HarvardX Data Science Professional Certificate
PH125.9x Capstone 2 - Malware Detection

Hugo Aquino

2022-02-27

Contents

,

1 Introduction

Data is the *new fuel* of this era in which many situations can be tracked using records that came from several sources. These data need to be cleaned and organized in a way that with the support of tools such as **R** and techniques such as **machine learning**, the data can talk to show patterns and relationships that allow us to anticipate and therefore take better decisions or protect digital assets.

Data Science is emerging as one of the most important knowledge areas and the **data scientists** are becoming one of the best jobs paid, this role combines computing/programming skills with statistics to analyze raw data and transforming it for its use on an industry. This knowledge area is spreading to be used in areas such as **Cybersecurity**.

According to Cisco, malware is a contraction for **"malicious sofware"** which is any intrusive software developed by **cybercriminals** or **hackers** to steal data and damage or destroy computers and computer systems. Examples of common malware include: *viruses, worms, Trojan viruses, spyware, adware* and *ransomware*.

On **2009, 12.4 million** malware infections were reported, by **2018** the attacks were **812.67 million**. In **2020**, **61%** of organizations experienced malware activity that spread from one employee to another. In **2021** Mimecast found that **61%** of organizations experienced a ransomware attack that led to at least a partial disruption of business operations.

The global cybercrime damage costs are: **6 Trillion USD a year**, **500 Billion USD by month**, **115.4 Billion USD by week**, **16.4 Billion USD by day**, **684.9 Million USD by hour**, **11.4 Million USD by minute** and **190,000 by second**.

Therefore and as a way to continue acquiring the skills needed to become a data scientist, this project will be focused on the evaluation of **four classification algorithms** to determine if a file is a **malware** or **not** using a dataset found on Kaggle called **Malware Analysis Dataset** of files that run **Microsoft Windows** as operating system.

In order to compare the different algorithms, these parameters will be evaluated:

- **Accuracy**, **Sensitivity** and **Specificity** with the highest value obtained.
- **FPR** with the lowest value obtained.

The files required for this project (pdf, rmd, R) are hosted on github.

,

2 Methods

2.1 Libraries

The libraries used for this project are:

```
if(!require(tidyverse)) install.packages(
  "tidyverse", repos = "http://cran.us.r-project.org",dependencies = TRUE)
if(!require(devtools)) install.packages(
  "devtools", repos = "http://cran.us.r-project.org",dependencies = TRUE)
if(!require(readr)) install.packages(
  "readr", repos = "http://cran.us.r-project.org",dependencies = TRUE)
if(!require(data.table)) install.packages(
  "data.table", repos = "http://cran.us.r-project.org",dependencies = TRUE)
if(!require(rvest)) install.packages(
  "rvest", repos = "http://cran.us.r-project.org",dependencies = TRUE)
if(!require(caret)) install.packages(
  "caret", repos = "http://cran.us.r-project.org",dependencies = TRUE)
if(!require(kableExtra)) install.packages(
  "kableExtra", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(corrr)) install.packages(
  "corrr", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(knitr)) install.packages(
  "knitr", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(e1071)) install.packages(
  "e1071", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(class)) install.packages(
  "class", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(tree)) install.packages(
  "tree", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(ISLR)) install.packages(
  "ISLR", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(randomForest)) install.packages(
  "randomForest", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(scales)) install.packages(
  "scales", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(ggplot2)) install.packages(
  "ggplot2", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(ggthemes)) install.packages(
  "ggthemes", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(rpart)) install.packages(
  "rpart", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(rpart.plot)) install.packages(
  "rpart.plot", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(gclus)) install.packages(
  "gclus", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(corrplot)) install.packages(
  "corrplot", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(ROCR)) install.packages(
  "ROCR", repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(tictoc)) install.packages(
  "tictoc", repos = "http://cran.us.r-project.org", dependencies = TRUE)

# Load libraries
library(tidyverse)
```

```
library(devtools)
library(readr)
library(data.table)
library(rvest)
library(caret)
library(kableExtra)
library(corrr)
library(knitr)
library(e1071)
library(class)
library(tree)
library(ISLR)
library(randomForest)
library(scales)
library(ggplot2)
library(ggthemes)
library(rpart)
library(rpart.plot)
library(corrplot)
library(ROCR)
library(tictoc)
```

,

2.2 Data extraction

A dataset was found on Kaggle called **Malware Analysis Dataset** that contains records of files running **Microsoft Windows** as operating system are used on this project and for download the dataset, the **Kaggle API** must be used.

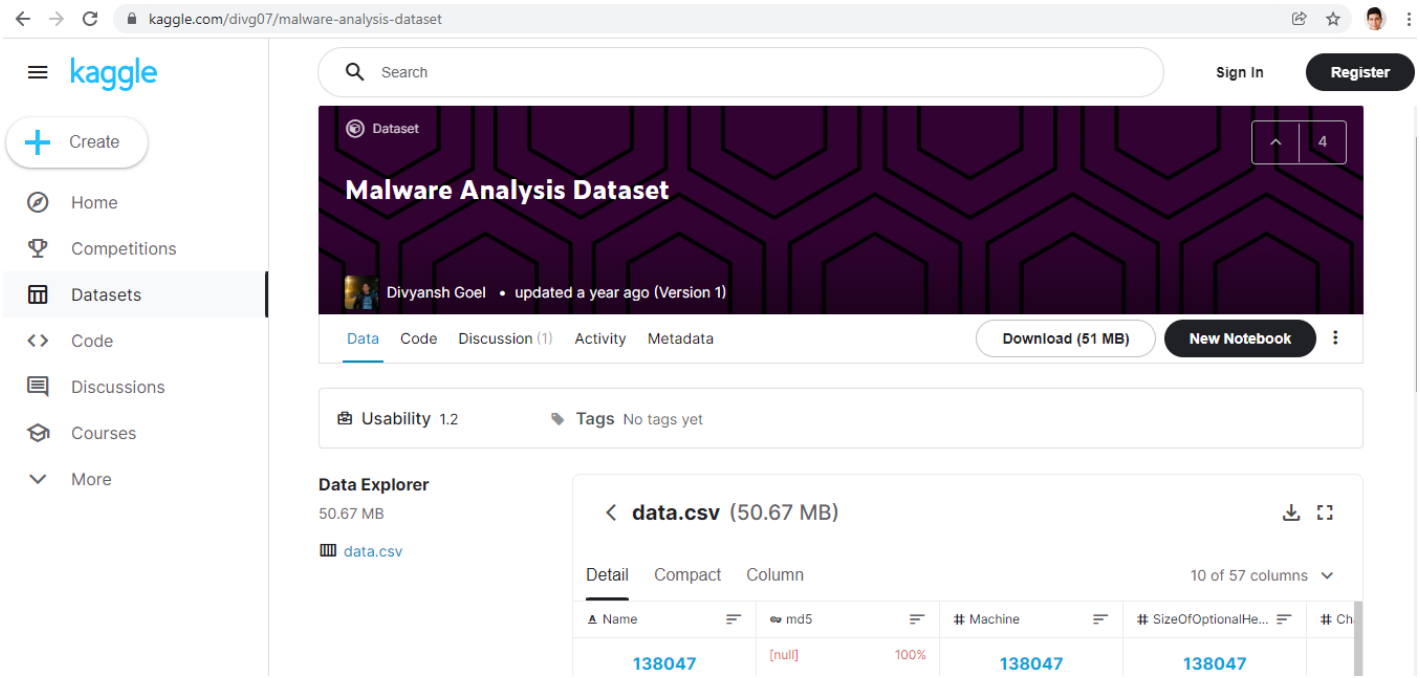

Figure 1: Malware Analysis Dataset used from Kaggle.com

The file originally has to be decompressed and parsed using "|" as delimiter and "." as decimal points.

```
Name|md5|Machine|SizeOfOptionalHeader|Characteristics|MajorLinkerVersion|MinorLinkerVersion|SizeOfCode|SizeOfInitializedData|SizeOfUninitializedData|
AddressOfEntryPoint|BaseOfCode|BaseOfData|ImageBase|SectionAlignment|FileAlignment|MajorOperatingSystemVersion|MinorOperatingSystemVersion|MajorImageVersion|
MinorImageVersion|MajorSubsystemVersion|MinorSubsystemVersion|SizeOfImage|SizeOfHeaders|CheckSum|Subsystem|DllCharacteristics|SizeOfStackReserve|SizeOfStackCommit|
SizeOfHeapReserve|SizeOfHeapCommit|LoaderFlags|NumberOfRvaAndSizes|SectionsNb|SectionsMeanEntropy|SectionsMinEntropy|SectionsMaxEntropy|SectionsMeanRawsize|
SectionsMinRawsize|SectionMaxRawsize|SectionsMeanVirtualsize|SectionsMinVirtualsize|SectionMaxVirtualsize|ImportsNbDLL|ImportsNb|ImportsNbOrdinal|ExportNb|
ResourcesNb|ResourcesMeanEntropy|ResourcesMinEntropy|ResourcesMaxEntropy|ResourcesMeanSize|ResourcesMinSize|ResourcesMaxSize|LoadConfigurationSize|
VersionInformationSize|legitimate
memtest.exe|631ea355665f28d4707448e442fbf5b8|332|224|258|9|0|361984|115712|0|6135|4096|372736|4194304|4096|512|0|0|0|0|1|0|1036288|1024|485887|16|1024|1048576|4096|
1048576|4096|0|16|8|5.7668065537|3.60742957555|7.22105072892|59712.0|1024|325120|126875.875|896|551848|0|0|0|0|4|3.26282271103|2.56884382364|3.53793936419|8797.0|216|
18032|0|16|1
ose.exe|9d10f99a6712e28f8acd5641e3a7ea6b|332|224|3330|9|0|130560|19968|0|81778|4096|143360|771751936|4096|512|5|1|0|0|5|1|159744|1024|188943|2|33088|1048576|4096|
1048576|4096|0|16|4|4.83968793753|2.37352509596|6.56690933416|35584.0|2048|130560|37322.0|1840|130296|7|181|0|0|2|4.2504605579|3.42074425303|5.08017686277|837.0|518|
1156|72|18|1
setup.exe|4d92f518527353c0db88a70fddcfd390|332|224|3330|9|0|517120|621568|0|350896|4096|811008|771751936|4096|512|5|1|0|0|5|1|1150976|1024|1159817|2|32832|1048576|
4096|1048576|4096|0|16|4|6.40955752803|4.88519106848|7.60095678141|273408.0|21504|517120|284498.0|21456|516760|14|235|21|1|11|4.42632398773|2.84644858861|
5.27181275925|31102.2727273|104|270376|72|18|1
DW20.EXE|a41e524f8d45f0074fd07805ff0c9b12|332|224|258|9|0|585728|369152|0|451258|4096|798720|771751936|4096|512|5|1|0|0|5|1|962560|1024|867570|2|33088|1048576|4096|
1048576|4096|0|16|4|6.64173122458|5.64256492784|7.59492880283|207872.0|15360|585728|238502.0|15208|585488|15|360|6|1|10|4.36429082616|2.66931388802|6.40071950185|
1457.0|90|4264|72|18|1
dwtrig20.exe|c87e561258f2f8650cef999bf643a731|332|224|258|9|0|294912|247296|0|217381|4096|536576|771751936|4096|512|5|1|0|0|5|1|552960|1024|579287|2|33088|1048576|
4096|1048576|4096|0|16|4|6.25268442524|4.18228226649|7.60830453625|128128.0|2560|294912|135350.0|2320|294816|10|194|4|1|2|4.30610018148|3.42159769887|5.19060266409|
1074.5|849|1300|72|18|1
```

Figure 2: Raw data

,

2.3 PE (Portable Executable) file

The structure of a **Windows** PE file is displayed in the next figure.

PE format is a **data structure** that tells Windows Operating System loader what information is required to manage the wrapped executable code. These **PE data structures** include:

- **DOS Header**
- **DOS Stub**
- **PE File Header**
- **Image Optional Header**
- **Section Table**
- **Data Dictionaries**
- **Sections**

On **Appendix A** a detailed explanation of the fields is provided.

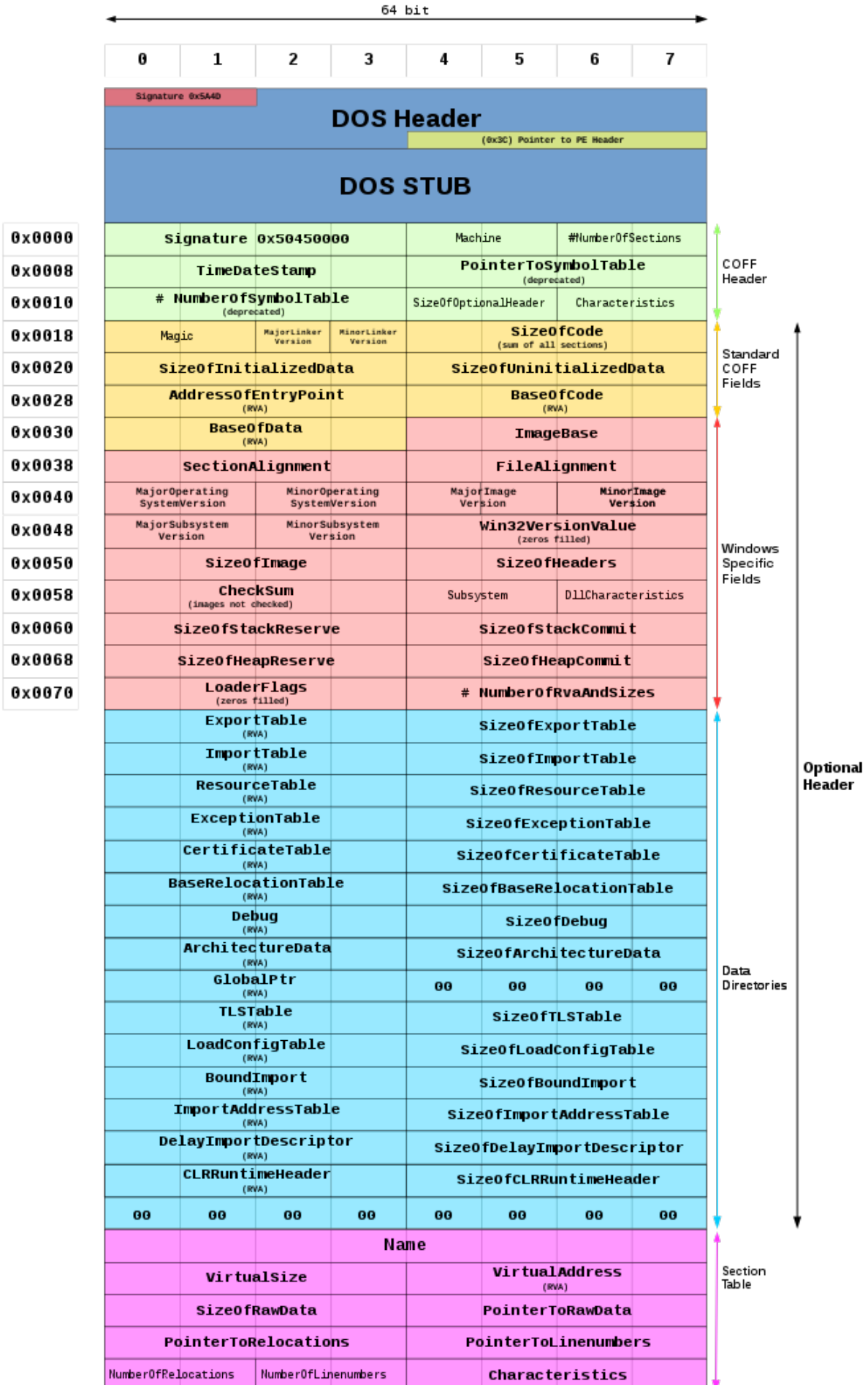

Figure 3: PE file structure

2.4 About data_loaded dataset

The dataset was named as **data_loaded** and is of the type **data.frame**. This dataset has **138,047** observations and **57** columns. Additionally these observations are records of processes running on a information technology infrastructure (server, desktop, laptop, tablet) using **Windows** as operating system. Technically these observations belongs to a **PE (Portable Executable)** file respectively.

The column **legitimate** is very important because its value indicate if the program is **valid** (value **"1"**) or not (**malware**, value **"0"**), the next table shows the amount of records, and for this project this variable will be our **predictor**.

Table 1: Amount of records on the dataset

Type	# records
Malware	96,724
Not malware	41,323
Total	138,047

The **entropy** for a malware and not malware program is displayed in the following histogram. The range of values a file's entropy must come in as per **Shanon's algorithm** is **0 to 8**. So, when its value is zero, one can say the outcome is certain. On contrary, when its value is 8, the outcome is most unpredictable it could be.

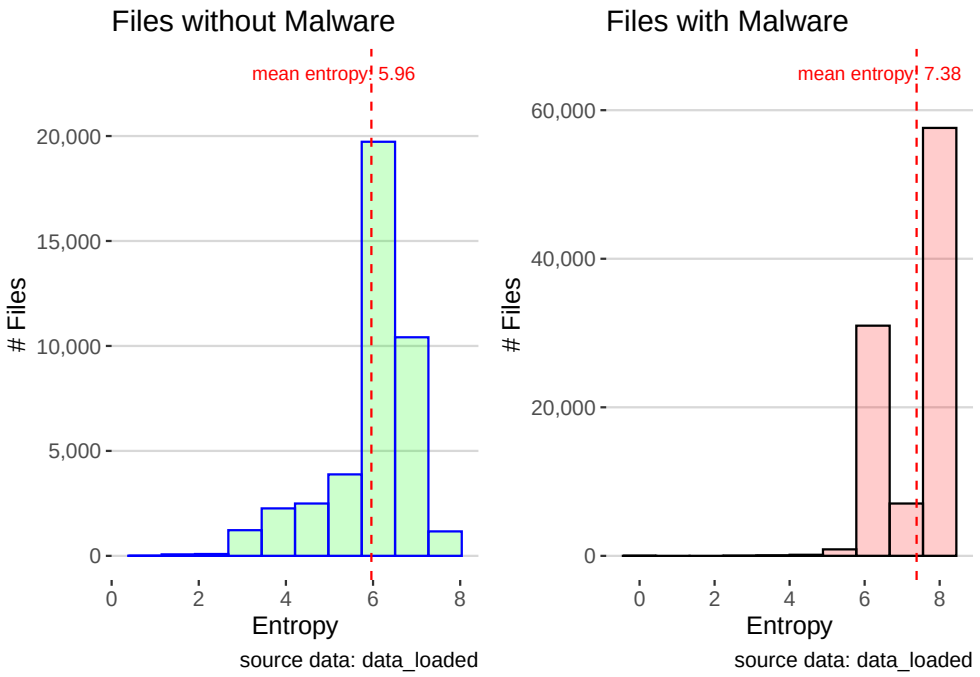

Figure 4: Files´ entropy

Now a correlation analysis between the variables will be done against the predictor (legitimate), being the results displayed in the following table

Table 2: Correlations focused on "legitimate" field

term	legitimate
Machine	0.5488345
SizeOfOptionalHeader	0.5474981
Subsystem	0.5143519
MajorSubsystemVersion	0.3803931
VersionInformationSize	0.3796460

Table 2: Correlations focused on "legitimate" field *(continued)*

term	legitimate
ResourcesMinEntropy	0.2991116
Characteristics	0.2219559
ExportNb	0.1344077
ImportsNbOrdinal	0.1281115
FileAlignment	0.1251687
ImportsNb	0.1164149
ResourcesNb	0.0904054
MajorImageVersion	0.0844102
MinorImageVersion	0.0832200
SectionsMinRawsize	0.0593461
SectionsMinVirtualsize	0.0564659
ImportsNbDLL	0.0383947
SizeOfCode	0.0174761
MajorLinkerVersion	0.0173197
SizeOfHeaders	0.0101246
ImageBase	0.0082445
MajorOperatingSystemVersion	0.0024021
SectionsMeanVirtualsize	0.0017341
SectionsMeanRawsize	0.0011748
AddressOfEntryPoint	-0.0001337
SectionMaxRawsize	-0.0007899
BaseOfData	-0.0011360
MinorSubsystemVersion	-0.0012132
SectionMaxVirtualsize	-0.0013323
MinorOperatingSystemVersion	-0.0017023
ResourcesMinSize	-0.0017744
SectionAlignment	-0.0024294
SizeOfHeapCommit	-0.0025063
SizeOfImage	-0.0026031
LoaderFlags	-0.0026487
SizeOfStackCommit	-0.0032258
NumberOfRvaAndSizes	-0.0035233
ResourcesMeanSize	-0.0038240
SizeOfUninitializedData	-0.0039971
SizeOfInitializedData	-0.0049576
ResourcesMaxSize	-0.0055291
BaseOfCode	-0.0062322
LoadConfigurationSize	-0.0116658
MinorLinkerVersion	-0.1466516
SectionsMinEntropy	-0.1528402
SizeOfHeapReserve	-0.1562604
CheckSum	-0.1953287
ResourcesMeanEntropy	-0.2024325
SectionsNb	-0.2077818
SectionsMeanEntropy	-0.3439326
ResourcesMaxEntropy	-0.3928552
SizeOfStackReserve	-0.5216416
SectionsMaxEntropy	-0.6242289
DllCharacteristics	-0.6301770

The next table summarize the information about how many variables correspond to a correlation factor. For the rest of

this project the correlation factor to be used will be **0.10**.

Table 3: Amount of variables based on a correlation factor

Correlation factor	# variables	Variables´ name
Up 0	24	Machine , SizeOfOptionalHeader , Subsystem , MajorSubsystemVersion , VersionInformationSize , ResourcesMinEntropy , Characteristics , ExportNb , ImportsNbOrdinal , FileAlignment , ImportsNb , ResourcesNb , MajorImageVersion , MinorImageVersion , SectionsMinRawsize , SectionsMinVirtualsize , ImportsNbDLL , SizeOfCode , MajorLinkerVersion , SizeOfHeaders , ImageBase , MajorOperatingSystemVersion, SectionsMeanVirtualsize , SectionsMeanRawsize
Below 0	30	AddressOfEntryPoint , SectionMaxRawsize , BaseOfData , MinorSubsystemVersion , SectionMaxVirtualsize , MinorOperatingSystemVersion, ResourcesMinSize , SectionAlignment , SizeOfHeapCommit , SizeOfImage , LoaderFlags , SizeOfStackCommit , NumberOfRvaAndSizes , ResourcesMeanSize , SizeOfUninitializedData , SizeOfInitializedData , ResourcesMaxSize , BaseOfCode , LoadConfigurationSize , MinorLinkerVersion , SectionsMinEntropy , SizeOfHeapReserve , CheckSum , ResourcesMeanEntropy , SectionsNb , SectionsMeanEntropy , ResourcesMaxEntropy , SizeOfStackReserve , SectionsMaxEntropy , DllCharacteristics
Up 0.10	11	Machine , SizeOfOptionalHeader , Subsystem , MajorSubsystemVersion , VersionInformationSize, ResourcesMinEntropy , Characteristics , ExportNb , ImportsNbOrdinal , FileAlignment , ImportsNb

2.5 About data_filtrated dataset

A new dataset **data_filtrated** will be created with variables which correlation factor are bigger than **0.10**.

This dataset is of the type **data.frame**, which has **138,047** observations and **12** columns.

A graph of the variables sorted based on correlation is the following.

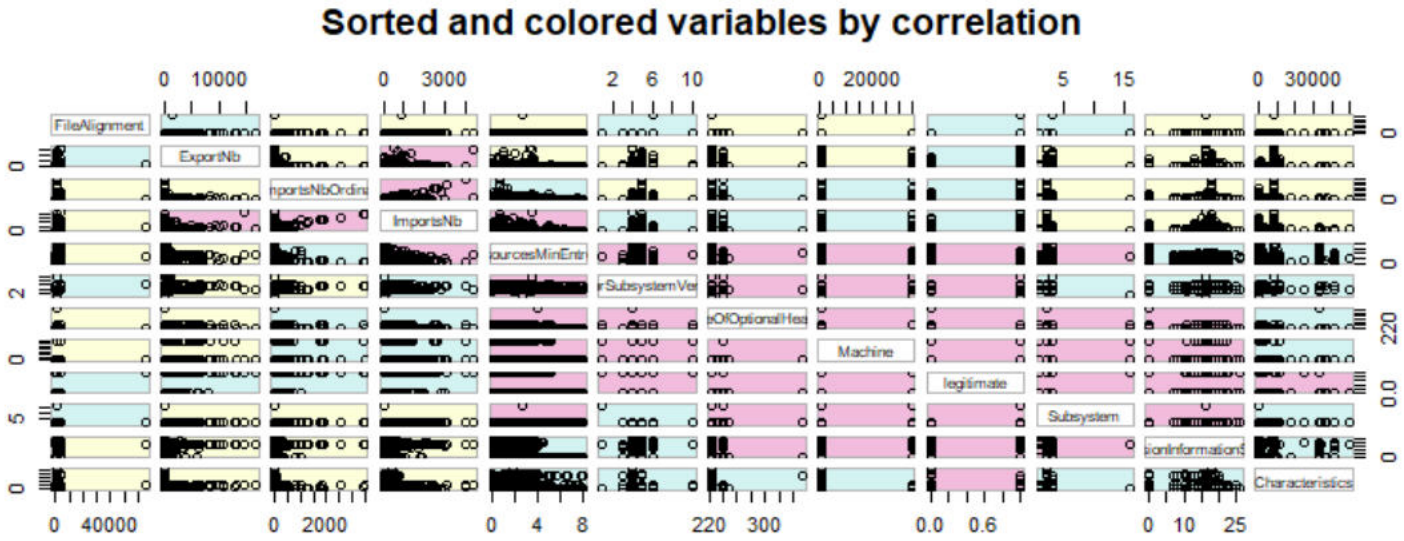

Figure 5: Sorted colored variables by correlation

Also a correlation map of these variables is generated

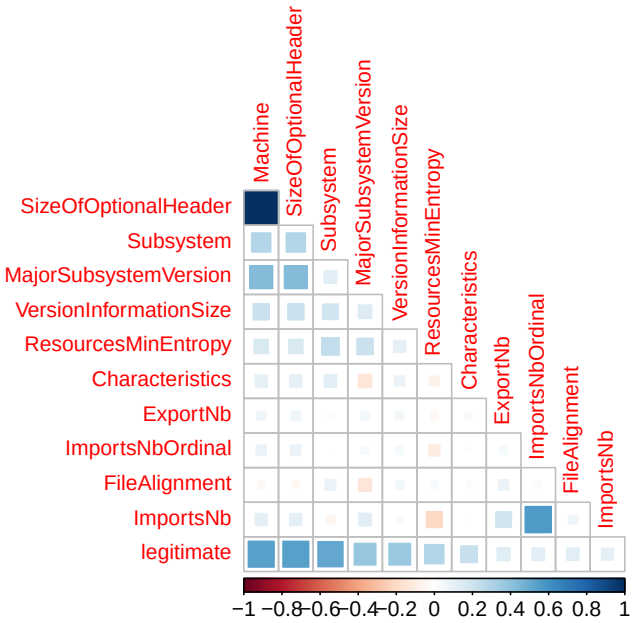

Figure 6: Correlation map

2.5.1 About Machine

Based on the **Machine** contained in the dataset the distribution is:

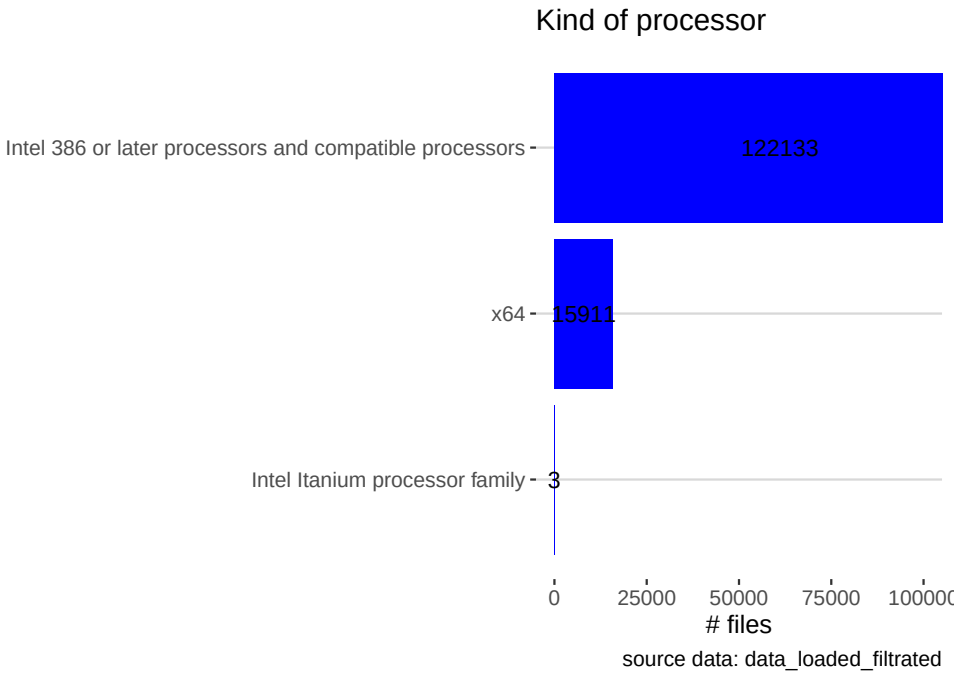

Figure 7: Kind of processors

The maximum amount of processors are **122,133** (**88.47%** of the records on the dataset) being of the type **"Intel 386 or later processors and compatible processors"**.

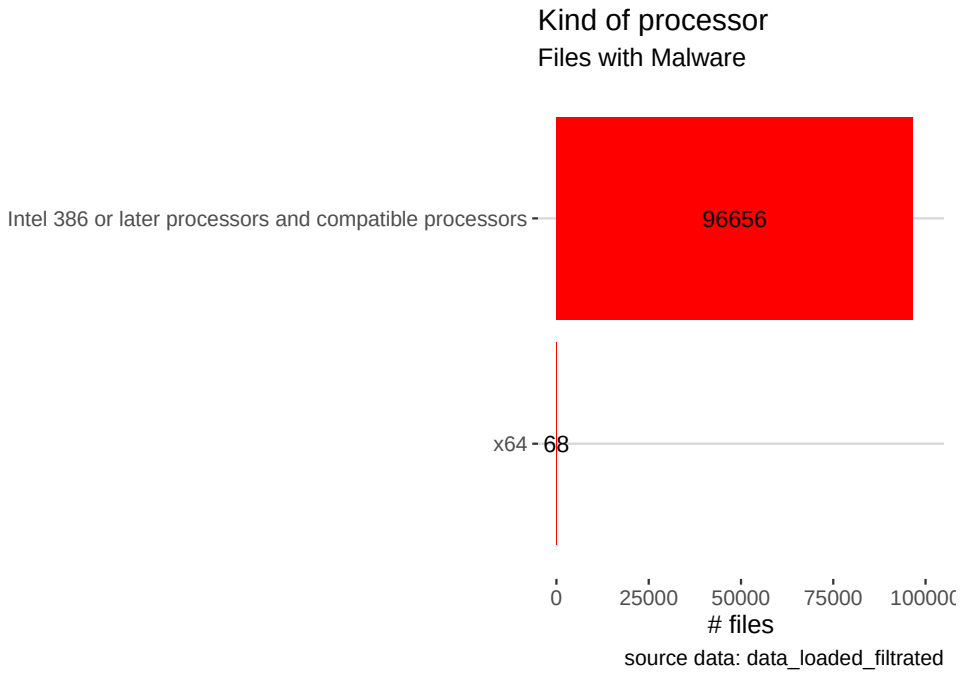

Figure 8: Machine - Files with malware

The maximum amount of processors with files **with malware** are **96,656** (**70.02%** of the records on the dataset) being of the type **"Intel 386 or later processors and compatible processors"**.

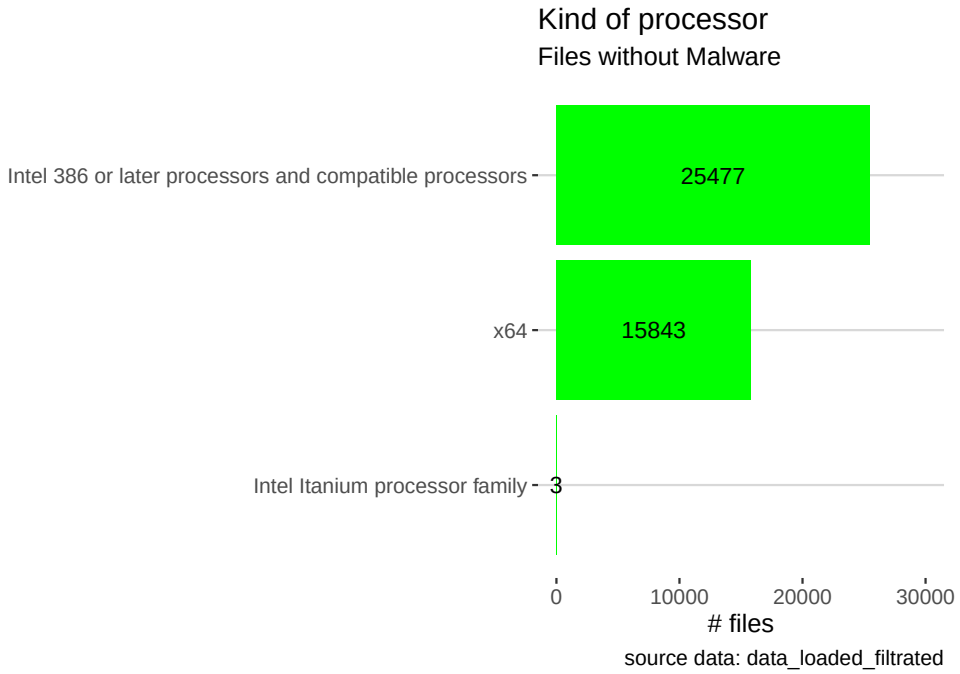

Figure 9: Machine - Files without malware

The maximum amount of processors with files **without malware** are **25,477** (**18.46%** of the records on the dataset) being of the type **"Intel 386 or later processors and compatible processors"**.

2.5.2 About SizeOfOptionalHeader

Based on the **SizeOfOptionalHeader** contained in the dataset the distribution is:

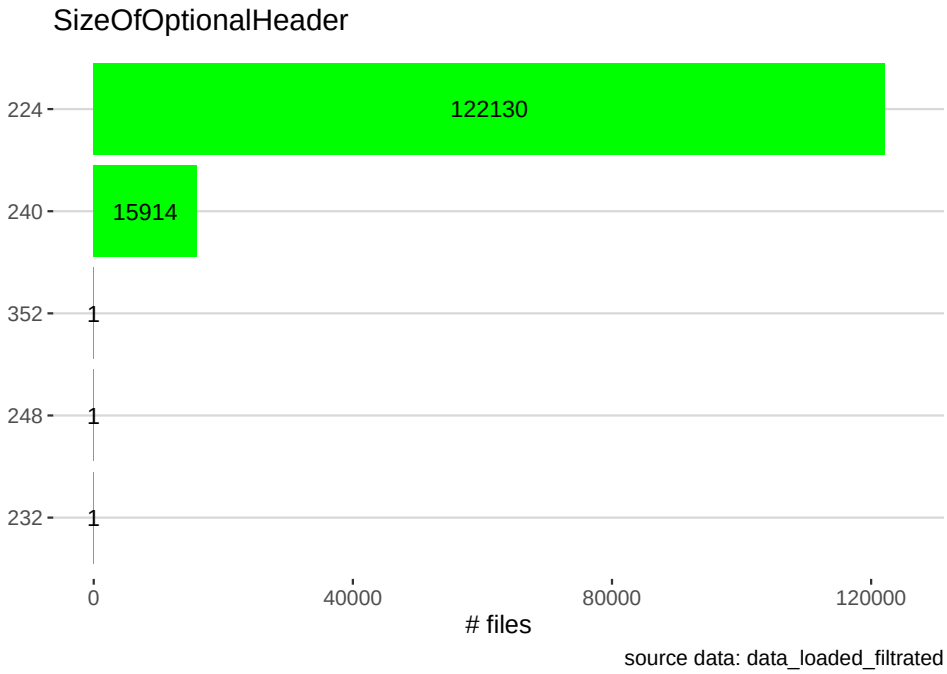

Figure 10: SizeofOptionalHeader

The **SizeOfOptionalHeader** with the highest amount of files is **224** with a total of **122,130** (**88.47%** of the records on the dataset) files.

Figure 11: Size of Optional Header - Files with and without Malware

For files **without malware** the **SizeOfOptionalHeader** with the highest amount is **224** with a total of **25,477** (**18.46%** of the records on the dataset) files.

While files **with malware** the **SizeOfOptionalHeader** with the highest amount is **224** with a total of **96,653** (**70.01%** of the records on the dataset) files.

2.5.3 About Subsystem

Based on the **Subsystem** contained in the dataset the distribution is:

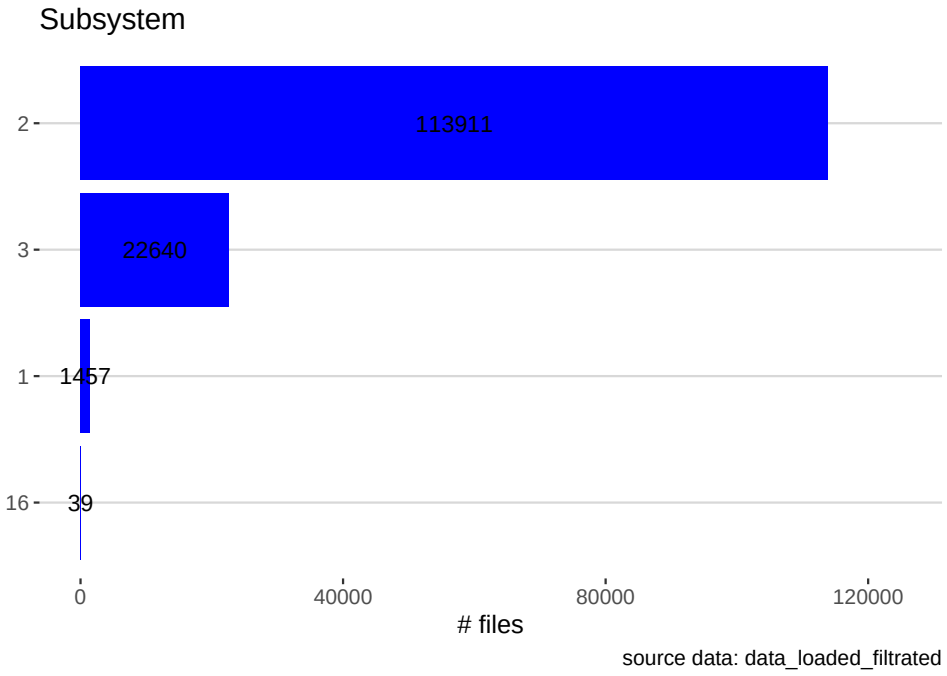

Figure 12: Subsystem

The **Subsystem** with the highest amount of files is **2** with a total of **113,911** (**82.52%** of the records on the dataset) files.

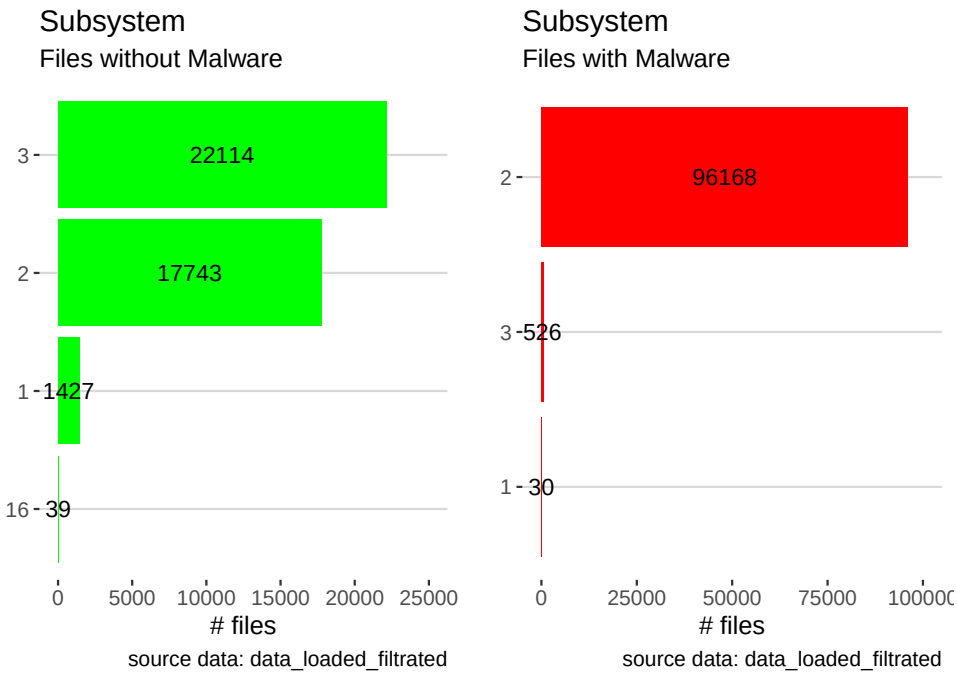

Figure 13: Subsystem - Files with and without Malware

For files **without malware** the **Subsystem** with the highest amount is **3** with a total of **22,114** (**16.02%** of the records on the dataset) files. The description of the Windows subsystem required to run the image is **"The Windows character subsystem"**.

While files **with malware** the **Subsystem** with the highest amount is **2** with a total of **96,168** (**69.66%** of the records on the dataset) files. The description of the Windows subsystem required to run the image is **"The Windows graphical user interface (GUI) subsystem"**.

2.5.4 About MajorSubsystemVersion

Based on the **MajorSubsystemVersion** contained in the dataset the distribution is:

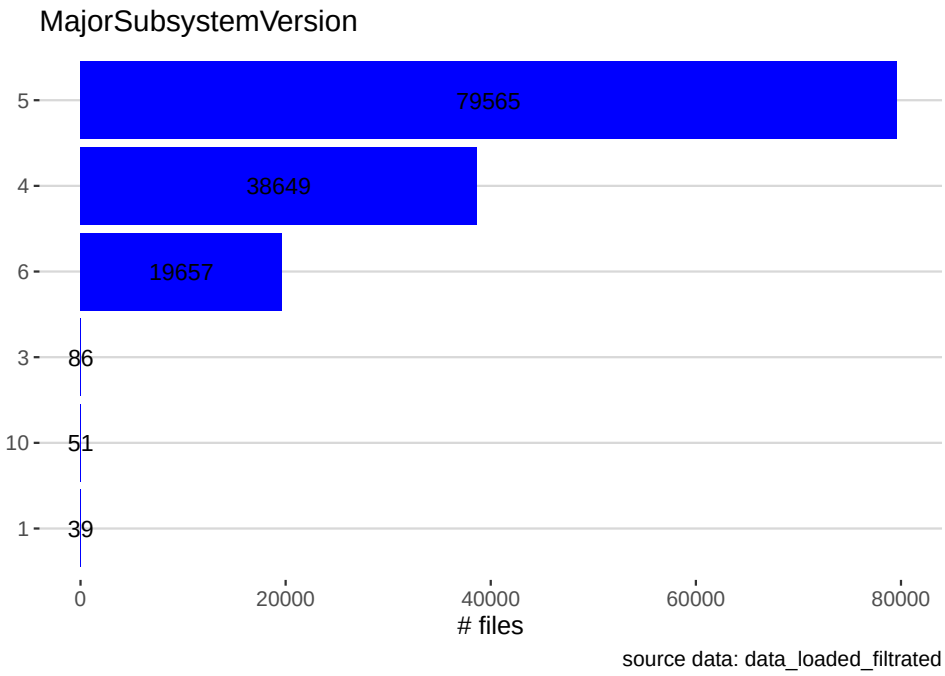

Figure 14: MajorSubsystemVersion

The **MajorSubsystemVersion** with the highest amount of files is **5** with a total of **79,565** (**57.64%** of the records on the dataset) files.

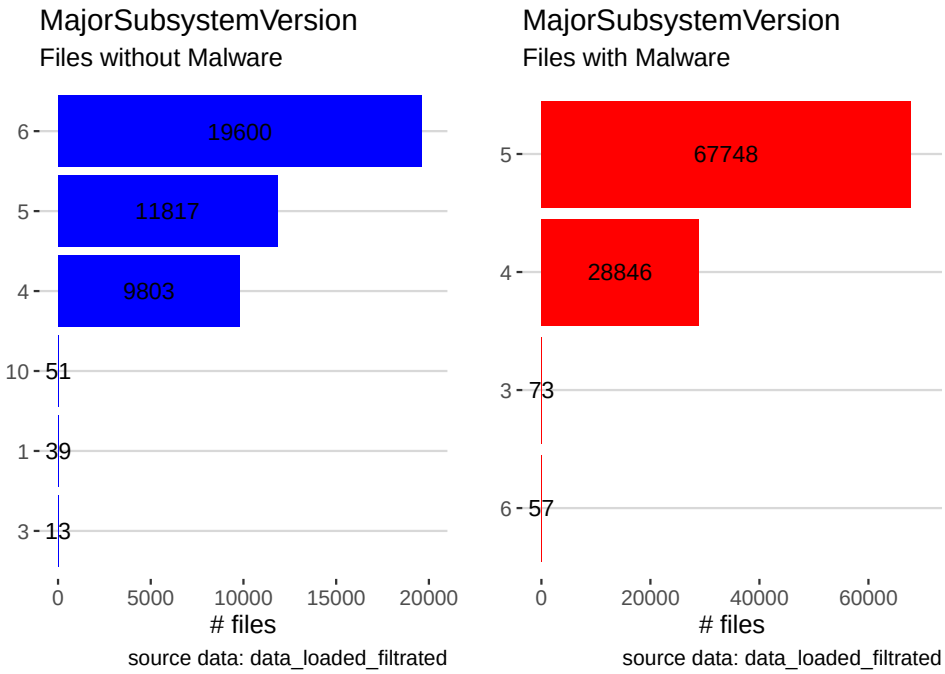

Figure 15: MajorSubsystemVersion - Files with and without Malware

For files **without malware** the **MajorSubsystemVersion** with the highest amount is **6** with a total of **19,600** (**14.20%** of the records on the dataset) files.

While files **with malware** the **MajorSubsystemVersion** with the highest amount is **5** with a total of **67,748** (**49.08%** of the records on the dataset) files.

2.5.5 About VersionInformationSize

Based on the **VersionInformationSize** contained in the dataset the distribution is:

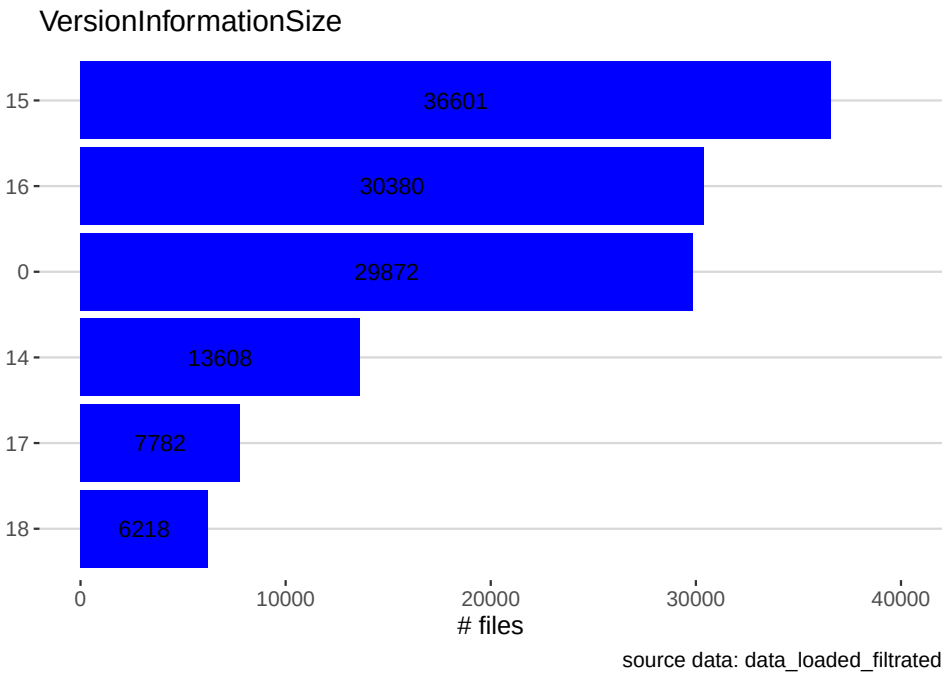

Figure 16: VersionInformationSize

The **VersionInformationSize** with the highest amount of files is **15** with a total of **36,601** (**26.51%** of the records on the dataset) files.

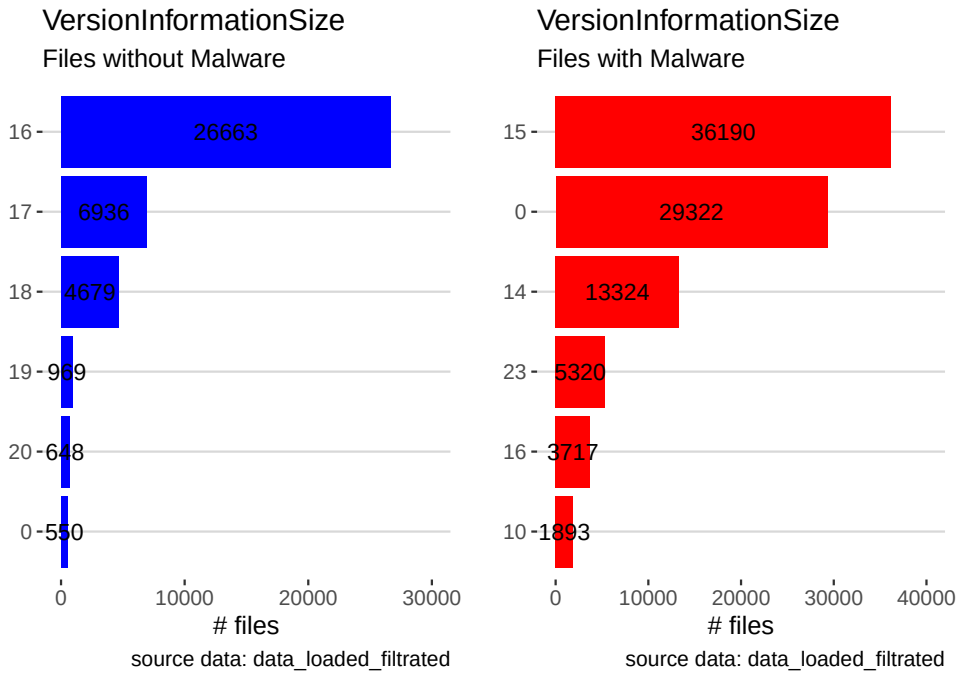

Figure 17: VersionInformationSize - Files with and without Malware

For files **without malware** the **VersionInformationSize** with the highest amount is **16** with a total of **26,663** (**19.31%** of the records on the dataset) files.

While files **with malware** the **VersionInformationSize** with the highest amount is **15** with a total of **36,190** (**26.22%** of the records on the dataset) files.

2.5.6 About ResourcesMinEntropy

Based on the **ResourcesMinEntropy** contained in the dataset the distribution is:

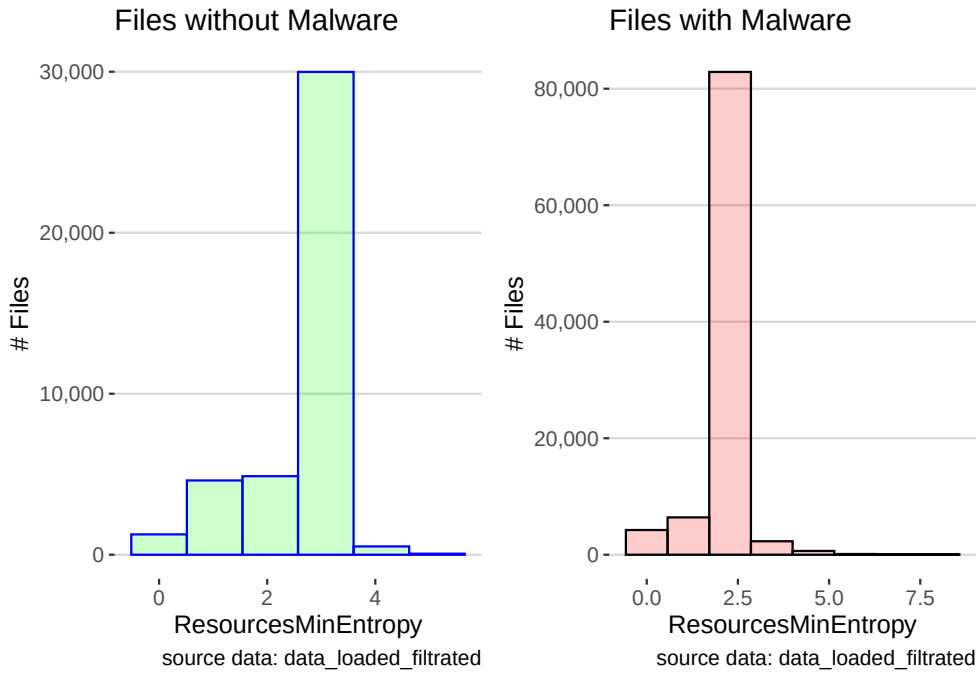

Figure 18: ResourcesMinEntropy - Files with and without Malware

For files **without malware** the **ResourcesMinEntropy** according to the histogram with the highest number of files is between **2.5 to 3.5**. While files **with malware** the **ResourcesMinEntropy** according to the histogram with the highest number of files is around **2.5**.

2.5.7 About Characteristics

Based on the **Characteristics** contained in the dataset the distribution is:

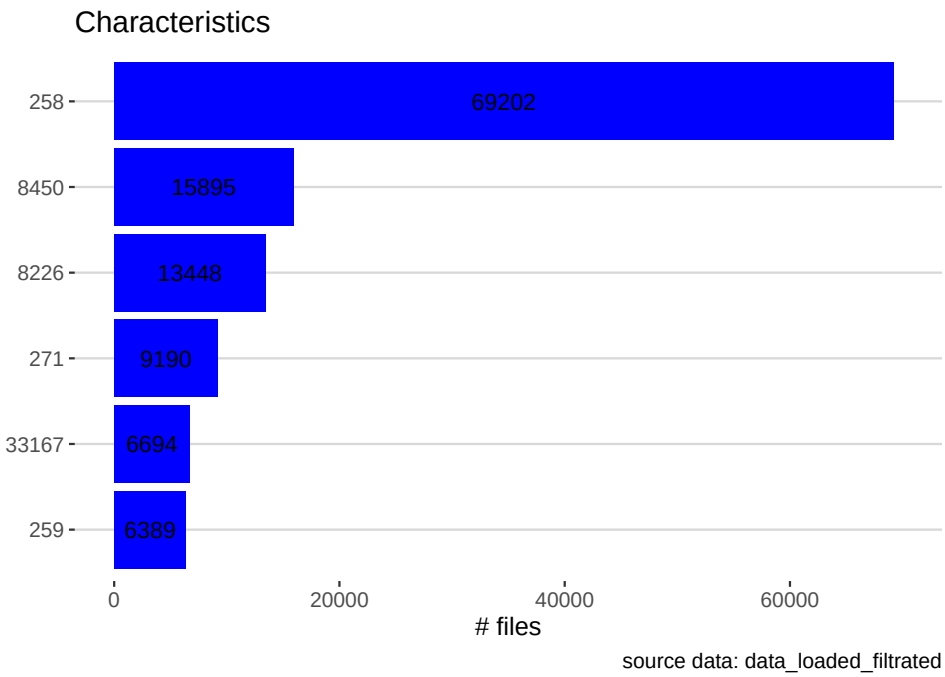

Figure 19: Characteristics

The **Characteristics** with the highest amount of files is **258** with a total of **69,202** (**50.13%** of the records on the dataset)

files.

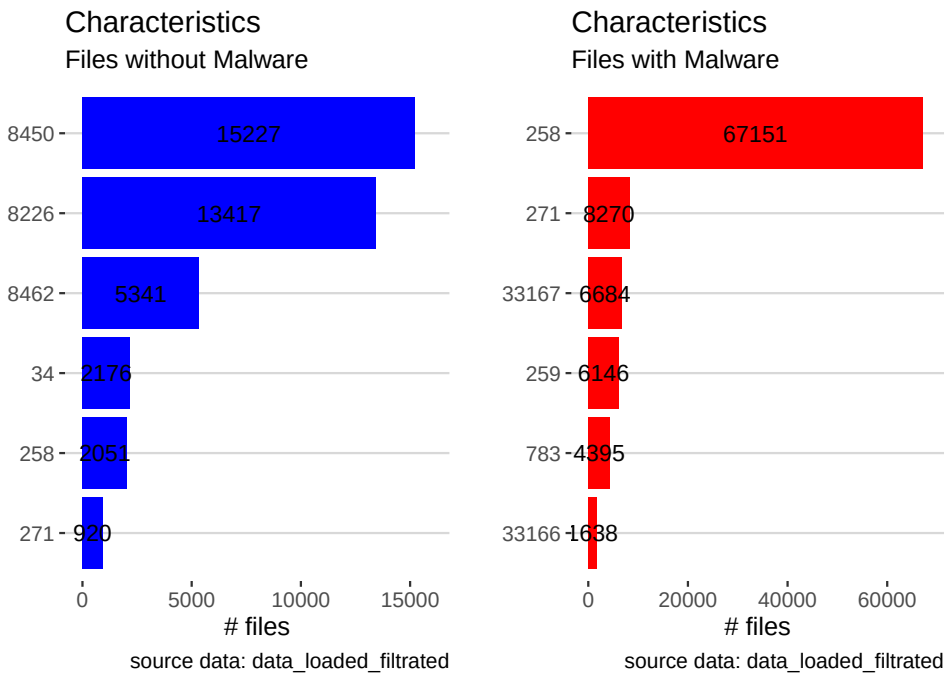

Figure 20: Characteristics - Files with and without Malware

For files **without malware** the **Characteristics** with the highest amount is **8450** with a total of **15,227** (**11.03%** of the records on the dataset) files. The description of the **Characteristics** that contains flags that indicate attributes of the object or image file is **"The image file is a dynamic-link library (DLL). Such files are considered executable files for almost all purposes, although they cannot be directly run."**.

While files **with malware** the **Characteristics** with the highest amount is **258** with a total of **67,151** (**48.64%** of the records on the dataset) files. The description of the **Characteristics** is **"Machine is based on a 32-bit-word architecture."**.

2.5.8 About ExportNb

Based on the **ExportNb** contained in the dataset the distribution is:

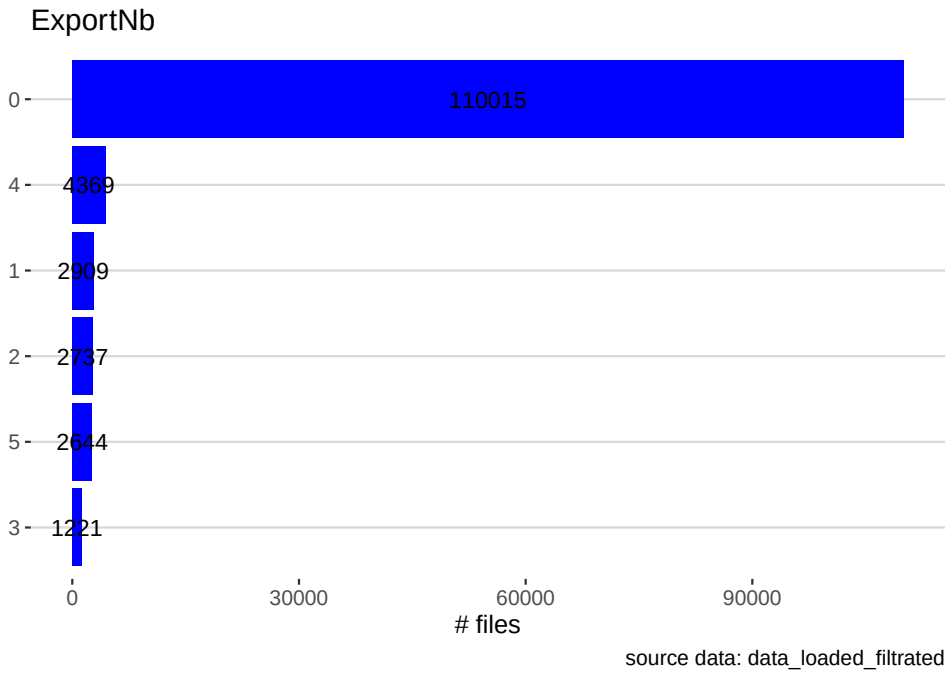

Figure 21: ExportNb

The **ExportNb** with the highest amount of files is **0** with a total of **110,015** (**79.69%** of the records on the dataset) files.

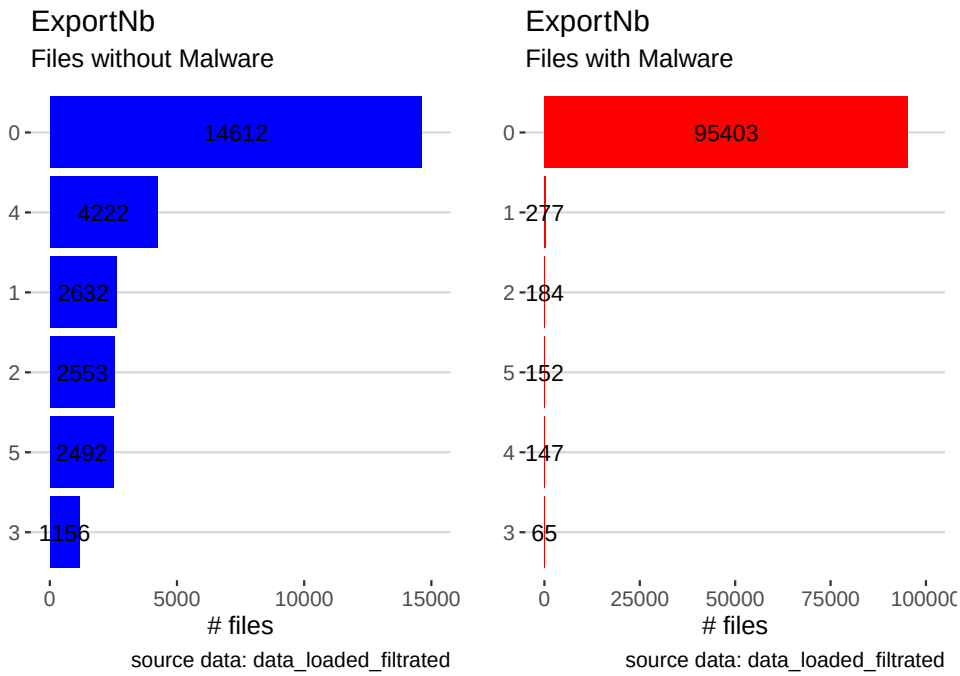

Figure 22: ExportNb - Files with and without Malware

For files **without malware** the **ExportNb** with the highest amount is **0** with a total of **14,612** (**10.58%** of the records on the dataset) files.

While files **with malware** the **ExportNb** with the highest amount is **0** with a total of **95,403** (**69.11%** of the records on the dataset) files.

2.5.9 About ImportsNbOrdinal

Based on the **ImportsNbOrdinal** contained in the dataset the distribution is:

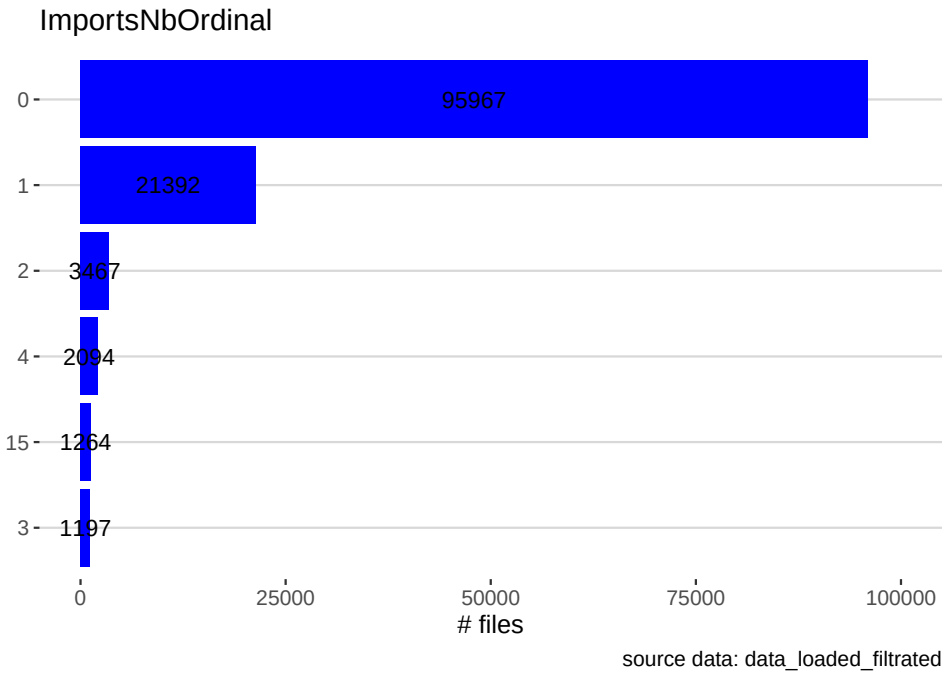

Figure 23: ImportsNbOrdinal

The **ImportsNbOrdinal** with the highest amount of files is **0** with a total of **95,967** (**69.52%** of the records on the dataset) files.

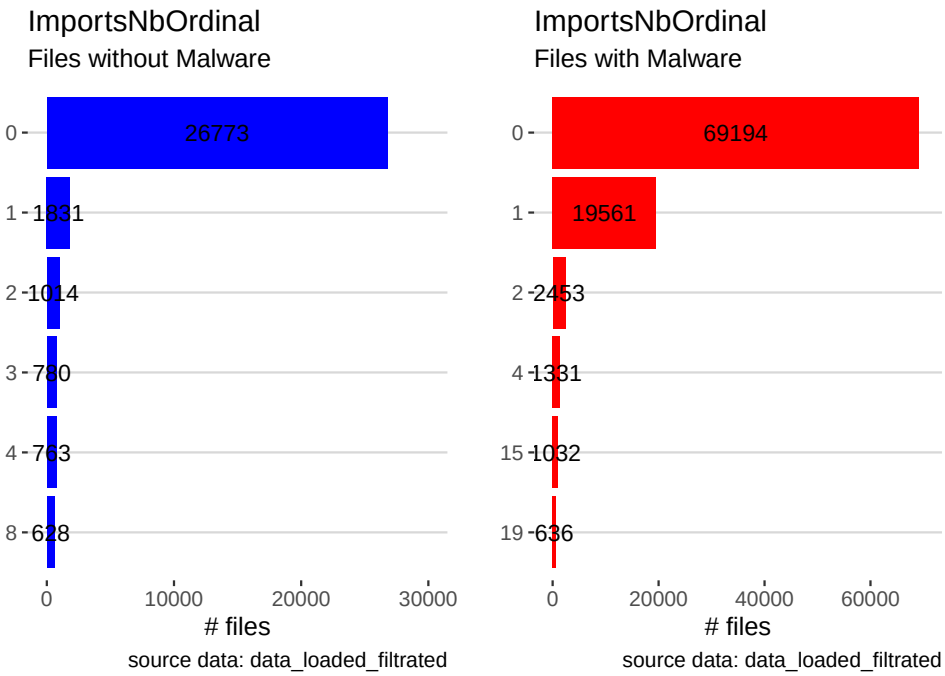

Figure 24: ImportsNbOrdinal - Files with and without Malware

For files **without malware** the **ImportsNbOrdinal** with the highest amount is **0** with a total of **26,773** (**19.39%** of the records on the dataset) files.

While files **with malware** the **ImportsNbOrdinal** with the highest amount is **0** with a total of **69,194** (**50.12%** of the records on the dataset) files.

2.5.10 About FileAlignment

Based on the **FileAlignment** contained in the dataset the distribution is:

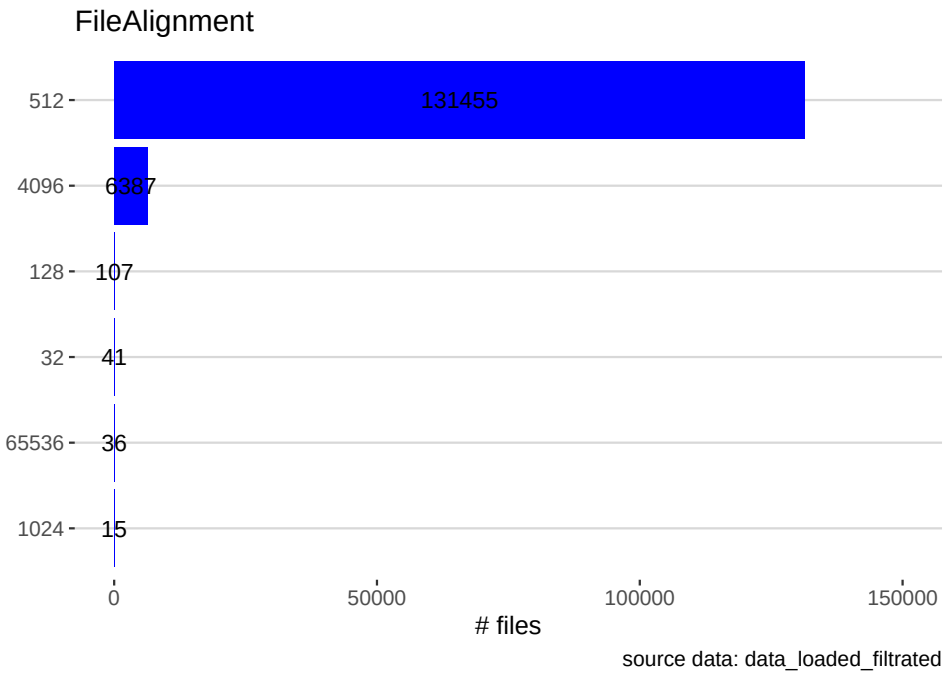

Figure 25: FileAlignment

The **FileAlignment** with the highest amount of files is **512** with a total of **131,455** (**95.22%** of the records on the dataset) files.

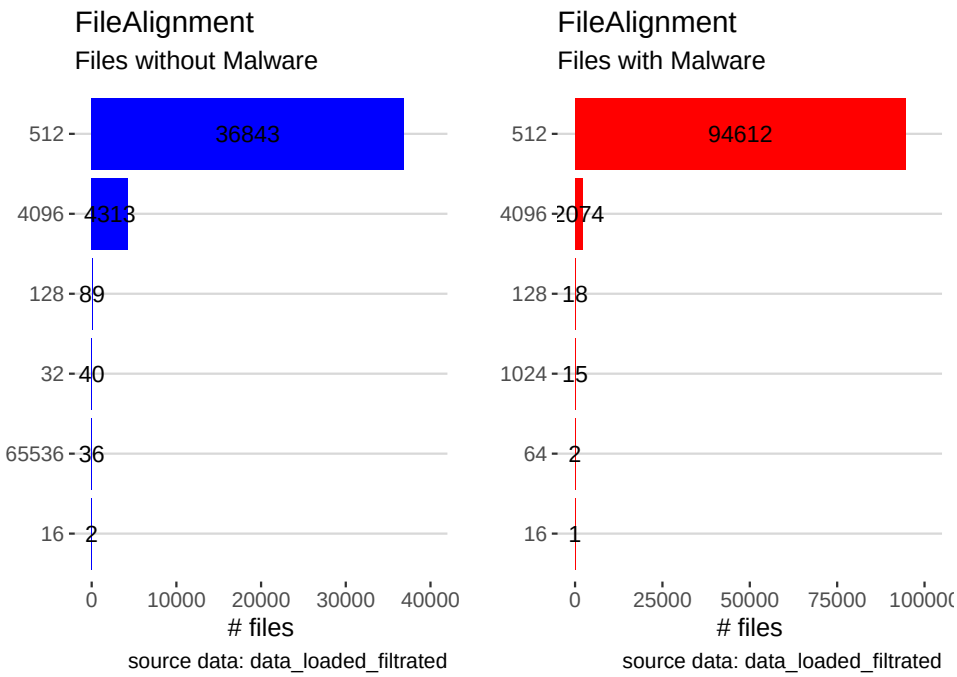

Figure 26: FileAlignment - Files with and without Malware

For files **without malware** the **FileAlignment** with the highest amount is **512** with a total of **36,843** (**26.69%** of the records on the dataset) files.

While files **with malware** the **FileAlignment** with the highest amount is **512** with a total of **94,612** (**68.54%** of the records on the dataset) files.

2.5.11 About ImportsNb

Based on the **ImportsNb** contained in the dataset the distribution is:

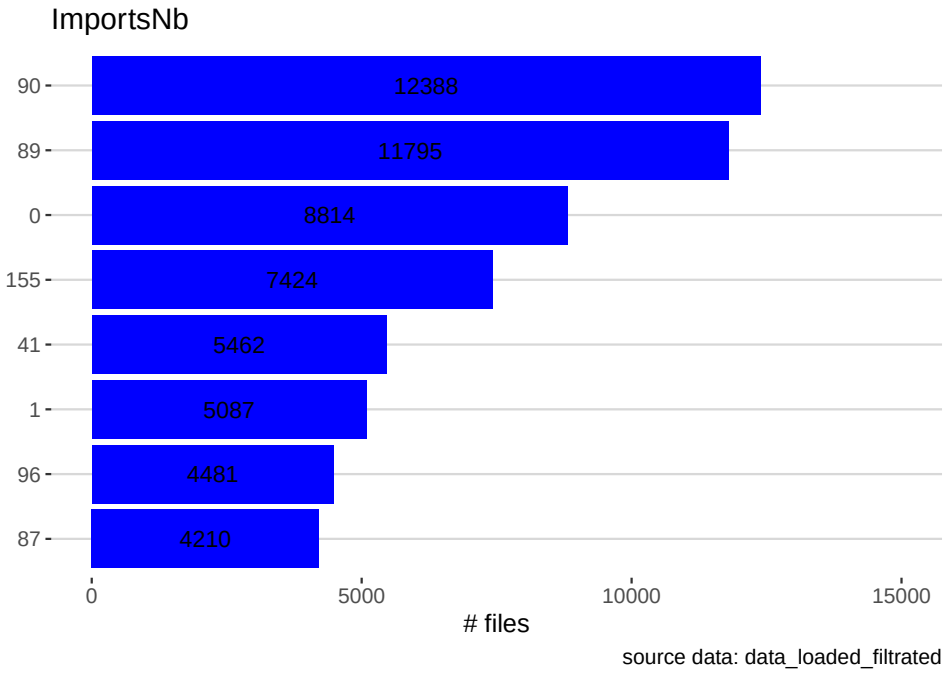

Figure 27: ImportsNb

The **ImportsNb** with the highest amount of files is **90** with a total of **12,388** (**8.97%** of the records on the dataset) files.

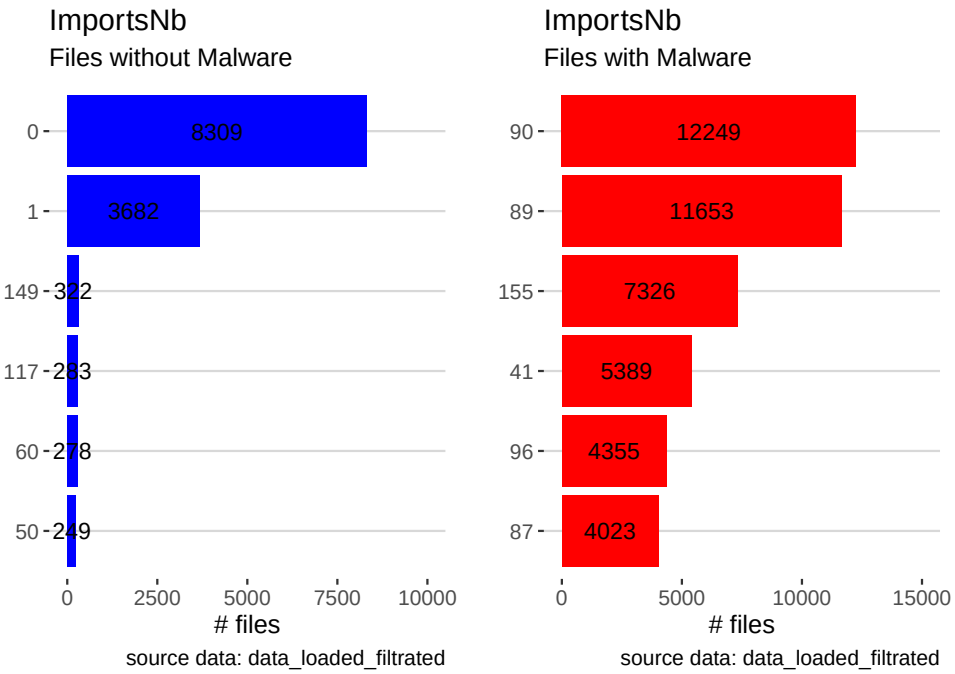

Figure 28: ImportsNb - Files with and without Malware

For files **without malware** the **ImportsNb** with the highest amount is **0** with a total of **8,309** (**6.02%** of the records on the dataset) files.

While files **with malware** the **ImportsNb** with the highest amount is **90** with a total of **12,249** (**8.87%** of the records on the dataset) files.

2.6 About train and test datasets

data_filtrated dataset is splitted on another 2 datasets:

- **train_set** contains **124,242** observations that will be used with every algorithm.
- **test_set** contains **13,805** observations that will be used at the moment to obtain the respective **confusion matrix**.

3 Results

3.1 About confusion matrix

For a **classification use case**, a **confusion matrix** or **error matrix** is used for summarizing the performance of a classification algorithm.

Correct and **incorrect** predictions are summarized in a table with their values and broken down by each class as is displayed in the next image.

The explanation of the key terms related to this performance tool are:

- **True Positive (TP)**: number of malicious PE files that classified as a malicious (malware).
- **True Negative(TN)**: number of benign PE files that classified as a benign (not malware).
- **False Positive (FP)**: number of benign PE files that classified as a malicious (malware).
- **False Negative (FN)**: number of malicious PE files that classified as a benign (not malware).
- **True Positive Rate (TPR) or Sensitivity**: ratio of malicious PE files that classified as a malicious to all the malicious (malware) files.
- **False Positive Rate (FPR)**: ratio of benign PE files that classified as a malicious to all the benign (not malware) files.
- **Accuracy**: ratio of correctly classified files to all files.
- **Specificity**: the proportion of negatives (not malware files) correctly identified as such (or true negative).

	Actually Positive	Actually Negative
Predicted positive	True positives (TP)	False positives (FP)
Predicted negative	False negatives (FN)	True negatives (TN)

Figure 29: Confusion Matrix

The confusion matrix´s formulas are:

Measure of	Name 1	Name 2	Definition	Probability representation
sensitivity	TPR	Recall	$\frac{TP}{TP+FN}$	$\Pr(\hat{Y} = 1 \mid Y = 1)$
specificity	TNR	1-FPR	$\frac{TN}{TN+FP}$	$\Pr(\hat{Y} = 0 \mid Y = 0)$
specificity	PPV	Precision	$\frac{TP}{TP+FP}$	$\Pr(Y = 1 \mid \hat{Y} = 1)$

Figure 30: Confusion Matrix Formulas

3.2 About ROC

ROC stands for **Receiver Operating Characteristics**, and it is used to evaluate the prediction accuracy of a classifier model. ROC curve is a metric describing the trade-off between the sensitivity (true positive rate, TPR) and specificity (false positive rate, FPR) of a prediction in all probability cutoffs (thresholds). It can be used for binary and multi-class classification accuracy checking.

3.3 Algorithms

3.3.1 First algorithm - Binary Logistic Regression

The **logistic model** (or **logit model**) is used to model the probability of a certain class or event existing such as **pass/fail**, **win/lose**, **alive/dead** or **healthy/sick**.

Logistic regression is an extension of linear regression that assures that the estimate of conditional probability $Pr(Y = 1|X = x)$ is between 0 and 1. This approach makes use of the logistic transformation:

$$g(p) = \log \frac{1}{1-p}$$

With logistic regression, the conditional probability is:

$$g\{Pr(Y = 1|X = x)\} = \beta_0 + \beta_1 x$$

The confusion matrix for this algorithm is:

```
##            Reference
## Prediction    0    1
##          0 9498  658
##          1  179 3470
```

The performance of this algorithm is:

- **True Positive**: 9498
- **True Negative**: 3470
- **False Positive**: 179
- **False Negative**: 658
- **Accuracy**: 0.9393698
- **True Positive Rate - Sensitivity**: 0.9815025
- **False Positive Rate**: 0.1593992
- **Specificity**: 0.8406008
- **AUC**: 0.950809474515211
- **Time to be executed**: 1.72 seconds

The ROC curve of this algorithm is

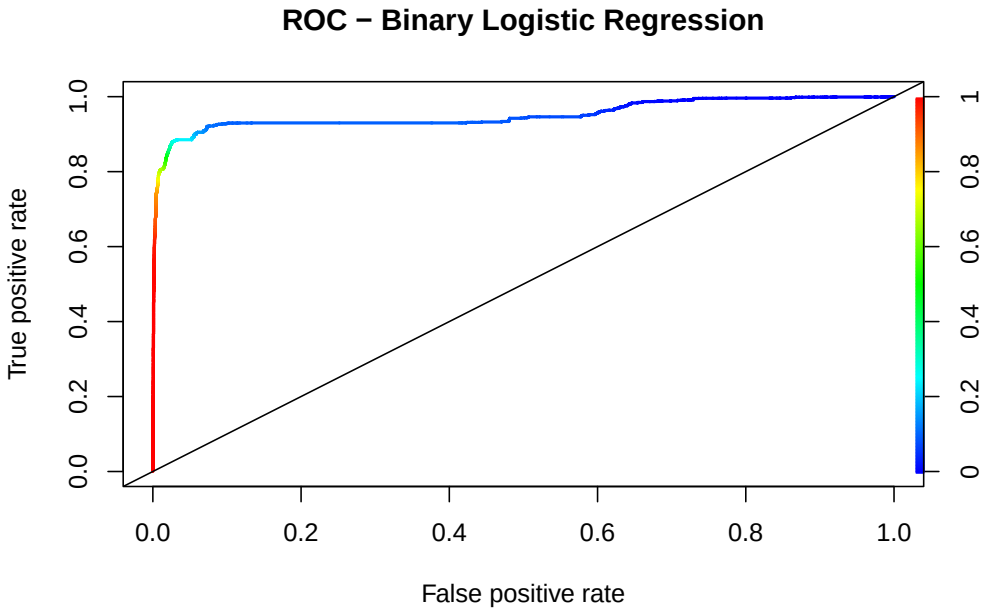

Figure 31: ROC for Binary Logistic Regression

3.3.2 Second algorithm - Naive Bayes

Naïve Bayes is a conditional probability model, which given a problem instance to be classified, represented by a vector $x = (x_1, ..., x_n)$ representing some n features (independent variables), it assigns to this instance probabilities

$$p(C_k|x_1, ..., x_n)$$

for each of **K** possible outcomes or classes C_k. Using **Bayes' theorem**, the conditional probability can be decomposed as:

$$p(C_k|x) = \frac{p(C_k) \cdot p(x|C_x)}{p(x)}$$

Other way to show this equation is:

$$p(x) = Pr(Y = 1|X = x) = \frac{f_{X|Y=1} \cdot (X) \cdot Pr(Y = 1)}{f_{X|Y=0} \cdot Pr(Y = 0) + f_{X|Y=1} \cdot Pr(Y = 1)}$$

The confusion matrix for this algorithm is:

```
##            Reference
## Prediction    0    1
##          0 9448  789
##          1  229 3339
```

The performance of this algorithm is:

- **True Positive**: 9448
- **True Negative**: 3339
- **False Positive**: 229
- **False Negative**: 789
- **Accuracy**: 0.9262586
- **True Positive Rate - Sensitivity**: 0.9763356
- **False Positive Rate**: 0.1911337

- **Specificity**: 0.8088663
- **AUC**: 0.892601
- **Time to be executed**: 3.11 seconds

The ROC curve of this algorithm is

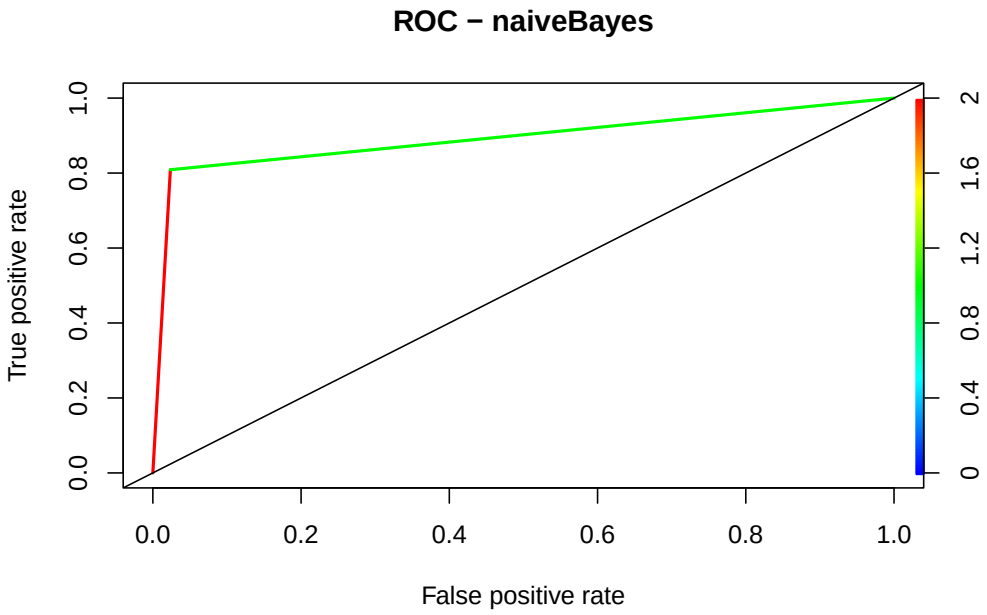

Figure 32: ROC for naiveBayes

3.3.3 Third algorithm - Decision Tree

A decision tree is a simple representation for classifying examples, its goal is to create a model that predicts the value of a target variable based on several input variables.

A tree is built by splitting the source set, constituting the root node of the tree, into subsets which constitute the successor children. The splitting is based on a set of splitting rules based on classification features. This process is repeated on each derived subset in a recursive manner called **recursive partitioning**. The recursion is completed when the subset at a node has all the same values of the target variable, or when splitting no longer adds value to the predictions.

For our case, this is tree created

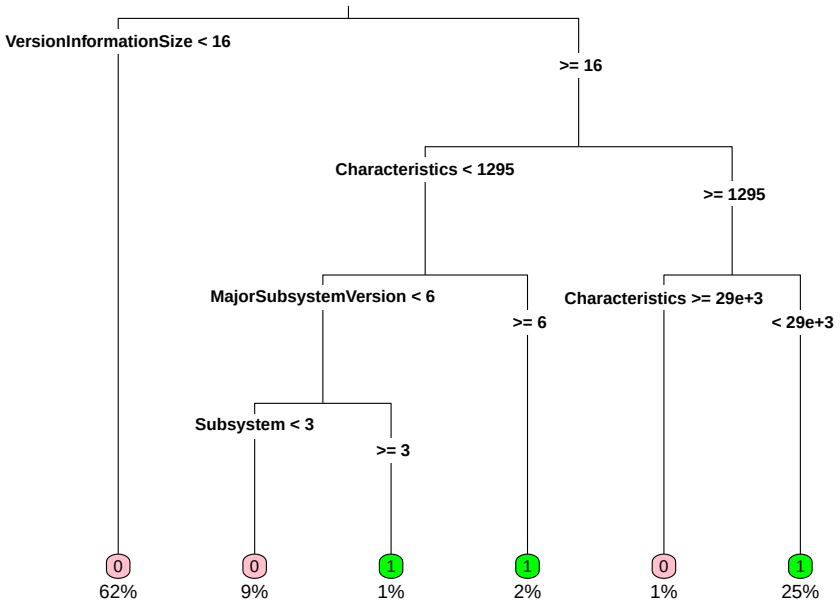

Figure 33: Decision Tree

The confusion matrix for this algorithm is:

```
##            Reference
## Prediction    0    1
##          0 9622  288
##          1   55 3840
```

The performance of this algorithm is:

- **True Positive**: 9622
- **True Negative**: 3840
- **False Positive**: 55
- **False Negative**: 288
- **Accuracy**: 0.9751539
- **True Positive Rate - Sensitivity**: 0.9943164
- **False Positive Rate**: 0.0697674
- **Specificity**: 0.9302326
- **AUC**: 0.9622745
- **Time to be executed**: 1.7 seconds

The ROC curve of this algorithm is

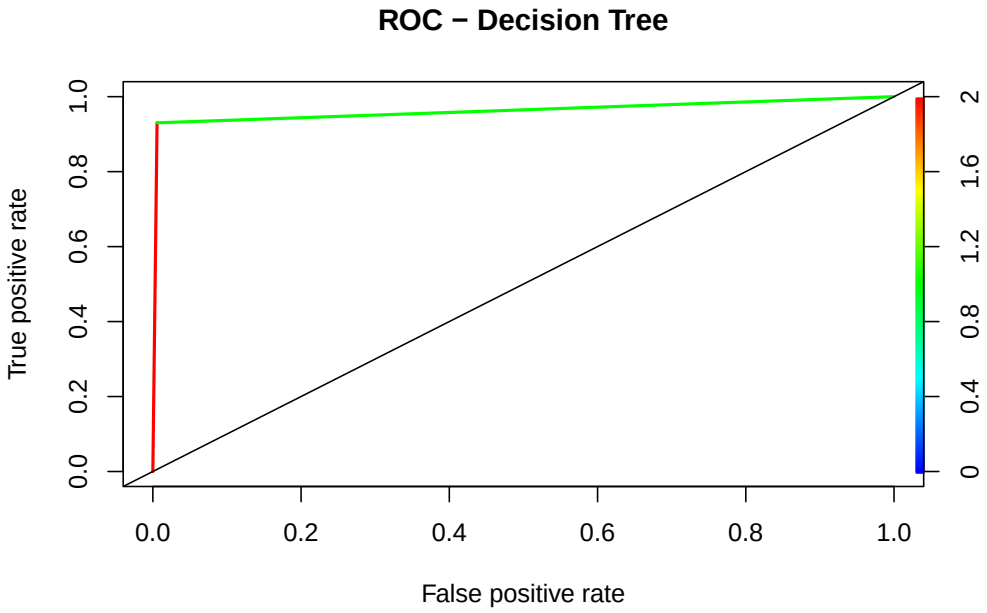

Figure 34: ROC for Decision Tree

3.3.4 Fourth algorithm - Random Forest

Random forest or **random decision forest** is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees.

Bagging is an abbreviation for **Bootstrapping Aggregating**. It is a technique that improves the stability and accuracy of regression and classification algorithms. The biggest advantages and reasons to use bagging are that it **reduces variance** and helps in **avoiding overfitting**. Bagging is most commonly used for decision tree algorithms.

Given a training set $X = x_1, x_2, x_3, ..., x_n$ with responses $Y = y_1, y_2, y_3, ..., y_n$ random samples are taken from the training set multiple times and fit trees to them. Let $f(x_i, y_i)$ be the decision tree for the variables x_i and y_i.

To predict the results for x^t the result can be averaged of all the trees f_i corresponding to x^t (in case of continous).

$$\hat{f} = \frac{1}{N} \cdot \sum_{i=1}^{N} f_i(x\frown)$$

In the case of categorical output, the majority output can be chosen. An estimation of the uncertainty of the prediction can be calculated as:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(f_i(x\frown) - \hat{f})}{N-1}}$$

```
## Total: 105.14 sec elapsed
```

The confusion matrix for this algorithm is:

```
##            Reference
## Prediction    0    1
##          0 9593  105
##          1   84 4023
```

The performance of this algorithm is:

- **True Positive**: 9593
- **True Negative**: 4023
- **False Positive**: 84
- **False Negative**: 105
- **Accuracy**: 0.9863093
- **True Positive Rate - Sensitivity**: 0.9913196
- **False Positive Rate**: 0.025436
- **Specificity**: 0.974564
- **AUC**: 0.9829418
- **Time to be executed**: 105.14 seconds

The ROC curve of this algorithm is

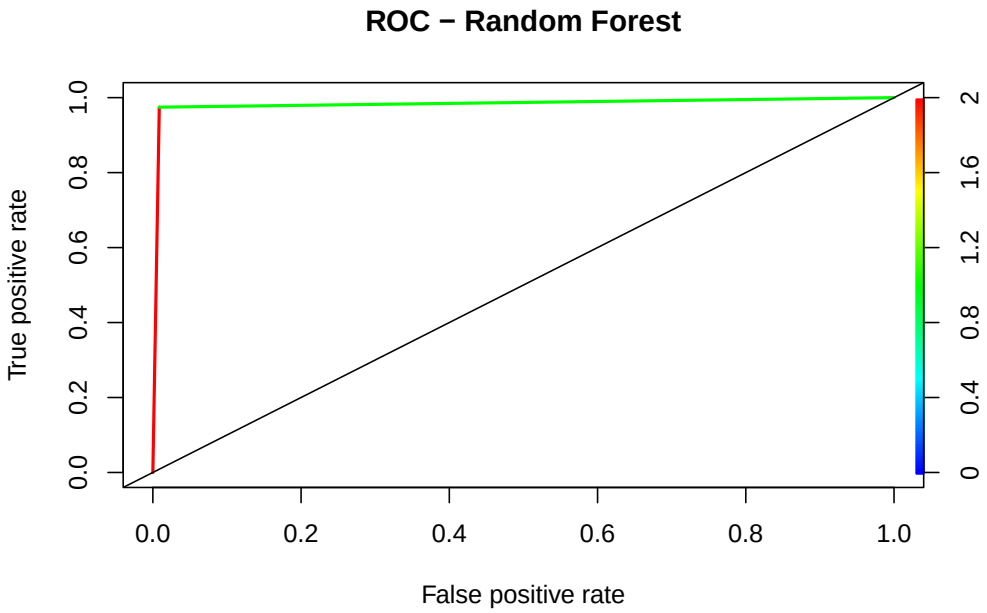

Figure 35: ROC for Random Forest

,

3.3.5 Resume

In the next table are indicated the algorithms´ performance metrics being **Random Forest** the one with the **best performance** followed by **Decision Tree**.

In the othe side, the algorithm with the **worst performance** is **Naive Bayes**.

The algorithm that takes more time to be executed is the one with the majority of the best performance metrics: **Random Forest**.

Table 4: Results obtained

Algorithm	TP	TN	FP	FN	Accuracy	Sensitivity	FPR	Specificity	AUC	Time
#1 - Binary Logistic Regression	9498	3470	179	638	0.9394	0.9815	0.1594	0.8406	0.9508	1.72
#2 - Naive Bayes	9448	3339	229	789	0.9263	0.9763	0.1911	0.8089	0.8926	3.11
#3 - Decision Tree	9622	3840	55	288	0.9752	0.9943	0.0698	0.9302	0.9623	1.70
#4 - Random Forest	9593	4023	84	105	0.9863	0.9913	0.0254	0.9746	0.9829	105.14

,

4 Conclusion

The algorithm that classify the files with the highest **accurary** (**0.9863**) is **Random Forest**, this algorithm is the one that takes more time to run: **105.14** seconds. The second best algorithm is **Decision Tree** (with an accuracy of **0.9752**) that takes **1.7** seconds to be executed.

About **sensitivity**, **Decision Tree** has the highest ratio (**0.9943**) with **9,622** files identified as **malware**, there is a difference with respect to **Random Forest** with a ratio of **0.9913** and **9,593** files identified as **malware**, which is good because with this techniques, the technological infrastructure can be protected against a cyberattack.

For **specificity**, **Random Forest** has the highest ratio (**0.9746**) with **4,023** files identified as **not malware**, there is a difference with respect to **Decision Tree** with a ratio of **0.9302** and **3,840** files identified as **not malware**.

An issue can be presented when a file **without malware** is classified as **malware**, in this case **Random Forest** is the best option with a ratio of **0.0254** with **84** files in this situation.

As an improvement **kNN algorithm** can be used, for this project an end-user device with **8 GiB RAM, 2 virtual cores, 2.90 GHz Intel processor speed running Windows 10 was used** and at the moment to run the following error message was generated: **"Error: cannot allocate vector of size 12.8 Gb"**.

Also techniques more advanced such as **neural networks** and **deep learning** can be explored as a way to increase the accuracy of the classification.

5 Appendix A - PE file fields structure

Table 5: PE file structure fields description

Column name	Type	Description
Name	character	Name of the application running. Some values obtained from the dataset are: memtest.exe, WMVSDECD.DLL, VirusShare_d7648eae45f09b3adb75127f43be6d11, VirusShare_d6abc9f5dbcd3812d48c82056520a866.
md5	character	Unique MD5 (Message-Digest algorithm 5) hash generated to every process running. Some values obtained from the dataset are: 631ea355665f28d4707448e442fbf5b8, 4440449d9212e4955d88e50bd46eaf4f, d7648eae45f09b3adb75127f43be6d11, d6abc9f5dbcd3812d48c82056520a866.
Machine	integer	The number that identifies the type of target machine. Some values obtained from the dataset are: 332, 512, 34404.
SizeOfOptionalHeader	integer	The size of the optional header, which is required for executable files but not for object files. This value should be zero for an object file. Some values obtained from the dataset are: 224, 240, 248, 352.
Characteristics	integer	The flags that indicate the attributes of the file. Some values obtained from the dataset are: 2, 6, 34, 258.
MajorLinkerVersion	integer	The linker major version number. Some values obtained from the dataset are: 0, 10, 83, 255.
MinorLinkerVersion	integer	The linker minor version number. Some values obtained from the dataset are: 0, 12, 82, 160.
SizeOfCode	integer	The size of the code (text) section, or the sum of all code sections if there are multiple sections. Some values obtained from the dataset are: 0, 7680, 102400, 2654208.
SizeOfInitializedData	numeric	The size of the initialized data section, or the sum of all such sections if there are multiple data sections. Some values obtained from the dataset are: 2048, 66560, 385024, 3014656.
SizeOfUninitializedData	numeric	The size of the uninitialized data section (BSS), or the sum of all such sections if there are multiple BSS sections. Some values obtained from the dataset are: 0, 1024, 348160, 23060480.
AddressOfEntryPoint	integer	The address of the entry point relative to the image base when the executable file is loaded into memory. For program images, this is the starting address. For device drivers, this is the address of the initialization function. An entry point is optional for DLLs. When no entry point is present, this field must be zero. Some values obtained from the dataset are: 8191, 61532, 260520, 76849240.
BaseOfCode	integer	The address that is relative to the image base of the beginning-of-code section when it is loaded into memory. Some values obtained from the dataset are: 4096, 36864, 417792, 5099520.
BaseOfData	integer	The address that is relative to the image base of the beginning-of-data section when it is loaded into memory. Some values obtained from the dataset are: 16384, 81920, 200704, 17649664.
ImageBase	numeric	The preferred address of the first byte of image when loaded into memory; must be a multiple of 64 K. The default for DLLs is 0x10000000. The default for Windows CE EXEs is 0x00010000. The default for Windows NT, Windows 2000, Windows XP, Windows 95, Windows 98, and Windows Me is 0x00400000. Some values obtained from the dataset are: 65536, 4194304, 1317601280, 8793776193536.

Table 5: PE file structure fields description *(continued)*

Column name	Type	Description
SectionAlignment	integer	The alignment (in bytes) of sections when they are loaded into memory. It must be greater than or equal to FileAlignment. The default is the page size for the architecture. Some values obtained from the dataset are: 32, 4096, 8192, 65536.
FileAlignment	integer	The alignment factor (in bytes) that is used to align the raw data of sections in the image file. The value should be a power of 2 between 512 and 64 K, inclusive. The default is 512. If the SectionAlignment is less than the architecture's page size, then FileAlignment must match SectionAlignment. Some values obtained from the dataset are: 16, 32, 512, 4096.
MajorOperatingSystemVersion	integer	The major version number of the required operating system. Some values obtained from the dataset are: 4, 5, 6, 10.
MinorOperatingSystemVersion	integer	The minor version number of the required operating system. Some values obtained from the dataset are: 0, 1, 3, 11.
MajorImageVersion	integer	The major version number of the image. Some values obtained from the dataset are: 8, 1741, 5645, 21315.
MinorImageVersion	integer	The minor version number of the image. Some values obtained from the dataset are: 0, 3, 31, 20512.
MajorSubsystemVersion	integer	The major version number of the subsystem. Some values obtained from the dataset are: 4, 5, 6, 10.
MinorSubsystemVersion	integer	The minor version number of the subsystem. Some values obtained from the dataset are: 1, 10, 17, 47600.
SizeOfImage	integer	The size (in bytes) of the image, including all headers, as the image is loaded in memory. It must be a multiple of SectionAlignment. Some values obtained from the dataset are: 3200, 335872, 942080, 336371712.
SizeOfHeaders	integer	The combined size of an MS-DOS stub, PE header, and section headers rounded up to a multiple of FileAlignment. Some values obtained from the dataset are: 512, 1024, 4096, 65536.
CheckSum	numeric	The image file checksum. The algorithm for computing the checksum is incorporated into IMAGHELP.DLL. The following are checked for validation at load time: all drivers, any DLL loaded at boot time, and any DLL that is loaded into a critical Windows process. Some values obtained from the dataset are: 0, 30802, 542760, 2018449232.
Subsystem	integer	The subsystem that is required to run this image. For more information, see Windows Subsystem. Some values obtained from the dataset are: 1, 2, 3, 16.
DllCharacteristics	integer	This field has the following descriptions: Image can handle a high entropy 64-bit virtual address space. DLL can be relocated at load time. Code Integrity checks are enforced. Image is NX compatible. Isolation aware, but do not isolate the image. Does not use structured exception (SE) handling. No SE handler may be called in this image. Do not bind the image. Image must execute in an AppContainer. A WDM driver. Image supports Control Flow Guard. Terminal Server aware. Some values obtained from the dataset are: 320, 33088, 34112, 49504.
SizeOfStackReserve	integer	The size of the stack to reserve. Only SizeOfStackCommit is committed; the rest is made available one page at a time until the reserve size is reached. Some values obtained from the dataset are: 262144, 1048576, 4194304, 33554432.
SizeOfStackCommit	integer	The size of the stack to commit. Some values obtained from the dataset are: 4096, 8192, 32768, 1048576.

Table 5: PE file structure fields description *(continued)*

Column name	Type	Description
SizeOfHeapReserve	integer	The size of the local heap space to reserve. Only SizeOfHeapCommit is committed; the rest is made available one page at a time until the reserve size is reached. Some values obtained from the dataset are: 0, 1048576, 2097152, 4194304.
SizeOfHeapCommit	integer	The size of the local heap space to commit. Some values obtained from the dataset are: 4096, 8192, 65536, 1196032.
LoaderFlags	numeric	Reserved, must be zero. Some values obtained from the dataset are: 0, 256, 7118, 2328297507.
NumberOfRvaAndSizes	numeric	The number of data-directory entries in the remainder of the optional header. Each describes a location and size. Some values obtained from the dataset are: 16, 33, 23203, 2415919120.
SectionsNb	integer	No information found about this field. Some values obtained from the dataset are: 1, 5, 8, 40.
SectionsMeanEntropy	numeric	Mean entropy of the file sections. Some values obtained from the dataset are: 3.34528001443, 5.5972663736, 6.85104258165, 7.98897384762.
SectionsMinEntropy	numeric	Minimal entropy of the file sections. Some values obtained from the dataset are: 1.41974640977, 4.11726411592, 5.44303766881, 7.98887102751.
SectionsMaxEntropy	numeric	Maximum entropy of the file sections. Some values obtained from the dataset are: 1.12708979428, 4.86426869994, 6.0685263873, 7.999990662.
SectionsMeanRawsize	numeric	Mean raw size of the file sections. Some values obtained from the dataset are: 402.285714286, 45348.5714286, 106496, 578845696.
SectionsMinRawsize	integer	Minimal raw size of the file sections. Some values obtained from the dataset are: 512, 6656, 9728, 17408.
SectionMaxRawsize	numeric	Maximum raw size of the file sections. Some values obtained from the dataset are: 4096, 145920, 387072, 321623040.
SectionsMeanVirtualsize	numeric	No information found about this field. Some values obtained from the dataset are: 3162.2, 54706.75, 102644.2, 3081557.33333.
SectionsMinVirtualsize	integer	No information found about this field. Some values obtained from the dataset are: 52, 3296, 9416, 182696.
SectionMaxVirtualsize	numeric	No information found about this field. Some values obtained from the dataset are: 29184, 177152, 847004, 335565920.
ImportsNbDLL	integer	No information found about this field. Some values obtained from the dataset are: 1, 7, 14, 24.
ImportsNb	integer	No information found about this field. Some values obtained from the dataset are: 7, 92, 209, 1437.
ImportsNbOrdinal	integer	No information found about this field. Some values obtained from the dataset are: 1, 16, 129, 1882.
ExportNb	integer	No information found about this field. Some values obtained from the dataset are: 18, 220, 2118, 12705.
ResourcesNb	integer	No information found about this field. Some values obtained from the dataset are: 2, 14, 547, 7457.
ResourcesMeanEntropy	numeric	No information found about this field. Some values obtained from the dataset are: 2.74841221217, 3.19530147365, 6.34249636714, 7.07383209462.
ResourcesMinEntropy	numeric	No information found about this field. Some values obtained from the dataset are: 1.36509557189, 2.45849222582, 3.17703622612, 6.51709259876.
ResourcesMaxEntropy	numeric	No information found about this field. Some values obtained from the dataset are: 3.48999501523, 5.23143173123, 6.27622622234, 7.99268848362.

Table 5: PE file structure fields description *(continued)*

Column name	Type	Description
ResourcesMeanSize	**numeric**	No information found about this field. Some values obtained from the dataset are: 504, 2738.83333333, 30987.1, 141905.426829.
ResourcesMinSize	**numeric**	No information found about this field. Some values obtained from the dataset are: 16, 48, 381, 1084.
ResourcesMaxSize	**numeric**	No information found about this field. Some values obtained from the dataset are: 908, 9640, 67624, 6237669.
LoadConfigurationSize	**numeric**	No information found about this field. Some values obtained from the dataset are: 72, 92, 148, 55958923.
VersionInformationSize	**integer**	No information found about this field. Some values obtained from the dataset are: 15, 18, 23, 26.
legitimate	**integer**	If a "0" appears indicate that the file is a malware and if a "1" appears indicate that the file is not a malware.

,

6 References

- Introduction to Data Science. Rafael A. Irizarry. https://rafalab.github.io/dsbook/
- Create Awesome LaTeX Table with knitr::kable and kableExtra. Hao Zhu. https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_pdf.pdf
- R Markdown Cookbook. https://bookdown.org/yihui/rmarkdown-cookbook/
- What is a malware?. https://www.cisco.com/c/en/us/products/security/advanced-malware-protection/what-is-malware.html
- 2022 Cyber Attack Statistics, Data, and Trends. https://parachute.cloud/2022-cyber-attack-statistics-data-and-trends/
- Malware statistics and facts for 2022. https://www.comparitech.com/antivirus/malware-statistics-facts/
- 2021 Cyber Security Statistics The Ultimate List Of Stats, Data & Trends. https://purplesec.us/resources/cyber-security-statistics/
- Cybercrime To Cost The World $10.5 Trillion Annually By 2025. https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/
- How to use Kaggle API to download datasets in R. https://medium.com/mcd-unison/how-to-use-kaggle-api-to-download-datasets-in-r-312179c7a99c
- A Comprehensive Guide To PE Structure, The Layman's Way. https://tech-zealots.com/malware-analysis/pe-portable-executable-structure-malware-analysis-part-2/
- PE Format. https://docs.microsoft.com/en-us/windows/win32/debug/pe-format
- Malware researcher's handbook (demystifying PE file). https://resources.infosecinstitute.com/topic/2-malware-researchers-handbook-demystifying-pe-file/
- Detailed analysis of PE file format. https://programmer.ink/think/detailed-analysis-of-pe-file-format.html
- File Entropy – Let's talk about Randomness (Malware Analysis – Chapter 2). https://www.talentcookie.com/2016/02/05/file-entropy-in-malware-analysis/
- Decoding the Confusion Matrix. https://towardsdatascience.com/decoding-the-confusion-matrix-bb4801decbb
- Logistic regression. https://en.wikipedia.org/wiki/Logistic_regression
- Naive Bayes classifier. https://en.wikipedia.org/wiki/Naive_Bayes_classifier
- Decision tree learning. https://en.wikipedia.org/wiki/Decision_tree_learning
- Random Forest. https://en.wikipedia.org/wiki/Random_forest
- R Random Forest – Ensemble Learning Methods in R. https://techvidvan.com/tutorials/r-random-forest/