PROJECT REPORT I : CIC-Evasive-PDFMal2022

Authors: Group 5

Himanshu Shekhar (220454)
Lokesh Yadav (220594)
Kamal Kant Tripathi (241110086)

October 7, 2024

1 Introduction

CIC-Evasive-PDFMal2022 is a notable variant of malware that specifically targets PDF documents to evade detection by security systems. This type of malware typically utilizes sophisticated techniques to conceal its malicious payload, such as embedding harmful scripts within seemingly benign PDF files. Its evasive capabilities pose significant challenges for cybersecurity professionals, as traditional detection methods often struggle to identify these hidden threats.

To improve upon existing defenses against CIC-Evasive-PDFMal2022, several strategies could be implemented. First, enhancing machine learning algorithms to analyze PDF file structures and behaviors can lead to more accurate identification of anomalies associated with malicious content. Continuous training of these models with the latest threat data will help in recognizing emerging variants.

In this project, we are working on implementing the same concept. The succeeding paragraphs describe the work done till date as provided via deliverables in project synopsis.

The **GitHub repository** can be found **here**

2 Dataset Analysis

We have downloaded the dataset from CIC Website which contains the folders with the names 'benign' and 'malicious' .The folders contain the benign and malicious pdf files respectively.

2.1 Data pre-processing

2.1.1 PDFMalware2022.CSV

The relevant features which we are going to utilize in this project have been extracted from the pdf files and a PDFMalware2022.CSV file has been created.

The CSV file contained several missing values that needed to be addressed to maintain the integrity of the data. To handle these gaps, we implemented imputation techniques, which involve replacing missing values with estimated ones. For each instance of missing data, we applied both row-wise and column-wise imputation methods. Row-wise imputation focuses on filling in missing values based on information from other entries in the same row, while column-wise imputation uses data from the same column to provide estimates. By employing both techniques, we aimed to ensure that the replacements were as accurate and contextually relevant as possible, ultimately enhancing the overall quality and usability of the dataset. Refer the screenshots below:

```
RangeIndex: 10026 entries, 0 to 10025
Data columns (total 33 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Fine name          10026 non-null  object
 1   pdfsize            10025 non-null  float64
 2   metadata size      10025 non-null  float64
 3   pages              10025 non-null  float64
 4   xref Length        10025 non-null  float64
 5   title characters   10025 non-null  float64
 6   isEncrypted        10025 non-null  float64
 7   embedded files     10025 non-null  float64
 8   images             10025 non-null  object
 9   text               10025 non-null  object
 10  header             10025 non-null  object
 11  obj                10023 non-null  object
 12  endobj             10023 non-null  object
 13  stream             10023 non-null  float64
 14  endstream          10023 non-null  object
 15  xref               10023 non-null  object
 16  trailer            10023 non-null  float64
 17  startxref          10023 non-null  object
 18  pageno             10023 non-null  object
 19  encrypt            10023 non-null  float64
 20  ObjStm             10023 non-null  float64
 21  JS                 10023 non-null  object
 22  Javascript         10023 non-null  object
 23  AA                 10023 non-null  object
 24  OpenAction         10023 non-null  object
 25  Acroform           10023 non-null  object
 26  JBIG2Decode        10023 non-null  object
 27  RichMedia          10023 non-null  object
 28  launch             10023 non-null  object
 29  EmbeddedFile       10023 non-null  object
 30  XFA                10023 non-null  object
 31  Colors             10023 non-null  float64
 32  Class              10025 non-null  object
dtypes: float64(12), object(21)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10026 entries, 0 to 10025
Data columns (total 32 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   pdfsize           10026 non-null  float64
 1   metadata size     10026 non-null  float64
 2   pages             10026 non-null  float64
 3   xref Length       10026 non-null  float64
 4   title characters  10026 non-null  float64
 5   isEncrypted       10026 non-null  float64
 6   embedded files    10026 non-null  float64
 7   images            10026 non-null  object
 8   text              10026 non-null  object
 9   header            10026 non-null  object
 10  obj               10026 non-null  object
 11  endobj            10026 non-null  object
 12  stream            10026 non-null  float64
 13  endstream         10026 non-null  object
 14  xref              10026 non-null  object
 15  trailer           10026 non-null  float64
 16  startxref         10026 non-null  object
 17  pageno            10026 non-null  object
 18  encrypt           10026 non-null  float64
 19  ObjStm            10026 non-null  float64
 20  JS                10026 non-null  object
 21  Javascript        10026 non-null  object
 22  AA                10026 non-null  object
 23  OpenAction        10026 non-null  object
 24  Acroform          10026 non-null  object
 25  JBIG2Decode       10026 non-null  object
 26  RichMedia         10026 non-null  object
 27  launch            10026 non-null  object
 28  EmbeddedFile      10026 non-null  object
 29  XFA               10026 non-null  object
 30  Colors            10026 non-null  float64
 31  Class             10026 non-null  object
```

It was found that the CSV file contained erroneous values, including entries like 1(1) and -1, which could lead to inconsistencies in data analysis. To standardize the dataset and eliminate potential confusion, we replaced these erroneous values with 0. This approach ensures that the data remains coherent and facilitates more accurate analyses.

Furthermore, we employed LabelEncoder to convert the header values into encoded labels. This transformation is crucial for preparing the dataset for further processing, as it allows for categorical variables to be represented numerically. By encoding these labels, we can enhance the compatibility of the data with various machine learning algorithms, ultimately streamlining the analysis and model-building process.

```
data.head() # Ensure data is loaded correctly
```

[27]:

	pdfsize	metadata size	pages	xref Length	title characters	isEncrypted	embedded files	images	text	header	...	AA	OpenAction	Acroform	JB
0	8.0	180.0	1.0	11.0	0.0	0.0	0.0	0	0	10.0	...	0.0	1.0	0.0	
1	15.0	224.0	0.0	20.0	7.0	0.0	0.0	0	0	21.0	...	0.0	0.0	1.0	
2	4.0	468.0	2.0	13.0	16.0	0.0	0.0	0	1	10.0	...	0.0	1.0	0.0	
3	17.0	250.0	1.0	15.0	0.0	0.0	0.0	0	0	10.0	...	0.0	1.0	1.0	
4	7.0	252.0	3.0	16.0	45.0	0.0	0.0	0	1	10.0	...	0.0	1.0	0.0	

The data was converted to numeric format using the ‘pd.to_numeric‘ function from the pandas library. During this process, any values that could not be converted were automatically set to NaN, allowing for the identification of non-numeric entries. To maintain a clean dataset suitable for analysis, all resulting NaN values were subsequently replaced with 0. This step ensures that the dataset remains consistent and avoids complications during analysis, thereby enhancing the reliability of any insights drawn from the data.

```
Data columns (total 32 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   pdfsize           10026 non-null  float64
 1   metadata size     10026 non-null  float64
 2   pages             10026 non-null  float64
 3   xref Length       10026 non-null  float64
 4   title characters  10026 non-null  float64
 5   isEncrypted       10026 non-null  float64
 6   embedded files    10026 non-null  float64
 7   images            10026 non-null  int64
 8   text              10026 non-null  int64
 9   header            10026 non-null  float64
 10  obj               10026 non-null  float64
 11  endobj            10026 non-null  float64
 12  stream            10026 non-null  float64
 13  endstream         10026 non-null  float64
 14  xref              10026 non-null  float64
 15  trailer           10026 non-null  float64
 16  startxref         10026 non-null  float64
 17  pageno            10026 non-null  float64
 18  encrypt           10026 non-null  float64
 19  ObjStm            10026 non-null  float64
 20  JS                10026 non-null  float64
 21  Javascript        10026 non-null  float64
 22  AA                10026 non-null  float64
 23  OpenAction        10026 non-null  float64
 24  Acroform          10026 non-null  float64
 25  JBIG2Decode       10026 non-null  float64
 26  RichMedia         10026 non-null  float64
 27  launch            10026 non-null  float64
 28  EmbeddedFile      10026 non-null  float64
 29  XFA               10026 non-null  float64
 30  Colors            10026 non-null  float64
 31  Class             10026 non-null  int64
dtypes: float64(29), int64(3)
```

2.1.2 final.CSV

The final.csv file was meticulously cleaned and now contains no null or erroneous values, ensuring the integrity of the dataset. To facilitate effective data analysis and model training, we employed LabelEncoder to convert the header values into properly encoded labels. This transformation is essential for allowing categorical variables to be processed as numerical data, enhancing compatibility with various analytical methods and machine learning algorithms. Overall, the dataset is now in optimal condition for further exploration and analysis.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30828 entries, 0 to 30827
Data columns (total 32 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   pdfsize            30828 non-null  int64
 1   metadata size      30828 non-null  int64
 2   pages              30828 non-null  int64
 3   xref length        30828 non-null  int64
 4   title characters   30828 non-null  int64
 5   isEncrypted        30828 non-null  int64
 6   embedded files     30828 non-null  int64
 7   images             30828 non-null  int64
 8   contains text      30828 non-null  object
 9      header            30828 non-null   object
 10  obj                30828 non-null  int64
 11  endobj             30828 non-null  int64
 12  stream             30828 non-null  int64
 13  endstream          30828 non-null  int64
 14  xref               30828 non-null  int64
 15  trailer            30828 non-null  int64
 16  startxref          30828 non-null  int64
 17  pageno             30828 non-null  int64
 18  Encrypt            30828 non-null  int64
 19  ObjStm             30828 non-null  int64
 20  JS                 30828 non-null  int64
 21  JavaScript         30828 non-null  int64
 22  AA                 30828 non-null  int64
 23  OpenAction         30828 non-null  int64
 24  AcroForm           30828 non-null  int64
 25  JBIG2Decode        30828 non-null  int64
 26  RichMedia          30828 non-null  int64
 27  Launch             30828 non-null  int64
 28  EmbeddedFile       30828 non-null  int64
 29  XFA                30828 non-null  int64
 30  Colors             30828 non-null  int64
 31  Malicious          30828 non-null  object
```

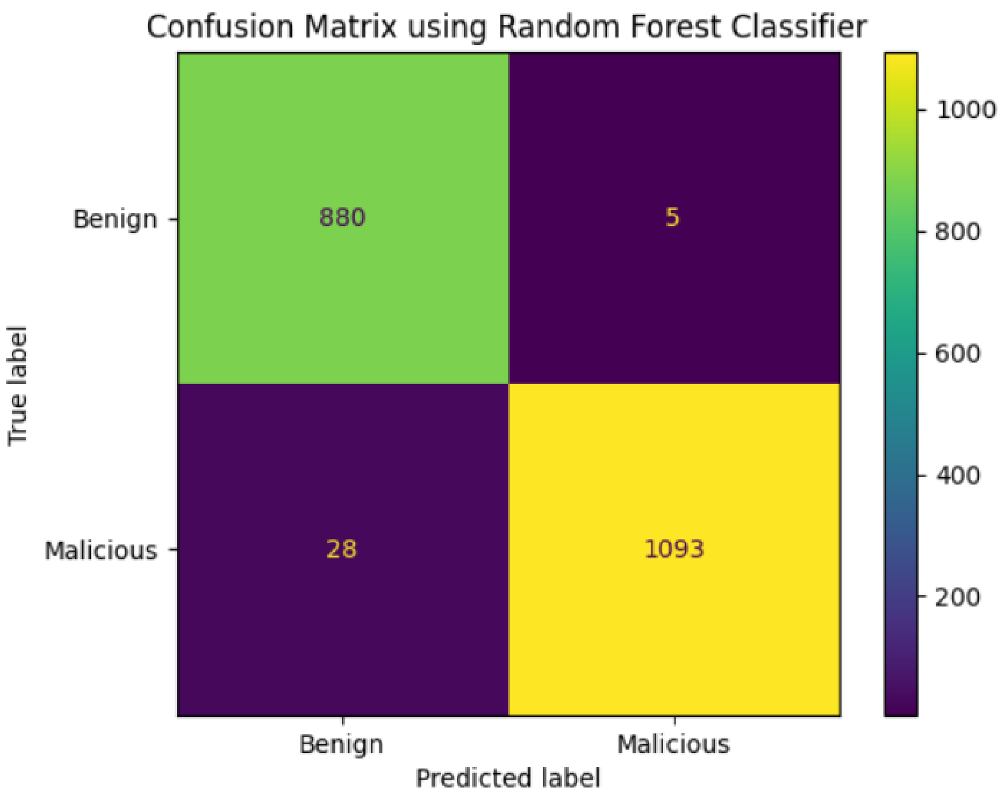
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30828 entries, 0 to 30827
Data columns (total 32 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   pdfsize           30828 non-null  int64
 1   metadata size     30828 non-null  int64
 2   pages             30828 non-null  int64
 3   xref length       30828 non-null  int64
 4   title characters  30828 non-null  int64
 5   isEncrypted       30828 non-null  int64
 6   embedded files    30828 non-null  int64
 7   images            30828 non-null  int64
 8   contains text     30828 non-null  int64
 9      header            30828 non-null  int64
 10  obj               30828 non-null  int64
 11  endobj            30828 non-null  int64
 12  stream            30828 non-null  int64
 13  endstream         30828 non-null  int64
 14  xref              30828 non-null  int64
 15  trailer           30828 non-null  int64
 16  startxref         30828 non-null  int64
 17  pageno            30828 non-null  int64
 18  Encrypt           30828 non-null  int64
 19  ObjStm            30828 non-null  int64
 20  JS                30828 non-null  int64
 21  JavaScript        30828 non-null  int64
 22  AA                30828 non-null  int64
 23  OpenAction        30828 non-null  int64
 24  AcroForm          30828 non-null  int64
 25  JBIG2Decode       30828 non-null  int64
 26  RichMedia         30828 non-null  int64
 27  Launch            30828 non-null  int64
 28  EmbeddedFile      30828 non-null  int64
 29  XFA               30828 non-null  int64
 30  Colors            30828 non-null  int64
 31  Malicious         30828 non-null  int64
```

3 ML Model

3.1 PDFMalware2022.CSV

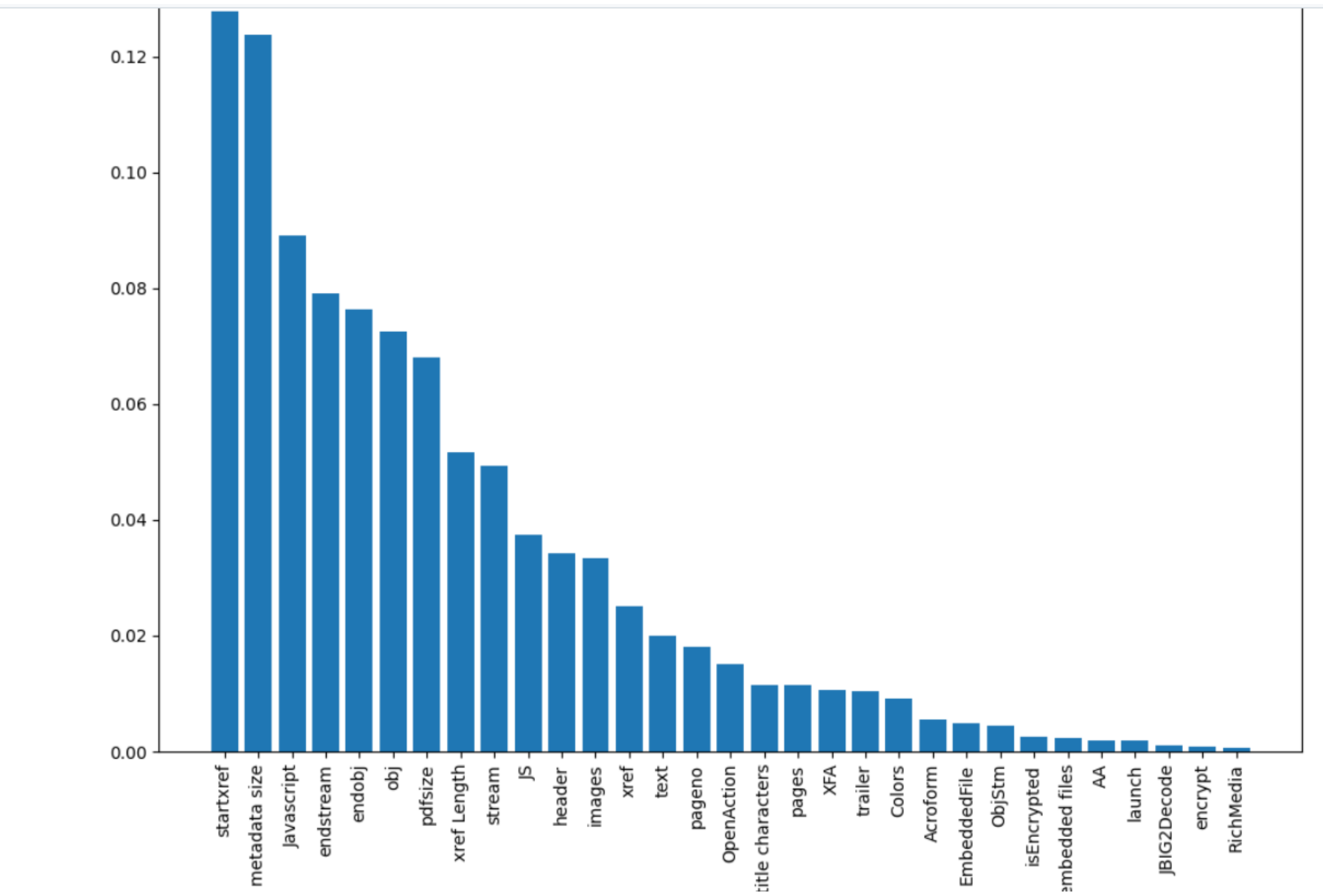
The dataset was split into two parts: 80% was allocated for training the model, while the remaining 20% was reserved as test data. This division is crucial for evaluating the model's performance and ensuring that it generalizes well to unseen data.

After training the model, we applied the Random Forest classifier, which is known for its robustness and ability to handle complex datasets. The model achieved an impressive accuracy of 98.35% on the test set, indicating its effectiveness in correctly classifying the data. This high level of accuracy demonstrates the successful application of the preprocessing techniques we implemented, ensuring that the model was trained on clean, well-structured data.

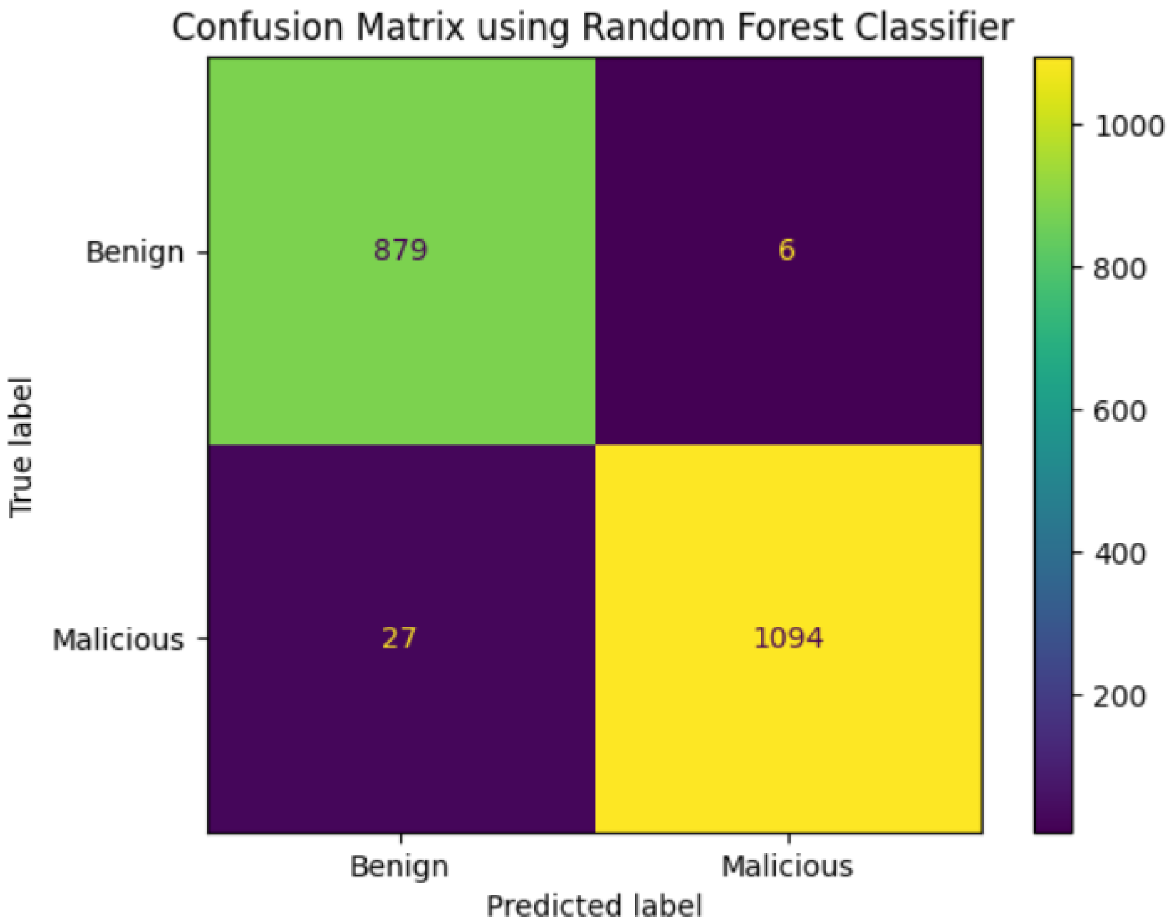



```
False Positive Rate: 0.005649717514124294
True Positive Rate: 0.9750223015165032
Accuracy Score: 0.9835493519441675
```

Refer the screenshot below depicting the feature importance graph



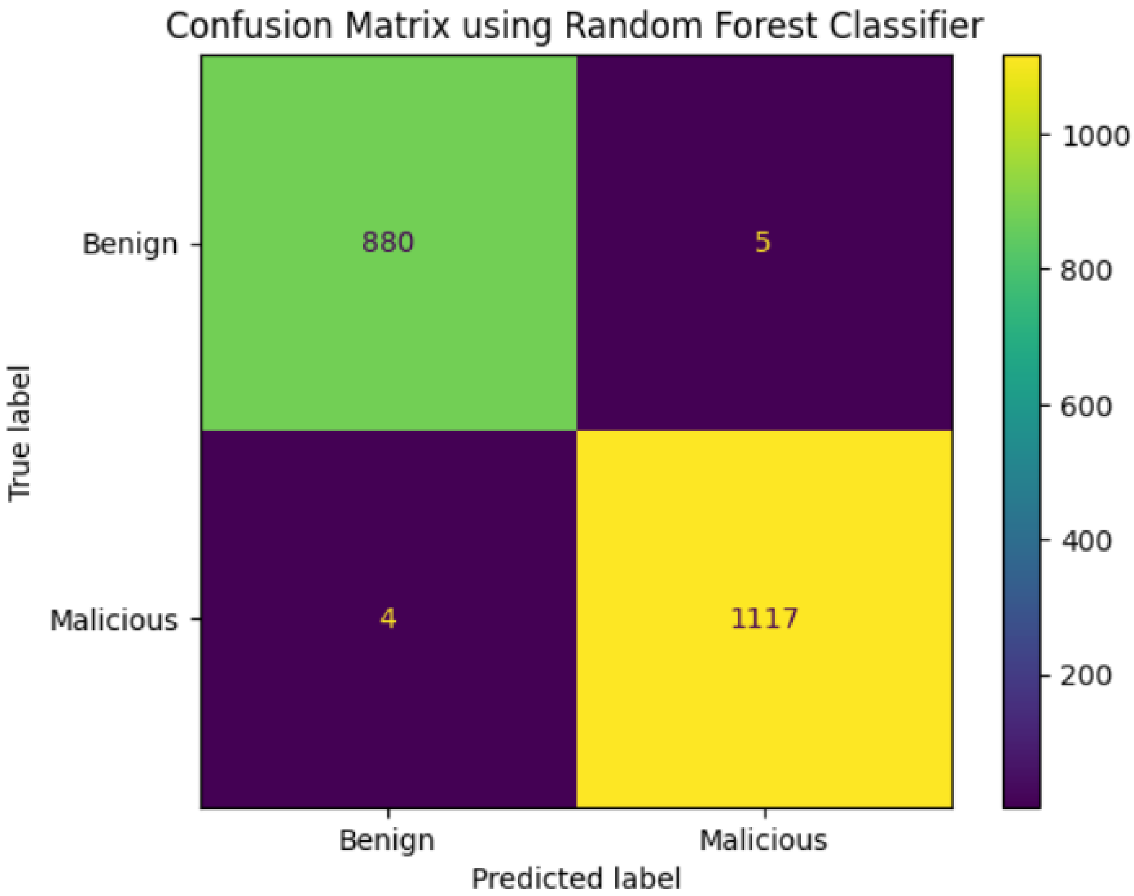
We removed the last seven features from the dataset as part of the feature selection process. After evaluating the model's performance post-removal, we found that the results remained largely unchanged. This outcome suggests that these features had minimal impact on the overall performance of the model. By streamlining the dataset in this way, we not only simplified the model but also reinforced the notion that effective feature selection is key to enhancing interpretability without sacrificing accuracy. This step highlights the importance of identifying and retaining only the most influential features for optimal model performance.




```
False Positive Rate: 0.006779661016949152
True Positive Rate: 0.975914362176628
Accuracy Score: 0.9835493519441675
```

After removing certain limits from the Random Forest classifier, the model achieved an impressive accuracy of 99.55%. However, this substantial increase in accuracy raised concerns about potential overfitting. Overfitting occurs when a model learns not only the underlying patterns in the training data but also the noise, resulting in exceptional performance on the training set but poor generalization to unseen data.

This observation underscores the need for careful model evaluation and validation techniques, such as cross-validation or testing on a separate dataset, to ensure that the model maintains its predictive power across different scenarios.



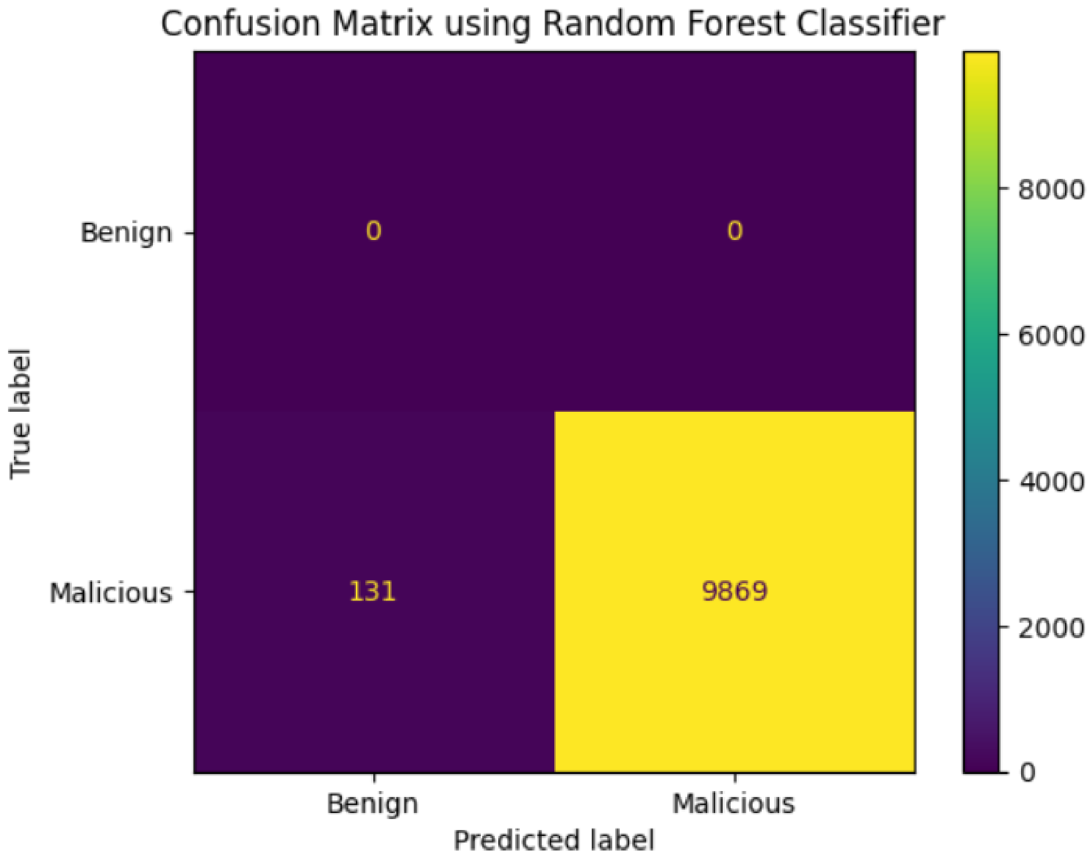

```
False Positive Rate: 0.005649717514124294
True Positive Rate: 0.9964317573595004
Accuracy Score: 0.9955134596211366
```

3.2 final.CSV

For the final evaluation of the model, 10,000 malicious samples were selected from a total of 30,828 samples for testing, while the remainder of the data was utilized for training. The model was trained using a Random Forest Classifier, which is known for its effectiveness in classification tasks.

Upon evaluation, the model achieved an impressive accuracy of 98.68% on the test set. This high accuracy indicates the model's strong capability in accurately identifying malicious samples, reflecting the effectiveness of the training process and the relevance of the selected features. The substantial test sample size enhances the statistical reliability of these results, demonstrating the model's potential for practical application in detecting malicious activity within larger datasets. Overall, this outcome highlights the successful integration of data preprocessing techniques and machine learning methodologies.

```
warnings.warn(
```




```
False Positive Rate: nan
True Positive Rate: 0.9869
Accuracy Score: 0.9869
```

4 PDFMalLyzer

PDFMalLyzer is a tool that extracts 44 different features (general and structural) from a set of pdf files specified by the user and writes them on a csv file. The resulting csv file can be further studied for variety of purposes, most importantly for detecting malicious pdf files.

The original repository (here) that is created by 2 authors of the paper is very outdated and has many bugs in it. Most of the time we spent till now was on fixing this tool so that we could extract features from our PDF Dataset.

Using the original repository we were able to extract around 20000 datapoints (malformed in many columns) combining Benign and Malicious files, as is mentioned in the paper, but this number is due to the fact that there are bugs in the tool. On fixing those bugs, we ended up with around 30000 datapoints. The paper also mentions that there are around 44% duplicate entries in the data but this is also due to bugs in the tool written by them. On fixing this we observed that there were no duplicate entries in the data.

There were values like "/bin/sh: 1: _Cunningham_Studio.pdf: not found", "Error opening file /mnt/hgfs/kali_stuff/CLEAN_PDF_9000_files/Albert_Berger", "[Errno 2] No such file or directory: '/mnt/hgfs/kali_stuff/CLEAN_PDF_9000_files/Albert_Berger", etc. on the csv file which was used for training and testing.

We also observed that the output csv created by the original tool had more data points in a row than there were columns in the dataset.

All these bugs were fixed and now we have a working tool which we have uploaded on our GitHub repository.

5 Things to be fixed

Currently we have implemented only a basic RandomForest model and achieved $98+\%$ accuracy without much of feature engineering. This clearly shows that somewhere our model is overfit and so is the model from the paper. We plan of fixing this unusually high accuracy by proper preprocessing of data and we will also try to add more data to our dataset and try to make the total number of entries to around 50000 files.

We would like to fix our timeline and include some parts of our first deliverable like feature set exploration and model exploration to our 2nd deliverable. Most of our time now was spent in fixing the PDFMalLyzer tool which is now working fine. We believe that by our next deliverable date we will have an extended up-to-date dataset and a model which is not overfit and achieves high accuracy.