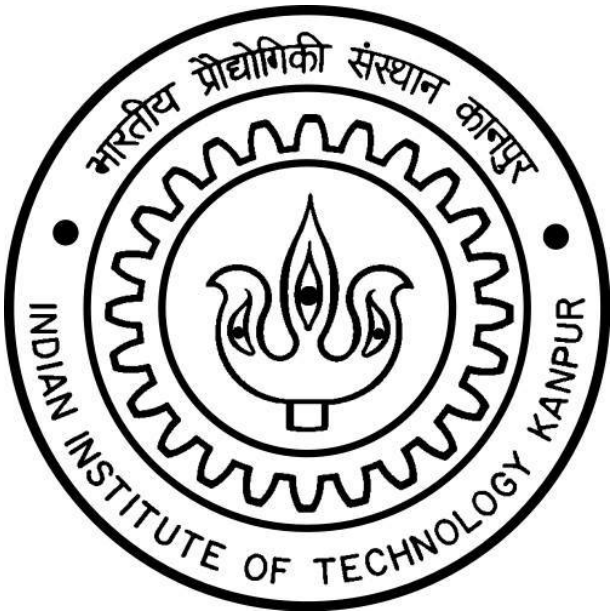


PROJECT REPORT II : CIC-Evasive-PDFMal2022

Authors: Group 5

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

November 8, 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.

In this project, a thorough analysis of the existing dataset and model has been carried out to identify the limitations and finding the innovative ways to overcome the same. The details have been explained in subsequent paragraphs.

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

2 Dataset Analysis

In addition to the existing dataset from CIC Website, we have added a diverse dataset obtained from VirusShare, GOVDocs, PDFRep, VirusTotal etc. The dataset contains equal proportion of malicious and benign PDF files making a balanced and robust set of data to test and validate our improved version of Model. The Dataset analysis is illustrated below:

2.1 Data Collection

In the process of data collection, multiple sources were explored to collect more than 1 lakh benign and approx 60 thousand malicious PDF files whereas the existing dataset were using only 9 and 11 thousand files respectively. **Hence, we have used an enhanced dataset which is a novel idea in this project.** Below is the list of unique malicious and benign files used for feature extraction after dropping the duplicate records:-

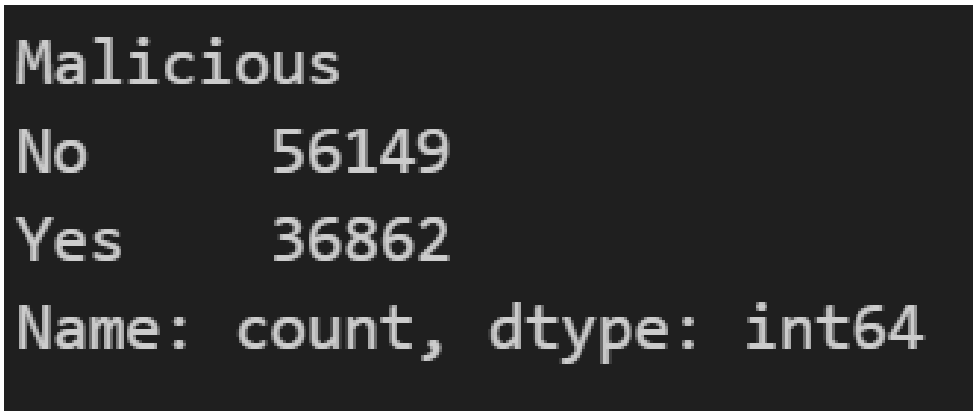


Figure 1

The image above shows the amount of Malicious and Benign data after preprocessing and dropping duplicates.

For training and testing, the distribution used by us are as follows -

```
Malicious
1    30000
0    30000
Name: count, dtype: int64
```

Training data

```
Malicious
0    10000
1     6862
Name: count, dtype: int64
```

Testing data

We tabulated the accuracy results of three models - **RandomForest**, **XGBoost** and **GBM** for the MalwareBazaar dataset and found that using the above balanced data for training works slightly better than using just train-test split.

Best Model for Each Feature Count:

	Model	Features	Accuracy	Precision	Recall	F1 Score
5	GBM	5	73.54%	100.00%	73.54%	84.75%
1	Random Forest	10	74.55%	100.00%	74.55%	85.42%
12	XGBoost	15	80.81%	100.00%	80.81%	89.39%
13	XGBoost	20	82.22%	100.00%	82.22%	90.24%
14	XGBoost	25	83.64%	100.00%	83.64%	91.09%

This table shows the best performing models for the respective number of top features (found using SHAP) when using the above balanced data for training.

Best Model for Each Feature Count:

	Model	Features	Accuracy	Precision	Recall	F1 Score
0	Random Forest	5	54.55%	100.00%	54.55%	70.59%
11	XGBoost	10	73.54%	100.00%	73.54%	84.75%
12	XGBoost	15	78.59%	100.00%	78.59%	88.01%
13	XGBoost	20	82.22%	100.00%	82.22%	90.24%
14	XGBoost	25	80.81%	100.00%	80.81%	89.39%

These are the best performing models when using train-test split for the respective number of top features (found using SHAP).

NOTE: Both these tables shown above list the metrics for the MalwareBazaar dataset without any hyperparameter tuning.

2.2 Data pre-Processing

Feature extraction was carried out from these files using the python script developed by the team. After dropping the duplicate records, a balanced proportion of 30 thousand records each of malicious and benign were selected to incorporate in the final CSV file.

For processing the data, first we checked for **NULL** entries in the data and found out that there were no entries with null value. We started initially with 2Lakh+ rows but after dropping duplicates we were left with the amount as shown in **Figure 1**.

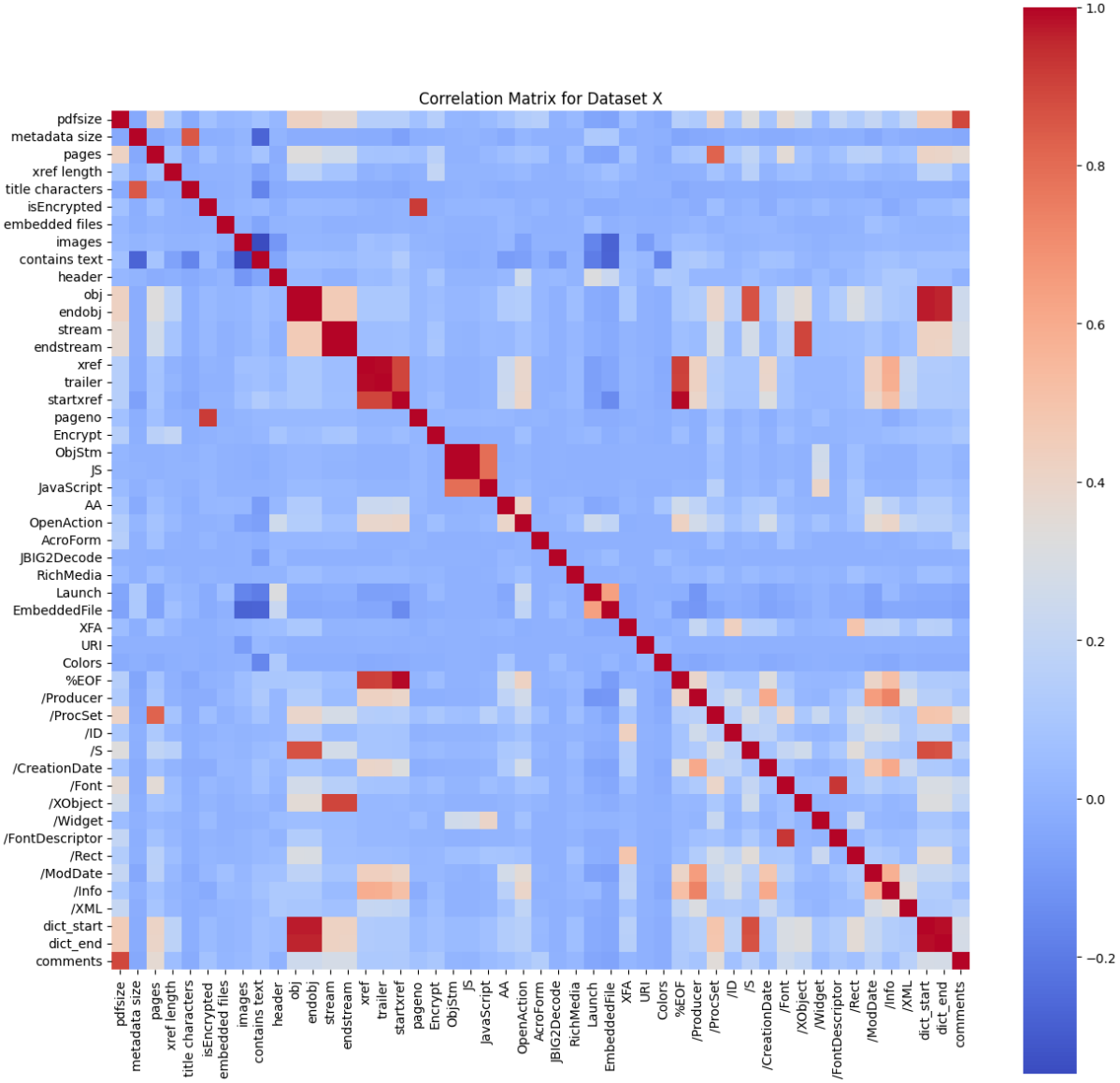
We manually dropped some columns which we deemed to be unnecessary based on

what they were and what they meant (like Obfuscated columns).

And finally we label encoded categorical values like Header etc.

2.3 Feature Selection

The existing work had taken into account only **31** features. However, we build upon that judiciously and explored a total of 61 features and finally selected **25** most important and effective features after due deliberation. **The enhanced feature set has contributed towards a robust model and better accuracy than the existing model and hence is a second novel idea in this project**. The correlation matrix and enhanced feature list are as under:-



```
Data columns (total 62 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   MD5                     210339 non-null  object
 1   pdfsize                 210339 non-null  int64
 2   metadata size           210339 non-null  int64
 3   pages                   210339 non-null  int64
 4   xref length             210339 non-null  int64
 5   title characters        210339 non-null  int64
 6   isEncrypted             210339 non-null  int64
 7   embedded files          210339 non-null  int64
 8   images                  210339 non-null  int64
 9   contains text           210339 non-null  object
 10  header                  210339 non-null  object
 11  obj                     210339 non-null  int64
 12  endobj                  210339 non-null  int64
 13  stream                  210339 non-null  int64
 14  endstream               210339 non-null  int64
 15  xref                    210339 non-null  int64
 16  trailer                 210339 non-null  int64
 17  startxref               210339 non-null  int64
 18  pageno                  210339 non-null  int64
 19  Encrypt                 210339 non-null  int64
 20  ObjStm                  210339 non-null  int64
 21  JS                      210339 non-null  int64
 22  JavaScript              210339 non-null  int64
 23  AA                      210339 non-null  int64
 24  OpenAction              210339 non-null  int64
 25  AcroForm                210339 non-null  int64
 26  JBIG2Decode             210339 non-null  int64
 27  RichMedia               210339 non-null  int64
 28  Launch                  210339 non-null  int64
 29  EmbeddedFile            210339 non-null  int64
 30  XFA                     210339 non-null  int64
```

31	URI	210339	non-null	int64
32	Colors	210339	non-null	int64
33	JS_Obfuscated	210339	non-null	int64
34	JavaScript_Obfuscated	210339	non-null	int64
35	AA_Obfuscated	210339	non-null	int64
36	OpenAction_Obfuscated	210339	non-null	int64
37	AcroForm_Obfuscated	210339	non-null	int64
38	JBIG2Decode_Obfuscated	210339	non-null	int64
39	RichMedia_Obfuscated	210339	non-null	int64
40	Launch_Obfuscated	210339	non-null	int64
41	EmbeddedFile_Obfuscated	210339	non-null	int64
42	XFA_Obfuscated	210339	non-null	int64
43	pageno_Obfuscated	210339	non-null	int64
44	%EOF	210339	non-null	int64
45	/Producer	210339	non-null	int64
46	/ProcSet	210339	non-null	int64
47	/ID	210339	non-null	int64
48	/S	210339	non-null	int64
49	/CreationDate	210339	non-null	int64
50	/Font	210339	non-null	int64
51	/XObject	210339	non-null	int64
52	/Widget	210339	non-null	int64
53	/FontDescriptor	210339	non-null	int64
54	/Rect	210339	non-null	int64
55	/ModDate	210339	non-null	int64
56	/Info	210339	non-null	int64
57	/XML	210339	non-null	int64
58	dict_start	210339	non-null	int64
59	dict_end	210339	non-null	int64
60	comments	210339	non-null	int64
61	Malicious	210339	non-null	object

The final **25** features used by us in our final model are as follows (sorted based on their SHAP scores)-

```
XFA                 2.373532
/ID                 2.139211
xref length         1.015798
endobj              0.861669
/Info               0.840707
/ModDate            0.833893
pdfsize             0.792699
/Rect               0.782789
/FontDescriptor     0.778657
pages               0.732491
/XObject            0.717858
obj                 0.715441
header              0.712542
title characters    0.678666
dict_end            0.644826
stream              0.629956
comments            0.621262
JS                  0.598691
xref                0.575184
/Font               0.572858
endstream           0.569602
contains text       0.549298
/ProcSet            0.489325
metadata size       0.484371
dict_start          0.448918
dtype: float64
```

2.4 Implementing SHAP (SHapley Additive exPlanations)

2.4.1 About SHAP

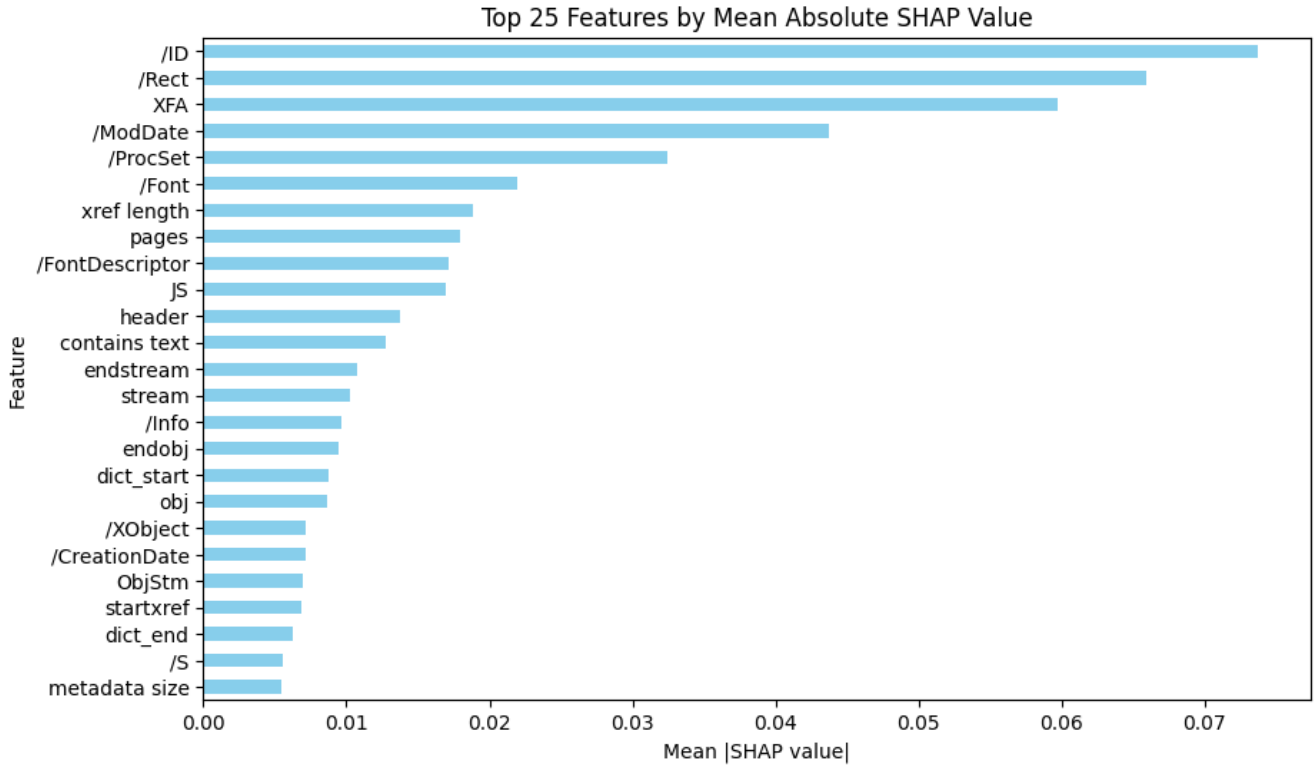
SHAP measures each feature's contribution to the model's output, is the third novel idea in this project and has been incorporated for feature selection process. SHAP (SHapley Additive exPlanations) is a popular method for explaining individual predictions of machine learning models. It's based on Shapley values from cooperative game theory, where the idea is to fairly distribute the "payout" (in this case, the prediction) among features based on their contributions to the final prediction. Shapley values assign a fair value to each feature based on its

contribution to the model's output. They measure each feature's contribution by considering all possible combinations of features, which leads to more robust and reliable explanations. SHAP operates under an additive feature attribution framework, meaning the sum of each feature's contribution equals the model's prediction. This makes SHAP particularly useful for feature importance explanations. SHAP can be used with any machine learning model (like decision trees, neural networks, etc.), though it may run more efficiently on some models (like tree-based models) due to optimizations. The details will be covered during our presentation.

2.4.2 Working of SHAP

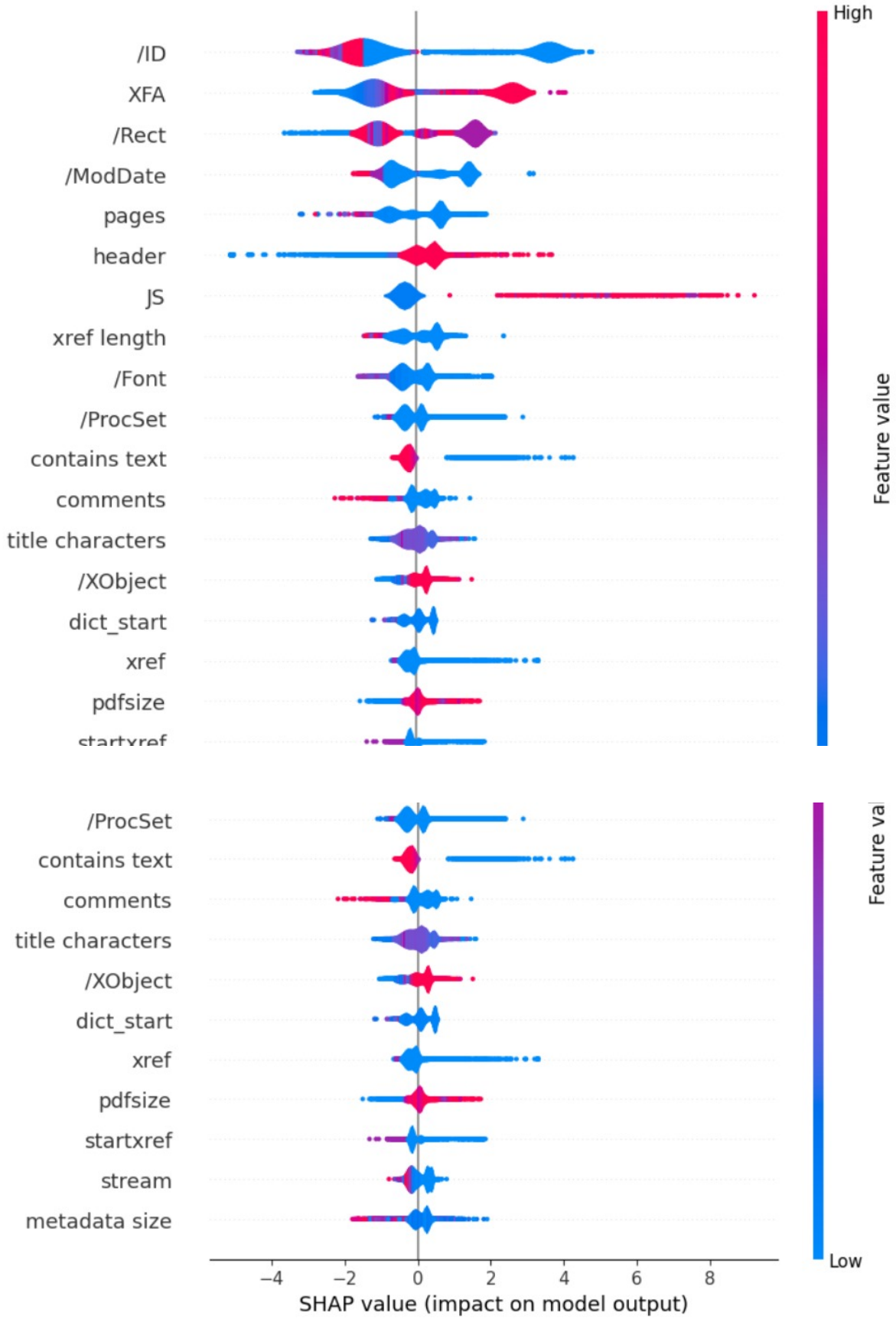
SHAP generates a baseline output (the average prediction if all features are missing) and then calculates how much each feature value changes the prediction relative to this baseline. For each feature, SHAP computes its value by considering its contribution with and without other features, averaging across all combinations to ensure fair attribution

2.4.3 Refer the Screenshots below:



The screenshots on next page depicts the impact of independent feature on Model's output as calculated by SHAP.

```
shap.plots.violin(shap_values)
```



NOTE: The above are for the XGBoost model before Hyperparameter tuning.

3 ML Model

The various independent models were trained and tested, as well as various combinations of different models were tried for stacking model on the dataset prepared by us rigorously. Highlights of this exhaustive process are enumerated in subsequent paragraphs:

3.1 The stacking model

The stacking model was dropped due to very high execution time and no significant improvement in overall accuracy. Below are the results for three stacking models -

Best Model for Each Feature Count:

	Model	Features	Accuracy	Precision	Recall	F1 Score
5	Stacking Model 2	5	73.94%	100.00%	73.94%	85.02%
6	Stacking Model 2	10	77.17%	100.00%	77.17%	87.12%
12	Stacking Model 3	15	82.02%	100.00%	82.02%	90.12%
13	Stacking Model 3	20	83.03%	100.00%	83.03%	90.73%
14	Stacking Model 3	25	83.84%	100.00%	83.84%	91.21%

Metrics for MalwareBazaar dataset

NOTE: These metrics are for the MalwareBazaar dataset.

Time issues: The top features used here are the top features given by SHAP for RandomForest, XGBoost and GBM, and not that of the Stacking Model itself because SHAP takes unusually longer time to execute for a stacking model because the number of combinations between features and models increase **exponentially**. For example, even when executing for only 100 rows of data, it took around 1 hour to run. These unusually high execution times are unrealistic and the accuracy changes are not significant. So we dropped the idea of using a Stacking model.

3.2 The XGBoost Classifier

The XGBoost classifier performed exceedingly well for all combination of features and also on the **MalwareBazaar** dataset, which is the most updated set of malicious data available on open source. Results for both balanced and imbalanced given above show that XGBoost was the better performer of the three.

The results of our final model are given in the table below -

Metric	Validation dataset	MalwareBazaar dataset
Accuracy	0.9957	0.8667
Precision	0.9955	1
Recall	0.9940	0.8667
F1 Score	0.9947	0.9286

3.3 Optuna

We have used Optuna, an open source library, for hyperparameter optimization framework which led to some increase in the overall accuracy.

Best parameters: {'n_estimators': 973, 'max_depth': 4, 'learning_rate': 0.29551846768638934, 'subsample': 0.6307508700115003, 'colsample_bytree': 0.9060368374784977, 'min_child_weight': 6}

4 Comparative Analysis and Results

The comparative analysis between the existing work in the selected research paper and our work was carried out and following are the highlights of the same:

4.1 Time taken on Enhanced and Diverse Dataset

Our model has performed exceedingly well in terms of time than the existing model on enhanced and diverse dataset prepared by us. The number of features used by us are less than that in the original paper and our model significantly outperforms their model even on live malware samples.

4.2 Accuracy on most Updated Open Source Dataset

Our model has achieved better accuracy than the existing model when tested on most updated open source dataset from MalwareBazaar as well as the original dataset. The accuracy of the papers model on the MalwareBazaar dataset was 83.03% and on the Validation dataset was 99.55%.

5 Conclusion

All the things proposed in the first deliverable of the project submitted on 07 Oct 24 have been complied with and successfully implemented in this deliverable. In addition, SHAP has been implemented for feature engineering.