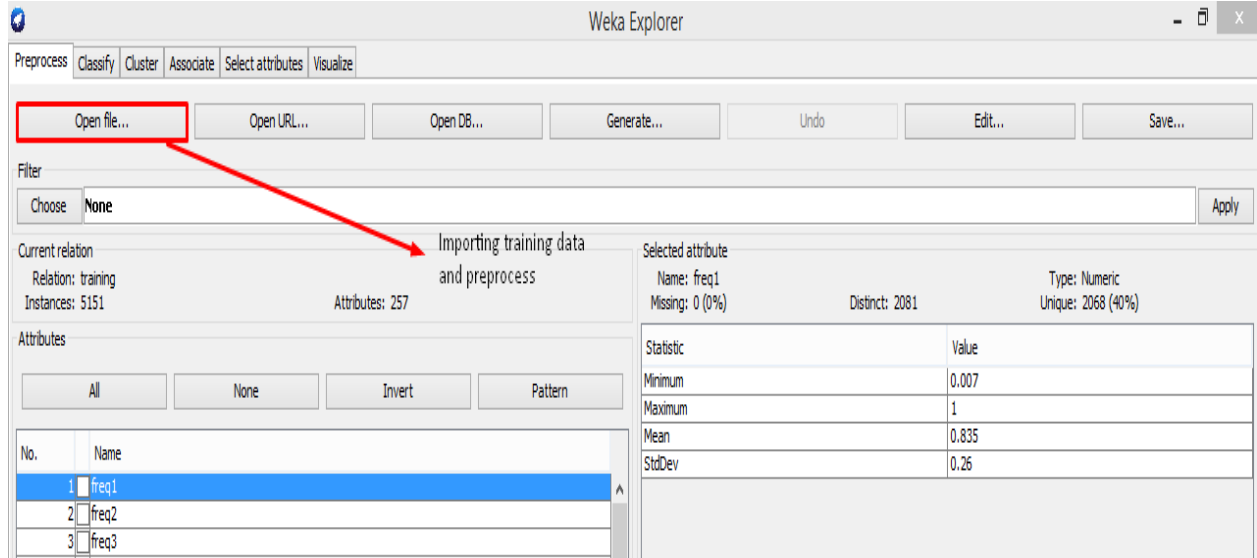


Figure A-1: Importing training set and preprocess in Weka

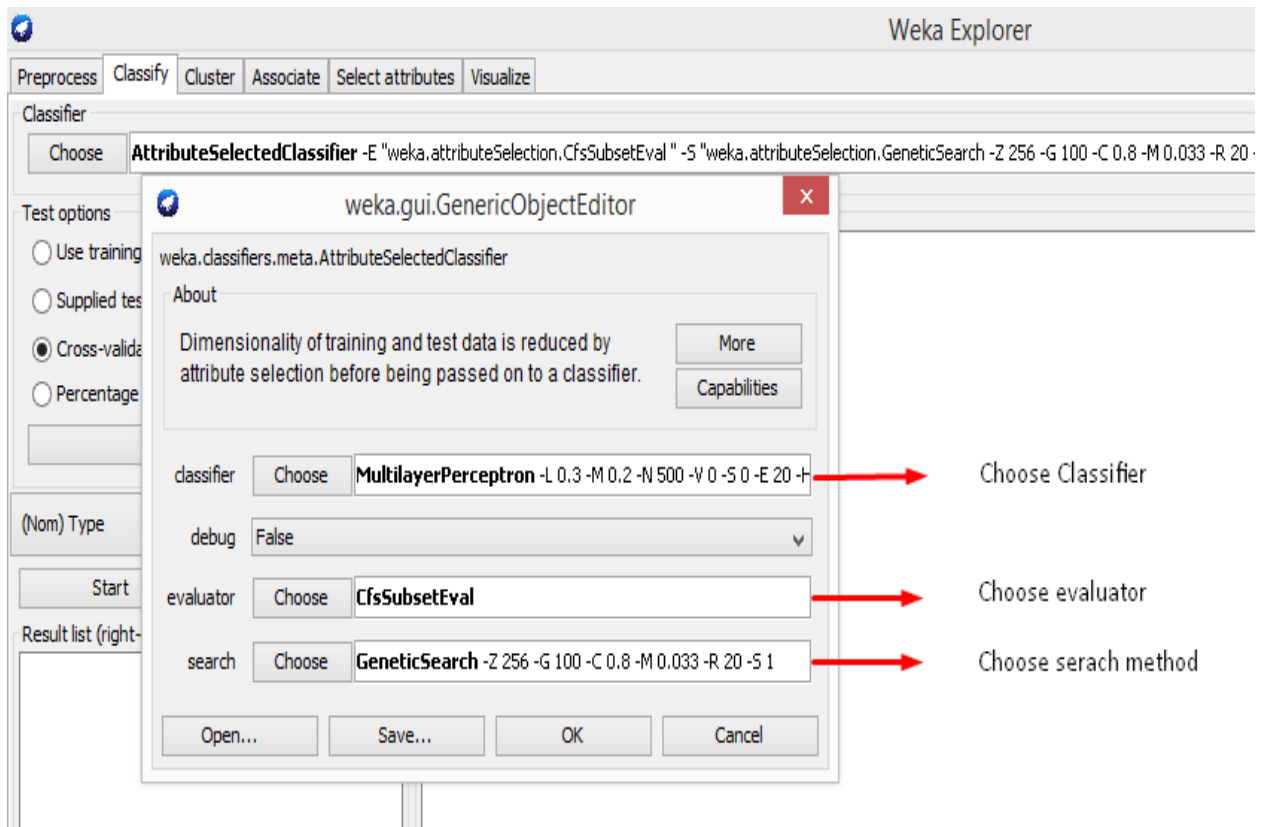


Figure A-2: Selecting parameters for the proposed method

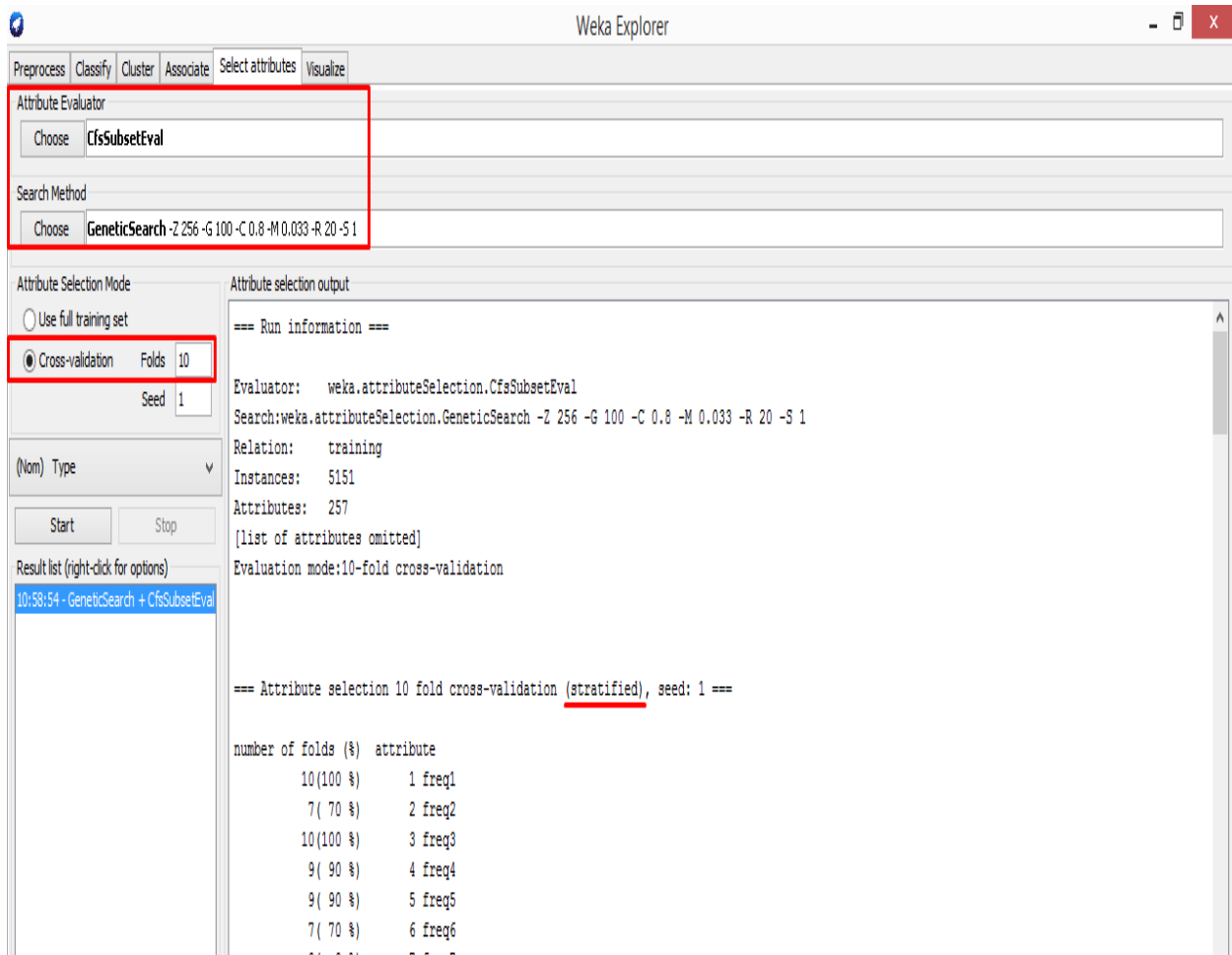


Figure A-3: Another way of selecting attributes using k-fold cross validation

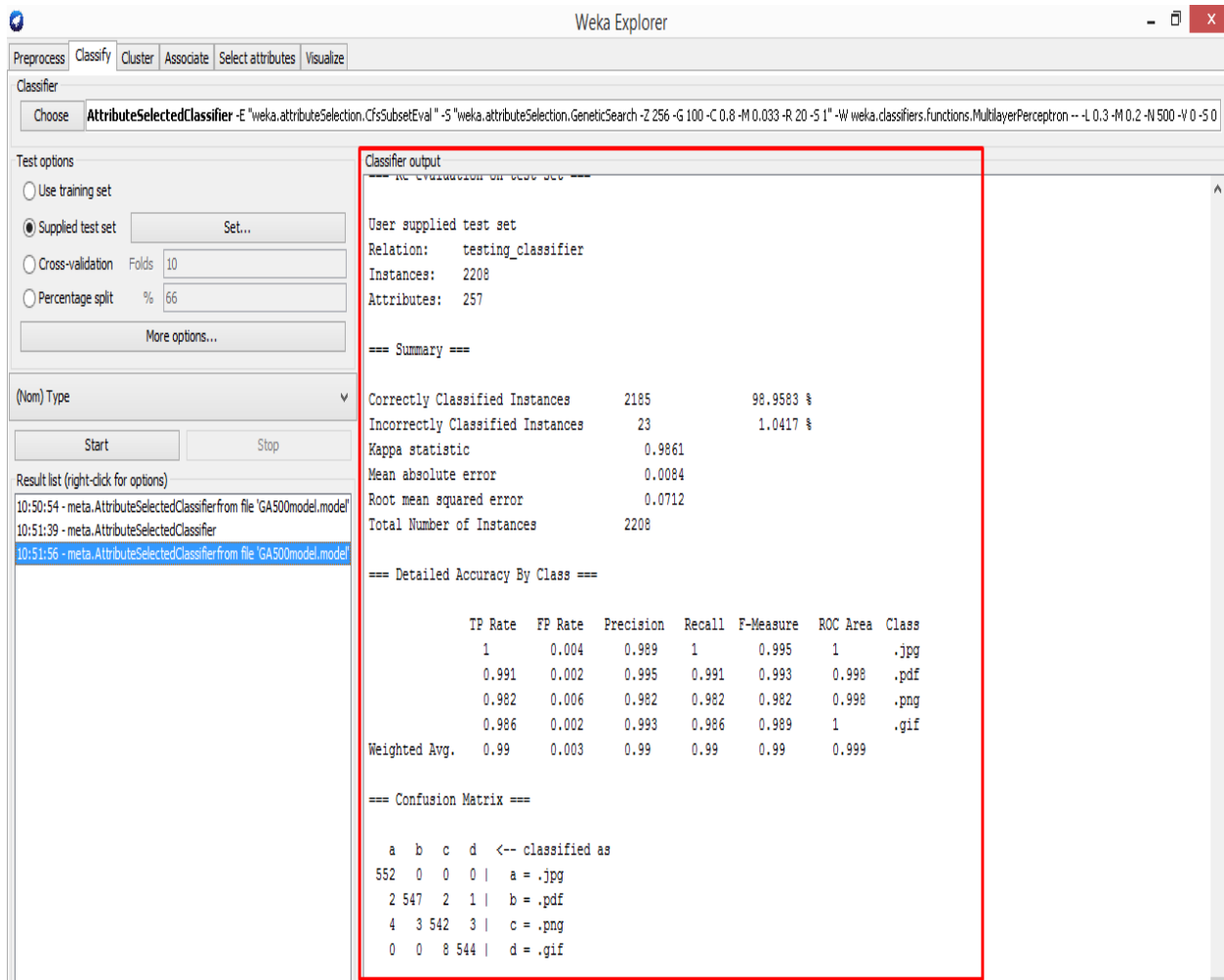


Figure A-4: Classifier results

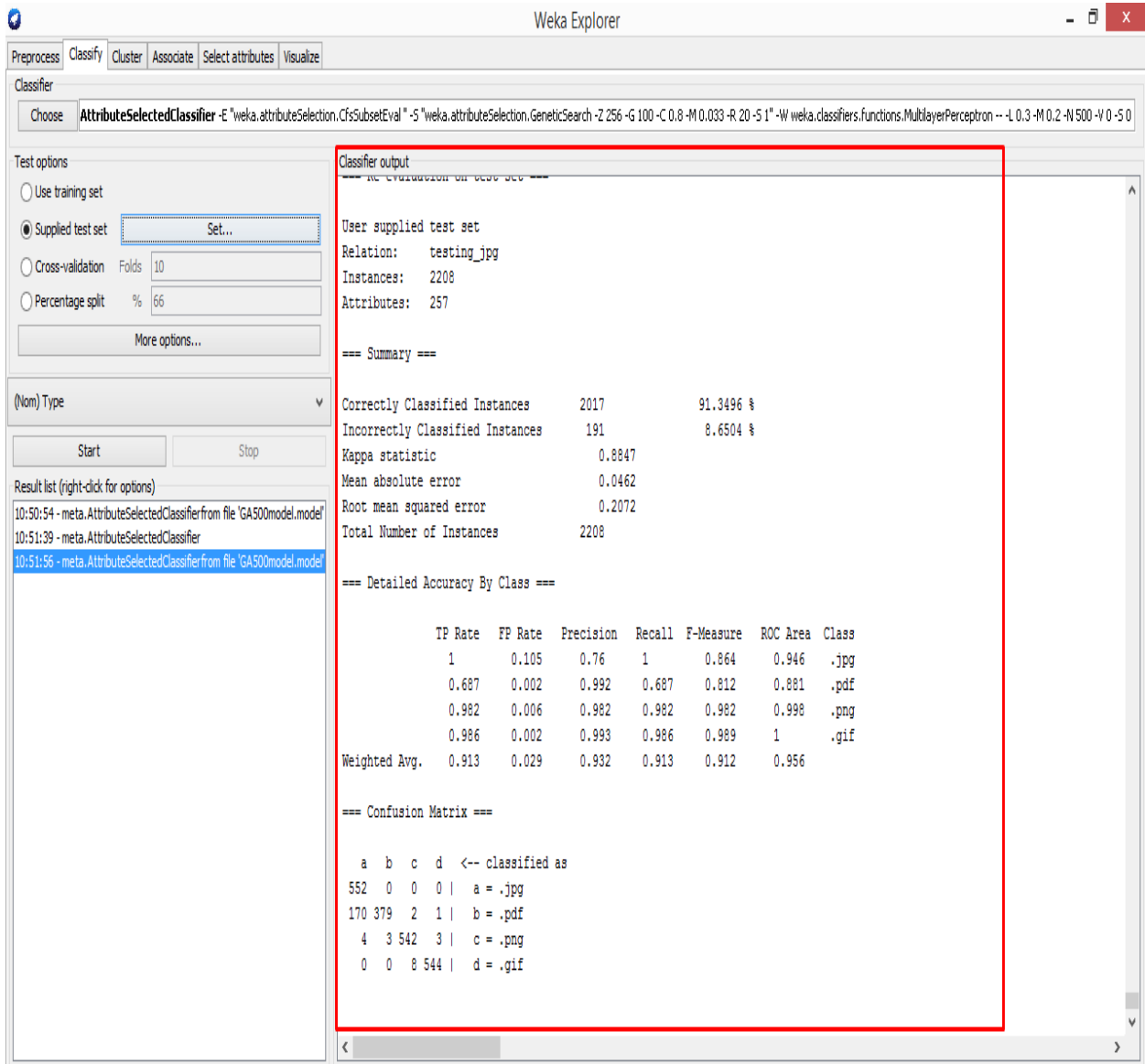


Figure A-5: Classifier results in forged jpg images

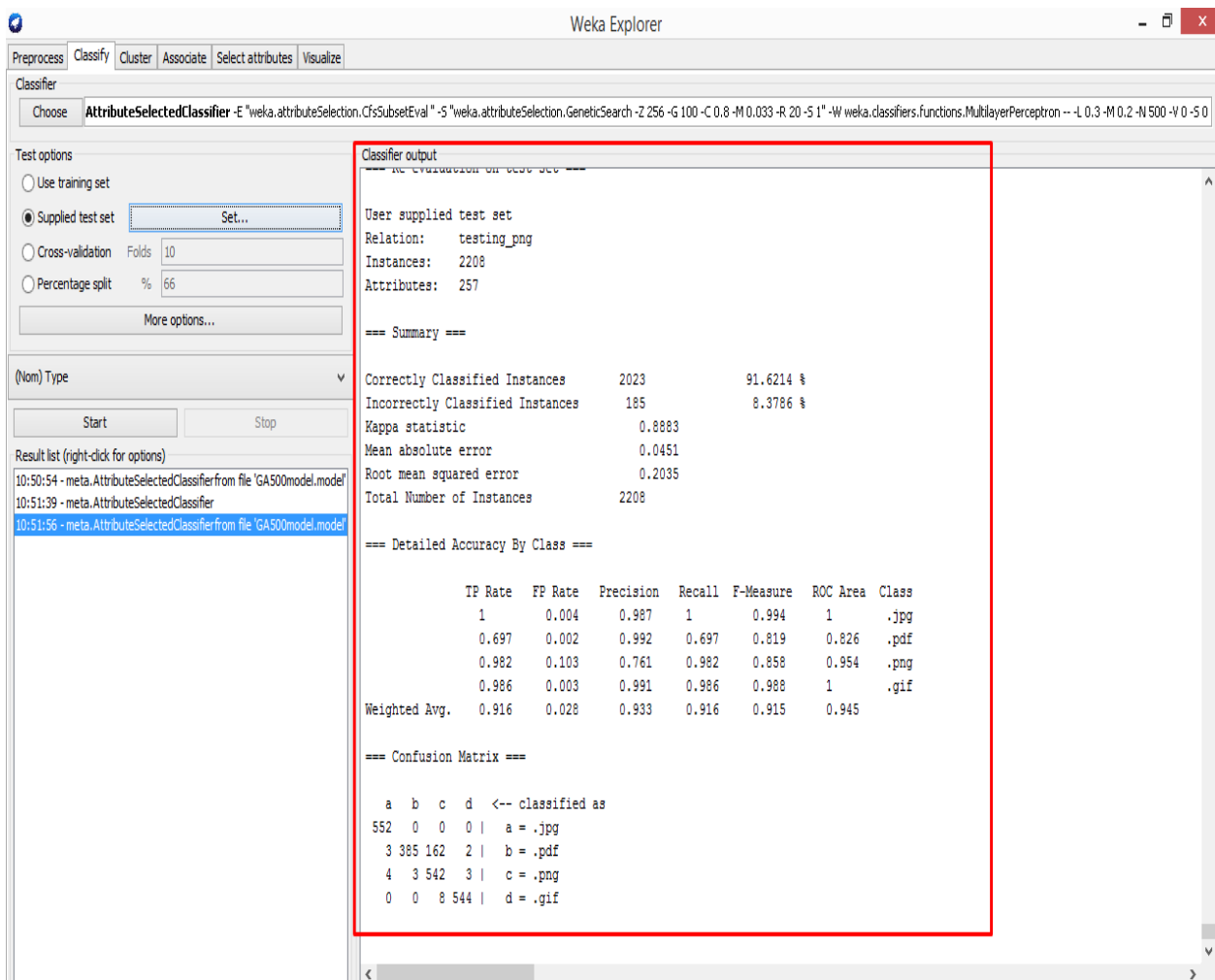


Figure A-6: Classifier results in forged png images

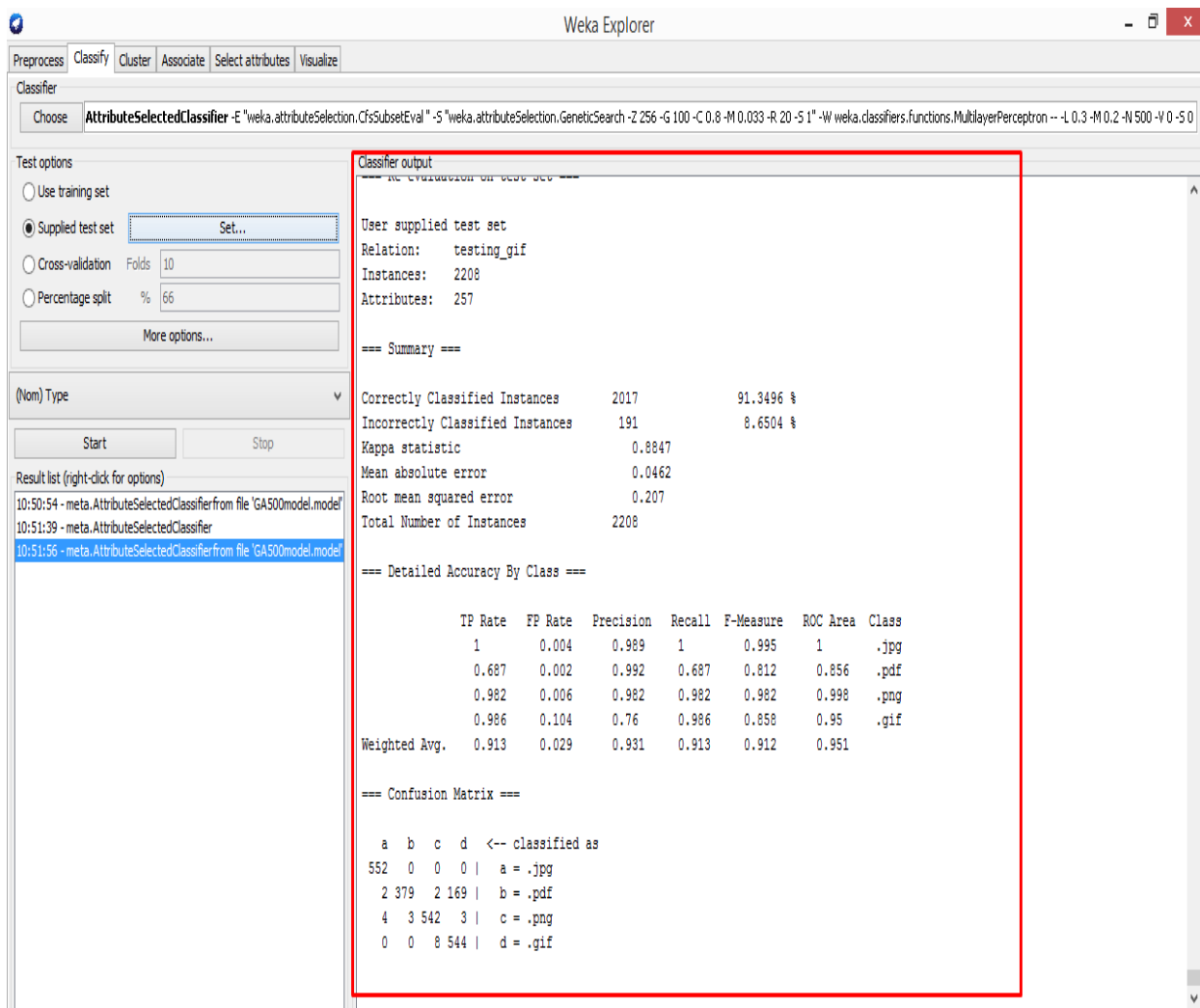


Figure A-7: Classifier results in forged gif images