CS5242 Project Report

Group 17 - Workload Distribution

Chan Jun Wei (e0374282) : Model Training, Model Tuning, Feature Extraction, Feature Selection
Chua Chin Siang (e0403439) : Data Preprocessing, Feature Extraction, Model Training
Tey Shiwei (e0403430) : Data Exploration, Data Preprocessing, Model Training

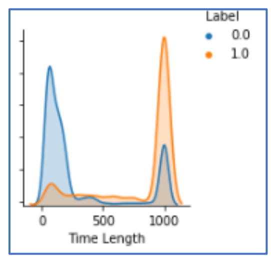
Background

The data provided was categorized in 9 feature groups. Preliminary validation works show that training the model based on feature groupings yields a better result. The following table is a summary of the data characteristics that is fitted into the final model. Note that FG stands for Feature Group.

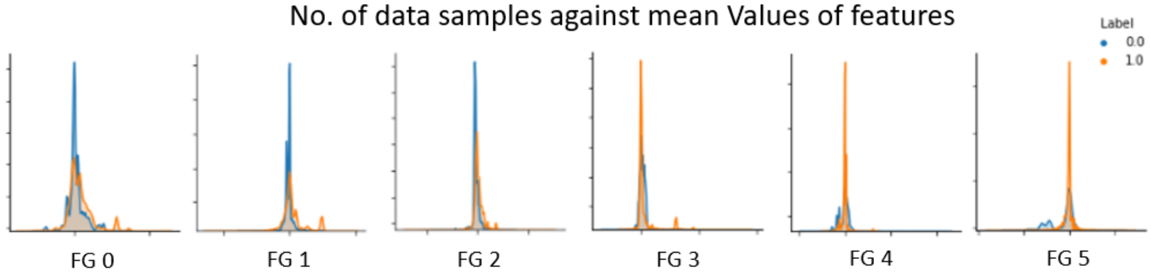
FG	Name	Mean	Sum	Max	Mean w/o 0s	Non-zero	Data type	Pre-Processing	Chosen
FG 0	API Name	0.03176	30.72	1.35	0.1847	28%	Hashing tricks	One-hot, Frequency	Yes
FG 1	API Category	0.02274	30.23	1.59	0.0178	65%	Hashing tricks	One-hot, Frequency	Yes
FG 2	Arguments Int	-0.00225	8.72	3.01	-1.8244	4%	Hashing tricks	One-hot, Frequency	Yes
FG 3	Paths	0.00964	6.76	0.87	0.2162	2%	Hashing tricks	One-hot, Frequency	No
FG 4	Dlls	-0.00093	0.11	0.92	0.0194	2%	Hashing tricks	One-hot, Frequency	Yes
FG 5	Registry Keys	-0.00784	-4.32	0.73	0.0053	4%	Hashing tricks	One-hot, Frequency	No
FG 6	Urls	0.000002	0.0019	0.0018	0.1893	0.00%	Hashing tricks	One-hot, Frequency	No
FG 7	IPs	0.000005	0.0062	0.0147	-0.089	0.02%	Hashing tricks	One-hot, Frequency	No
FG 8	Statistics	5.72203	3625.02	76.5	14.8809	26%	String Statistics	Frequency, Mean	Yes

Data Exploration

1. P1: Data Sparsity – Most of the features have very few data. As shown in the non-zero column of the table above, FG 6 and FG 7 have very few useful where feature 1 has almost 65% of filled data.
2. P2: Time length – It was found that time length of the data plays an important role in defining malwares. Based on the figure on the right, malwares tend to have a longer time length based on intuition. Time length was represented as frequency in our training model.
3. P3: Each different feature has a unique mean, sum and max distribution. Below is the sample of mean distribution for the first 6 features and the trend for malware is obvious.



No. of data samples against mean Values of features



Traditional Model Attempt

Data Preprocessing

As the data provided are feature-engineered, data preprocessing is not necessary. However, to fit the data into the traditional model such as LightGBM, One-hot encoding were used on hashed strings and integers (FG 0- FG 7).

Model Training

LightGBM is selected to train the transformed data. As the model which scores well in our validation dataset might not score well in Kaggle, only Kaggle Public Score is shown. Our model is tuned using HyperOpt due to its efficiency.

Features Extraction

We did a few experiments with respect to TSFresh and Frequency:

1. As P2 shows the importance of the time length, we sum one-hot Encoding of FG 0 - FG 7 into a row and denote it as "One Hot Feature". As Feature 8 is statistics, we extract the sum and mean of them as the features of the transformed data, we can call them "Stats Feature". They are all referred as "Frequency".
2. TSFresh library is used to extract features of time series data. Given a multivariable time series, TSFresh will return many time series characteristics. It is used to make sure we did not miss out other characteristics.

As shown in the table below, Frequency perform better than TSFresh. Thus, only Frequency is included at the end.

FG	Preprocessing	Model	Tuning	Public Score
All	TSFresh	LightGBM	None	0.98533
0,1,2,3,4,5,6,7	**Frequency**	**LightGBM**	**None**	**0.98893**
0,1,2,3,4,5,6,7 (Frequency), All (TSFresh)	Frequency + TSFresh	LightGBM	HyperOpt	0.98881
0,1,2,3,4,5,6,7 (Frequency), 8 (TSFresh)	Frequency + TSFresh	LightGBM	HyperOpt	0.98855

Feature Groups Selection

Feature Groups were chosen based on the top 5 individual AUC scores as shown in the below table. The results were further validated by training all the 2^9 combinations.

Score	FG 0	FG 1	FG 2	FG 3	FG 4	FG 5	FG 6	FG 7	FG 8
AUC	**97.990834**	**97.899485**	**97.758197**	96.400343	**96.698516**	96.604820	50.652901	51.864846	**96.611583**

The result is also verified in the experiment result shown below.

FG	Preprocessing	Model	Tuning	Public Score
0,1,2,4,6	Frequency	LightGBM	HyperOpt	0.98984
0,1,2,4,6,8	Frequency	LightGBM	HyperOpt	0.99025
0,1,2,4,8	**Frequency**	**LightGBM**	**HyperOpt**	**0.99036**

Features Selection

FG 0 – FG7: After transformations, the features in FG 0 – FG 7 are one-hot encodings, applying feature selection to these feature groups does not make sense to us, hence no features selection technique is applied to them.

FG 8: String statistics were handled by the frequency method and column 7,9,10 (numUrls, numRegistryKeys and numMZ) were excluded due to low importance based on LightGBM model. We will refer this as "Selected Stats".

Transformation	FG 8 Column Importance based on LightGBM									
	numStrings	avLength	numChars	entropy	numPaths	numDlls	numUrls	numIPs	numRegistryKeys	numMZ
Sum	**179**	**164**	**108**	**154**	**215**	**215**	39	**158**	78	105
Mean	**231**	**176**	**185**	**210**	**212**	**245**	52	**174**	31	69

After applying the feature selection, the result is improved as shown in the table below.

FG	Preprocessing	Model	Tuning	Public Score
01248	Frequency, Stats	LightGBM	HyperOpt	0.99036
01248	Frequency + Selected Stats	LightGBM	Hyper opt	0.99056

Neural Network Exploration

The input data for dynamic malware analysis project has a fix feature size with varying time length. Hence, we preprocess this data with zero padding for them to have same shape. With a simple convolution network, it scores 95% as AUC score. Several Neural Network architectures were explored, such as VGG and Inception but their results were comparatively poor. Possible reason could be due to the complexity of VGG/Inception which doesn't fit into our small training data. Natural curiosity leads us to try manual implementation of convolutional, pooling, dropout layers, however the performance of AlexNet couldn't be surpassed. Our final NN model is done with AlexNet + Dropout + Early Stopping which gives us 97% AUC before investigating feature selections. At last, Time series data is padded to 500 length so that it runs smoothly in our laptop without affecting the auc. However, LightGBM scores significantly better.

FG	Preprocessing	Model	Tuning	Public Score
All	Zero-padding, max length: 1000	Simple Convolution Network	None	0.9509
All	Zero-padding, max length: 1000	VGG	None	0.96737
All	Zero-padding, max length: 1000	AlexNet with dropout	None	0.97464
All	Zero-padding, max length: 500	AlexNet with dropout	None	0.97878
All	Zero-padding, max length: 500	AlexNet with dropout	Early Stopping	0.97737
012345	Zero-padding, max length: 500	AlexNet with dropout for each features and averaging results	Early Stopping	0.97953
01248	**Frequency with Selected Stats**	**LightGBM**	**HyperOpt**	**0.99056**

Image Approach

We first convert time series with 100 length to image using Recurrence Plot, which is the correlation matrix of time length. Then, we tried CNN for Recurrence Plot suggested by Tigurius (2018) for the submission. However, training the model takes a long time and LightGBM is better, thus we give up on this approach.

FG	Preprocessing	Model	Tuning	Public Score
All	100 * 100 Recurrence Plot	Simple Convolution Network	None	0.95544
01248	**Frequency with Selected Stats**	**LightGBM**	**HyperOpt**	**0.99056**

Final Model

FG	Preprocessing	Model	Tuning	Public Score	Private Score
01248	**Frequency with Selected Stats**	**LightGBM**	**HyperOpt**	**0.99056**	**0.99044**

References

tsfresh. (n.d.). Retrieved from https://tsfresh.readthedocs.io/en/latest/.

LightGBM. (n.d.). Retrieved from https://lightgbm.readthedocs.io/en/latest/.

Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830.

Bergstra, J., Yamins, D., Cox, D. D. (2013) Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. To appear in Proc. of the 30th International Conference on Machine Learning (ICML 2013).

Tigurius. (2018) Recurrence plots and CNNs for time-series classification. Retrieved from https://www.kaggle.com/tigurius/recuplots-and-cnns-for-time-series-classification