Ransomware Classification and Feature Analysis Using Machine Learning Models: Online Appendix

Faithful Chiagoziem Onwuegbuche[1,2] [0000−0001−9580−4260], Anca Delia Jurcut[2] [0000−0002−2705−1823], and Liliana Pasquale[2] [0000−0001−9673−3054]

[1] SFI Center for Research Training in Machine Learning (ML-Labs)
faithful.chiagoziemonwuegb@ucdconnect.ie
[2] School of Computing, University College Dublin, Ireland
{anca.jurcut, liliana.pasquale}@ucd.ie

Abstract. This online appendix contains the following items. First, in Appendix A we provide the different hyperparameters that were used in tuning the machine learning models. It also contains the ranking of the performance of the different models for the binary and multi-class ransomware classification. Second, appendix B provides the feature names of the top 30 features determined by SHAP for the binary and multi-class classification. Appendix C and D contains the LIME explanation for the binary and multi-class classification respectively.

Keywords: Ransomware detection · Malware classification · Machine learning · Feature analysis.

1 Appendix A - Hyperparameters and model ranking

Table 1. Hyperparameters tuned for the different classifiers

Key	**Values**
Logistic Regression	
solvers	['newton-cg', 'lbfgs']
c_values	[1000, 100, 10, 1.0, 0.1, 0.01, 0.001]
Random Forest, Decision Trees	
criterion	['gini', 'entropy']
max_depth	[2, 4, 5, 8, 10, 16, 32, None]
min_samples_leaf	[1,2,3,4,5,6,7,8,9,10,11]
XGBoost	
booster	['gbtree', 'gblinear', 'dart']
eta	[0, 0.01, 0.2, 0.3]
SVM	
C	[0.1, 1, 10, 100, 1000]
gamma	[1, 0.1, 0.01, 0.001, 0.0001]
kernel	['rbf']

Table 2. Ranking of performance of the different models for the binary and multi-class classification of ransomware

Models	**Binary classification**			**Multi-class classification**		
	Standard	**SMOTE**	**COST**	**Standard**	**SMOTE**	**COST**
DT	3	2	1	2	1	1
LR	2	1	1	2	3	1
RF	2	1	3	1	3	2
SVM	2	3	1	3	1	2
XGB	1	2	1	2	1	3
Average	2	1.8	1.4	2	1.8	1.8

Table 2 provides an overall look at the performance of the different approaches across their respective models. The SMOTE and cost sensitive learning models show similar performance on average for the multi-class classification while cost-sensitive models perform better than SMOTE and standard approach for the binary classification.

2 Appendix B - Top 30 feature names

Table 3. Top 30 feature names for the binary classification

Feature Number	**Feature Name**	**Feature Number**	**Feature Name**
169	API:CoInitializeEx	2607	REG:OPENED:HKEY_LOCAL_MACHINE \SOFTWARE\Microsoft\
107	API:SetErrorMode	2916	REG:OPENED:HKEY_LOCAL_MACHINE \SOFTWARE\Classes\Directory\
12243	FILES_EXT:WRITTEN:dll	369	DROP:bmp
158	API:GetTempPathW	132	API:DeviceIoControl
231	API:GetSystemMetrics	69	API:OleInitialize
86	API:FindResourceW	172	API:NtCreateSection
192	API:WSAStartup	2966	REG:OPENED:HKEY_ CURRENT_USER\Software\CodeGear\
104	API:SizeofResource	1219	REG:OPENED:HKEY_LOCAL_MACHINE \Software\Policies\Microsoft\Windows NT\
148	API:RegEnumKeyW	126	API:NtUnmapViewOfSection
195	API:LdrGetProcedureAddress	106	API:GetSystemWindowsDirectoryW
16	API:VirtualProtectEx	168	API:Process32FirstW
159	API:FindResourceExW	225	API:GetFileType
11402	FILES_EXT:OPENED:Manifest	210	API:NtCreateMutant
166	API:GetNativeSystemInfo	202	API:CoInitializeSecurity
110	API:SetFileAttributesW	229	API:RegOpenKeyExA

Table 4. Top 30 feature names for the multi-class classification

Feature Number	**Feature Name**	**Feature Number**	**Feature Name**
169	API:CoInitializeEx	2607	REG:OPENED:HKEY_LOCAL_MACHINE \SOFTWARE\Microsoft\
107	API:SetErrorMode	2916	REG:OPENED:HKEY_LOCAL_MACHINE \SOFTWARE\Classes\Directory\
231	API:GetSystemMetrics	153	API:NtOpenThread
158	API:GetTempPathW	132	API:DeviceIoControl
234	API:FindWindowW	69	API:OleInitialize
86	API:FindResourceW	172	API:NtCreateSection
192	API:WSAStartup	89	API:Module32NextW
104	API:SizeofResource	1219	REG:OPENED:HKEY_LOCAL_MACHINE \Software\Policies\Microsoft\Windows NT\
148	API:RegEnumKeyW	126	API:NtUnmapViewOfSection
195	API:LdrGetProcedureAddress	106	API:GetSystemWindowsDirectoryW
16	API:VirtualProtectEx	168	API:Process32FirstW
159	API:FindResourceExW	225	API:GetFileType
11402	FILES_EXT:OPENED:Manifest	210	API:NtCreateMutant
166	API:GetNativeSystemInfo	202	API:CoInitializeSecurity
109	API:NtWriteVirtualMemory	229	API:RegOpenKeyExA

3 Appendix C - LIME explanation for ransomware binary classification

Using LIME we perform some local model explainability to understand why some samples were correctly or wrongly classified by the model for the binary classification.

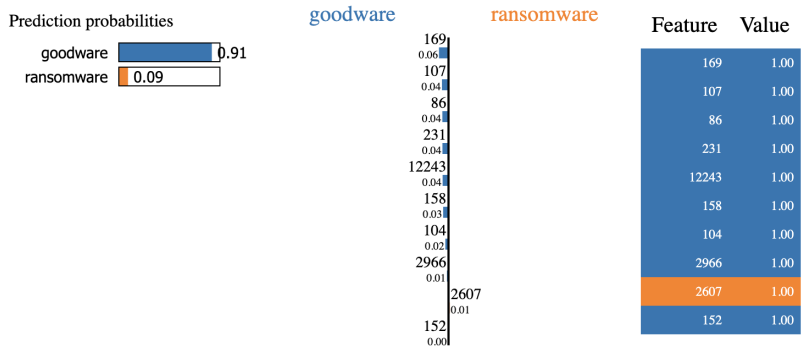

Fig. 1. LIME explanation for a correctly classified goodware sample

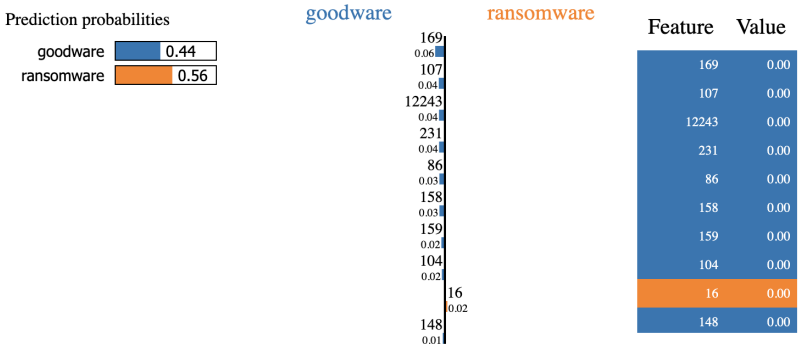

Fig. 2. LIME explanation for a correctly classified ransomware sample

Figure 1 and 2 shows a correctly classified sample (a goodware sample that was correctly classified as a goodware and a ransomware correctly classified as a ransomware sample by the model). The vertical line between goodware and ransomware indicates the 10 most significant features responsible for the prediction of the model. The features that contributes to the model being classified as goodware are shown in blue while those that contribute to the model being classified as ransomware are shown in orange.

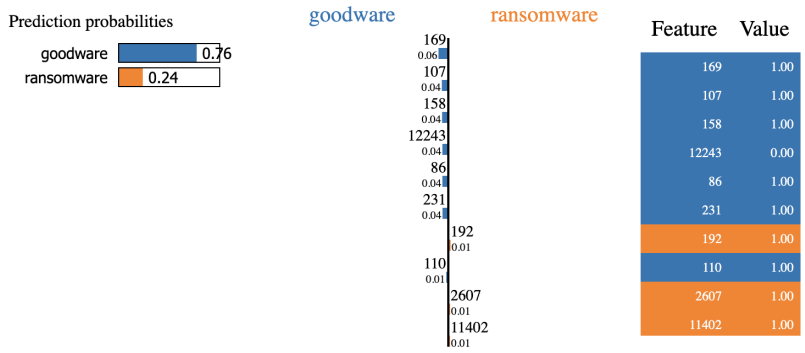

Fig. 3. LIME explanation for a ransomware sample incorrectly classified as goodware

Figure 3 shows a ransomware sample that was misclassified as a goodware sample by the model. Some of the features responsible for the model being classified as a goodware sample were *API:CoInitializeEx*, *API:SetErrorMode*, *API:GetTempPathW*, *FILES_EXT:WRITTEN:dll* and *API:FindResourceW*.

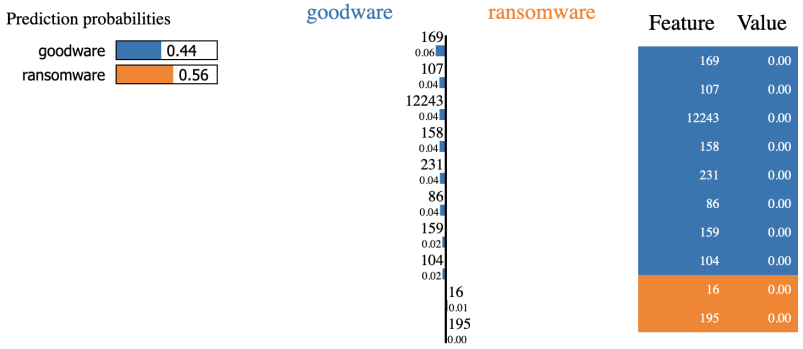

Fig. 4. LIME explanation for a goodware sample incorrectly classified as ransomware

Figure 4 shows a goodware sample that has been incorrectly classified as a ransomware sample. Some of the features that contributed to the sample being misclassified as ransomware were *API:CoInitializeEx* and *API:LdrGetProcedureAddress*.

4 Appendix D - LIME explanation for ransomware multi-class classification

Using LIME we perform some local model explainability to understand while some samples were correctly or wrongly classified by the model for the multi-class classification.

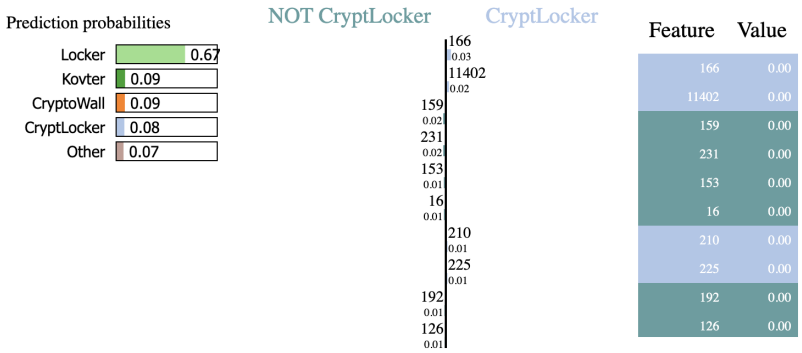

Fig. 5. LIME explanation for a correctly classified ransomware family - Locker

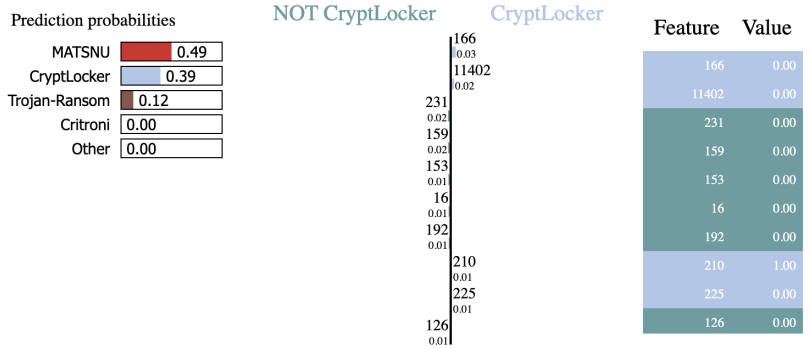

Fig. 6. LIME explanation for a correctly classified ransomware family - MATSNU

Figure 5 and 8 shows a correctly classified sample (locker and MATSNU respectively) for the multi-class classification. The vertical line between NOT CryptLocker and CryptLocker indicates the 10 most significant features responsible for the prediction of the model. The features that contributes to the model being classified as a specific ransomware family with different colours corresponding to the specific family as seen in the colour key on the left.

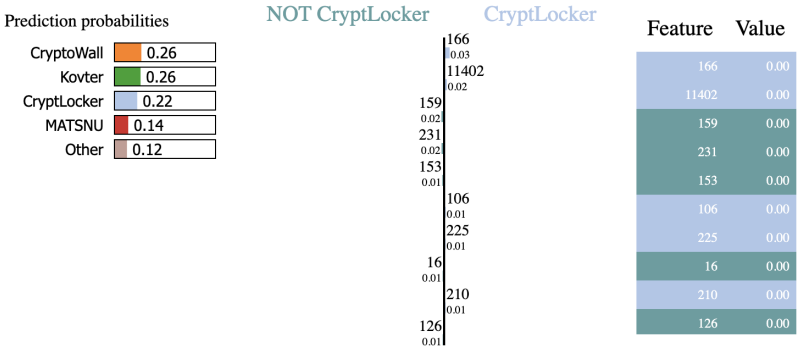

Fig. 7. LIME explanation for a Locker misclassified as CryptoWall

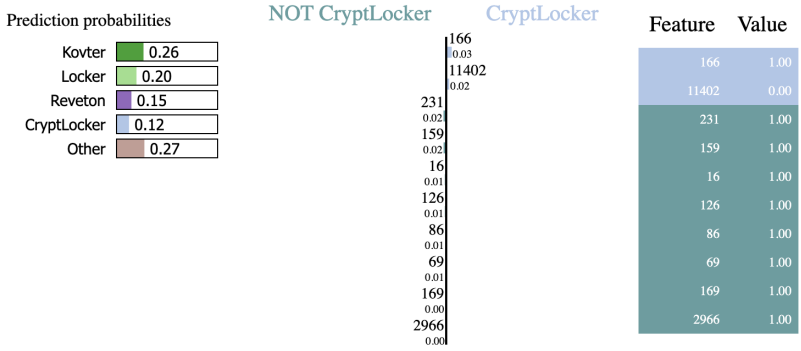

Fig. 8. LIME explanation for a CryptoWall misclassified as KOLLAH