Cyber Analytics Visual Analytics Team 1 Project 1 Report

Zicheng Liu
Department of Computer Science
University of Delaware
zcliu@udel.edu

Ezeanaka Kingsley. U
Department of Computer Science
University of Delaware
ekingsly@udel.edu

Wanxin Li
Department of Computer Science
University of Delaware
wanxinli@udel.edu

Abdulrahman Alshammari
Department of Computer Science
University of Delaware
aturqi@udel.edu

ABSTRACT

This report recorded an experiment done by graduate students from Cyber Analytics Course lectured by John Cavazos from University of Delaware made an extension to an technique called Class Signature to analyze a dataset of 1100 malware samples to distinguish discriminative features of malwares. The target is to set up a widget consisting of 66 T-SNE map representing 66 sub-cluster of 11 malware families with 347 features with 10 top ranked discriminative features below.

1. INTRODUCTION

Class Signature proposes a visual analytics workflow to interpret predictive associations between a large set of binary features and a binary target. It usually uses 4 step pipelines, Model, Contrast, Cluster and Rank.[1]. In our Project, we extended this technique to apply it on multiple targets not binary anymore. Our target is to use Class Signature to analyze a dataset of 1100 malware samples with 347 features and 11 target families, to distinguish discriminative features among each sub-cluster of 11 families as shown in Figure 1.

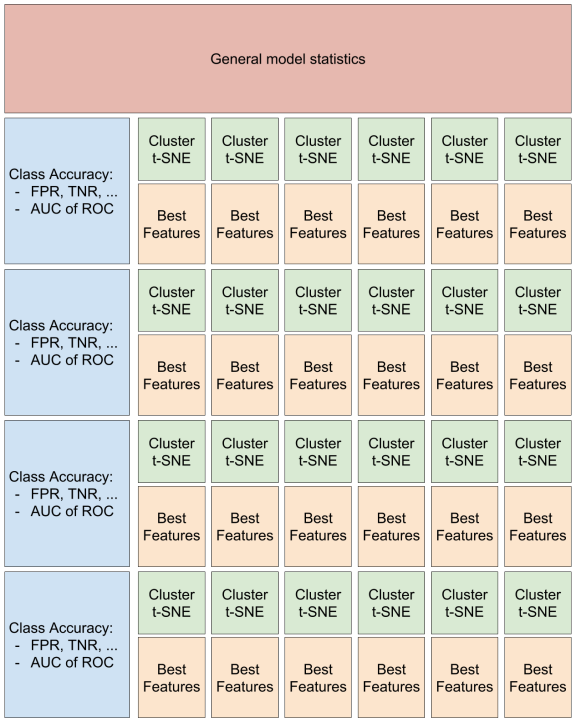


Fig.1 Target Sample

1.1 Overview of Clustering

The original data we got from Tristan are malware ID in sha256 values labeled with family names

(financial-100-c.csv) and correspondent malware feature vectors(financial-100-c-metrics.csv). There are 3 modules to accomplish our clustering process. Firstly, we needed to apply pre-processing work on the original data. We split the malware samples into 11 groups in accordance to malware family names. Secondly, we applied built-in k-means method of Scit-learn library on each family after splitting original dataset. The K-means generated 6 sub-clusters for 11 malware families. This groups together malware samples that have the similar configuration of features. Thirdly, as for generating proper input for T-SNE and Feature Ranking, Module 3 generated 66 label files in accordance to 66 sub-clusters of original dataset. The 66 label files consist of value 1 and value 0, where value 1 indicates a sub-cluster of malware samples that should be highlighted in T-SNE phase and correspondent features be analyzed in Feature Ranking Phase.

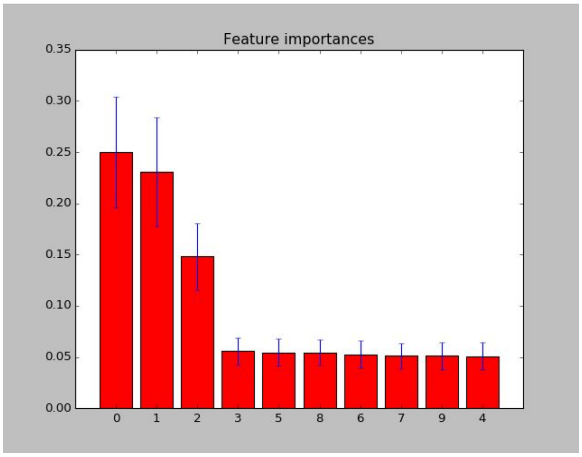
1.2 Overview of T-SNE

T-Distributed Stochastic Neighbor Embedding (t-SNE) is a technique that is used for reducing the dimensions of a particular data that has too many dimensions and visualize the data in a convenient way.[2] In this project, we applied clustering algorithms on the data and generate many sub-clusters. In each sub-cluster, we would like to highlight where the instances located when we visualize it.

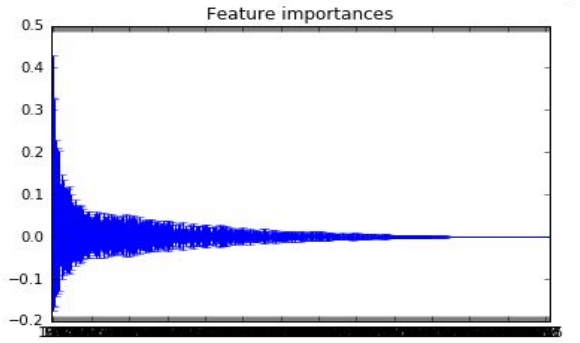
1.3 Overview of Feature ranking

In the area of machine learning and statistics, feature selection is a term commonly used to refer to the process of selecting a subset of relevant features, which could be variables or predictors, for using when developing a training model. The process of grouping the entire set of features in some order of relevance is termed feature ranking. The values associated with each rank are called feature importance values or feature ranking values. The idea is based on the fact that datasets are bound to contain many features that are either redundant or irrelevant, and thus can be removed without incurring much of information[4].

Feature selection techniques simplify models for easier interpretation by researchers and users, reduce model training times, avoid unnecessary complications associated with dimensionality and also to enhance generalization of dataset by reducing overfitting.[5] A number of algorithms are used to compute the importance scores of various features of any dataset. The most common ones which are utilized in the project include: Linear Regression, Ridge method, Lasso algorithm, MIC model, and Random forest.



(a)



(b)

Fig.2 a) 10 − feature dataset b) 347 − feature dataset

Figure 2 above illustrates a visual picture of the relative importances of features in a dataset. The vertical axis represents the range of importance scores and the horizontal axis represent the labels for the features. The features are sorted with the most important features coming first.

The error bars are reflective of the OOB (out of bag error) estimates which calculates the mean prediction error on each data point of the dataset being fit into the random forest and is averaged over the entire forest. The importance score for the j-th feature is computed by averaging the difference in out-of-bag error before and after the permutation over all trees. The score is normalized by the standard deviation of these differences.

2. RELATED WORK

Scikit-learn (formerly scikits.learn)is a machine learning library for the Python programming language. It features various classification, regression, and clustering algorithms including support vector machines, random forests, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries Numpy and Scipy. [6]

K-means clustering is a method of vector quantization.
It aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. [7]

For high dimensionality' reduction, there are some tools beside T-SNE used for visualization like PCA which stands for Principal Components Analysis. This technique is used to emphasize variation and make data easy to visualize. It obviously reduces the number of dimensions of a complex data set. PCA works better with linear data. Compared with T-SNE, PCA is a free parameter which T-SNE relies on many different parameters such as learning rate and number of iterations.

Feature ranking and selection is a fairly popular concept in the field of machine learning, and as such, there are a number of other algorithms in the industry that also compute the importance scores of dataset features. A number of them worthy of note include the recursive feature elimination algorithm. Also related with feature ranking are the dimensionality reduction techniques which adopt the similar goal of producing a compressed version of the datasets.

Confusion matrix is required for our target.

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class (or vice versa).[2] The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabelling one as another).[8]

It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table).

3. PROGRAM PIPELINES

We divided our entire project work into 4 steps.
Step 1 is to generate 6 clusters for each family of malware. Step 2 is to visualize the entire dataset of malware using t-SNE and highlight each cluster, thus we will have 66 t-SNE figures, one for each cluster). Step 3 is to compute Feature Ranking for each cluster. Step 4 is to automate this process and apply on all datasets.

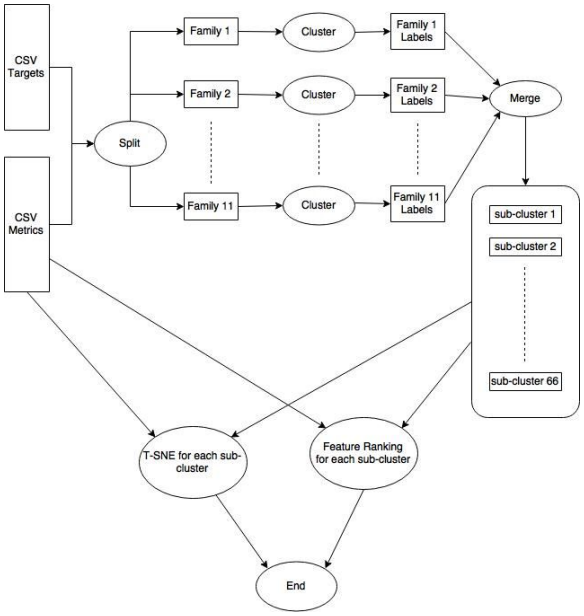


Fig.3 Program Workflow

Clustering

As the project pipeline shows, the first part of this project is to generate sub-clustered files as inputs for feature ranking and T-SNE. In project 1, we choose the subset c from course Github repository as the original input data for clustering. This subset contains 1,100 malwares which are all recorded in two CSV files. The first CSV file "Metrics" records malware ID and matching features. There are 347 feature columns in total. The second CSV file "Targets" records family label in matching order. There are 11 different family labels in total. And each family contains 100 malwares.

The goal for clustering part is to generate 6 sub-clusters for each family based on K-Means algorithm. And we will get 66 sub-clusters in total. We divide this clustering part into three modules: Split, Cluster, Merge. All three modules are written in a same python script "cluster.py".

For the first module, we need to split 1,100 malwares into 11 groups based on family labels. Our solution is to import all values from "Targets" and "Metrics" into a python two-dimensional list data structure. Each row contains a malware's ID, 347 features and matching family label. Then we sort the 1,100 malwares based on family label in alphabetical order. After that, we can split the 1,100 malwares into 11 groups, which are saved in 11 two-dimensional list data structure. In the meanwhile, the family labels are deleted before saving into these 11 new sub-lists. Because the family label are not the expected input for K-means algorithm. The new sub-lists are in the same format with the 1,100 malwares list. The only difference is that the new sub-list contains 100 rows of malwares.

For module 2, we applied built-in K-Means method of Scit-learn library on each family after splitting original dataset. The K-Means generated 6 sub-clusters for 11 malware families. This groups together malware samples that have the similar configuration of features.

After applying K-Means algorithm for each family, we generate 11 clustered label lists, each list has two column: malware ID and clustered label value. These clustered labels are represented in integer from 0 to 5, which are 6 different sub-clusters for a specific family. For the third module, we need to merge these 11 clustered label lists and output 66 label files in CSV or TXT format as the inputs for feature ranking and T-SNE part.

For module 3, we firstly sum up all the 11 clustered label lists into one list which contains the whole 1,100 malwares. Obviously, this list has a wider range of clustered label values from Integer 1 to 66. Next, we use this list to generate 66 sub-cluster label files. The file format is CSV. Each label file has 1,100 rows and two columns: malware ID and label value. The label value contains only integer 1 or 0. The integer 1 represents the sub-cluster which we want to display as highlight in T-SNE. The integer 0 represents the other 65 sub-clusters which we want to display as the background in T-SNE.

T-SNE

In this project, we applied clustering algorithms on the data and generate many sub-clusters. In each sub-cluster, we would like to highlight where the instances located when we visualize it. To do that, we apply T-SNE on each sub-cluster and we try visualize only the data belongs to the selected one as one color, and the rest of other sub-clusters as a different color. We will do the same process among all other sub-clusters. To run that, the T-SNE script is written in python that takes two parameters, the whole clustered data and the labels (sub-cluster). The expected result will be two dimensional plot with many points where represent instances from clustered data. If the instance belongs to the selected sub-clustered, it will have a certain color, otherwise it will have the second color. Note that the implementation is available in many different languages, but we pick python for simplicity. For explanation, figure 4 is an example of one of sub-clusters TSNE. If a sample belongs to the selected sub-cluster, it will be represented as a green point. Otherwise, the sample will be represented as a dark red point as shown in.

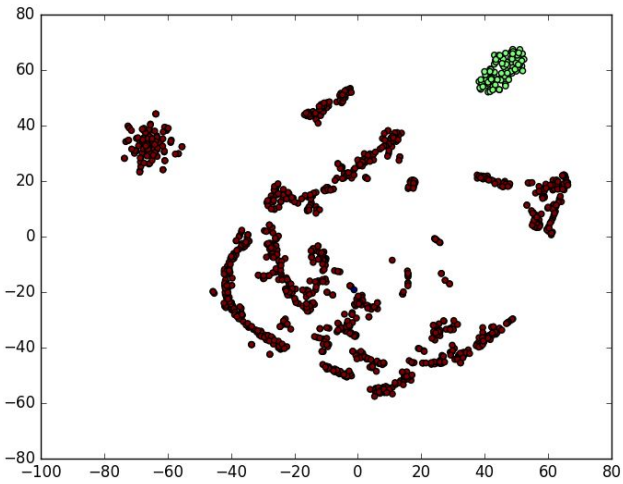


Fig.4 A result example of TSNE

Feature Ranking

Thus we ranked 347 features in the computed clusters using discriminative analysis across all 66 clusters.

The feature ranking program that generates the GUI. The GUI takes a pair of datasets as inputs the first input is the input dataset with as many features and any of the 66 labels or target datasets containing the clusters of output. The user is prompted to choose any of the listed algorithms with which to train the dataset. Choosing any of the algorithm buttons activates the corresponding algorithm, subsequently trains the dataset and produces the list of importance values sorted in descending order for that particular cluster. By selecting each of the seven algorithms listed on the GUI we get a sense of the how each of the methods compare against each other on the same dataset, and thus, get a sense of the optimal algorithm for such datasets.

Figure 2 shows the distribution of ranking scores with respect to instance of the dataset cluster pair, As we can observe, with **linear correlation**, the features are evaluated independently, the noise features tend to be grouped together at the bottom of the ranking list (in fact, this applies almost all other methods except for MIC). It's also clear that while the method is able to measure the linear correlation between each feature and the response variable, it is not optimal for selecting the top performing features for improving the generalization of a model, since all top performing features would essentially be picked twice.

Lasso picks out the top performing features, while forcing other features to be close to zero. It is clearly useful when reducing the number of features is required, but not necessarily for data interpretation.

MIC is similar to correlation coefficient in treating all features "equally", additionally it is able to find the non-linear a relationship between ***X***3 and the response.

Random forest's impurity based ranking is typically aggressive in the sense that there is a sharp drop-off of scores after the first few top ones. .

Ridge regression forces regressions coefficients to spread out similarly between correlated variables. This is clearly visible in the example where similar features tend to be ranked close to each other in terms of scores.

Stability selection is often able to make a useful compromise between data interpretation and top feature selection for model improvement. This is illustrated well in the example. Just like Lasso it is able to identify the top features. At the same time their correlated shadow variables also get a high score, illustrating their relation with the response.[9]

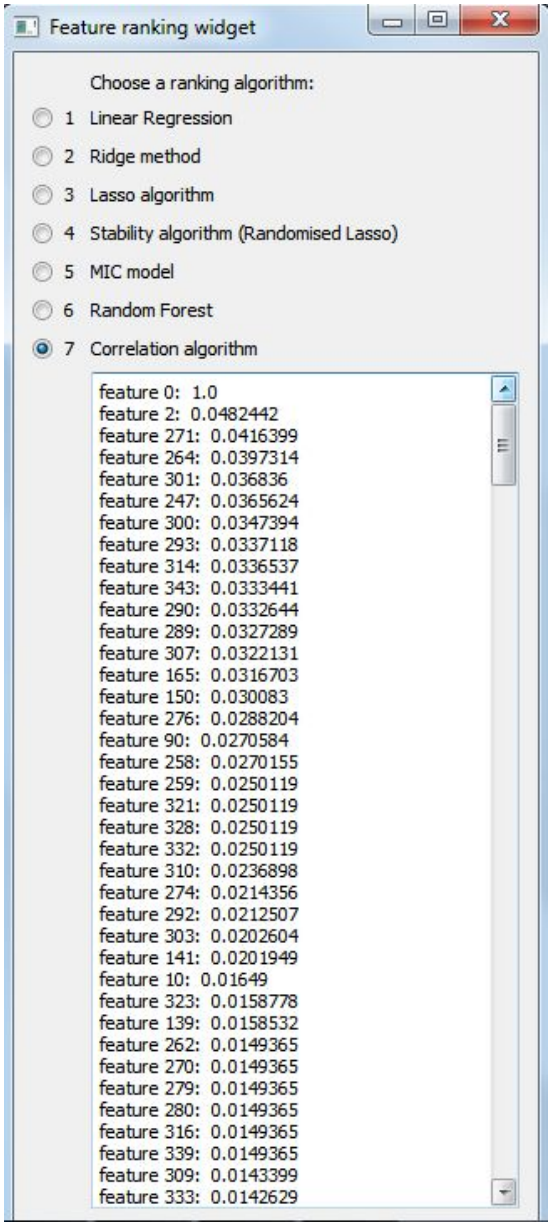


Fig.5 the widget after executing the Lin. correlation algorithm

As shown in the Fig. 5 above, the feature ranking values are shown to be sorted from the important for the selected algorithm, down to the list important, hence the most important features are easily spotted at the top of the list.

4. LIMITATIONS

In our project, we use our cluster.py program to accept the original data files and generate output files that can be used as input for TSNE and Feature Ranking. However, our

labor was divided into these three main function blocks, which causes that there are some asynchronized parts among them. For example, cluster program has to be developed first to ensure input for TSNE and Feature Ranking. We developed two versions of cluster program during project 1, where the first version can generate TXT files as outputs and the second version can generate csv files as outputs. And the TSNE can take csv files as input, whereas Feature Ranking code can take txt files as the input. So that our first version code of cluster can serve Feature Ranking and the second version can serve TSNE.Thus, the whole program can not be ran in a single run. Since the first version of cluster code has been overridden, only output in txt format of first version can be accessible in our Repository's Feature Ranking input folder.

Below few points summarize the difficulties we faced in the TSNE part.Firstly, we start learning Python when we realized that it is needed to write T-SNE script in Python. The workflow of this part would be faster if we depend on a language such as Java.Secondly, The result of each time we run the T-SNE script has different shapes despite the fact that have the same patterns of design. This is not related to the script itself, but this is one of T-SNE features. Thirdly, to ensure that we get the right result design, we decided to run the data on T-SNE java versions and compare the two results. This expand time of getting the final result.

As for the challenges encountered when carrying out the feature ranking task, First and foremost, prior to taking up this project, I had little or beginner level experience with the python programming language. I had to learn on my feet. Most of what I coded was done after looking up resources online, implementing the things learned into ways suitable for my code. As a result, certain tasks that would have otherwise been easier to implement for seasoned python developers took a little extra time to undertake.

The feature ranking GUI is currently takes a while to train the data when some algorithms are utilized. For instance, the stability algorithm doesn't work well with our dataset as it produces 0's across for all the features. The GUI currently doesn't distinguish between a clustered dataset and a raw untampered dataset.

The feature ranking code was developed in the python 3.5 interpretation while the code for the clustering and T-SNE visualisation were implemented in python 2.7. So we are not able to easily combine both codes for this part of the project at this time.

5. CONCLUSIONS & FUTURE WORK

Feature ranking is very useful in both machine learning and data mining. We can verify how well a method works compared to others when we select top features to improve the performance of the model. Cross validation helps us to achieve this. It's not as straightforward when using feature ranking for data interpretation, where stability of the ranking method is crucial and a method that doesn't have this property (such as lasso) could easily lead to incorrect conclusions. It helps to carry out some subsampling the data and running the selection algorithms on the subsets. Consistency of outputs across the subsets is usually a sign that it is relatively safe to trust the stability of the method on this particular data and therefore easy to interpret the data in terms of the ranking.

In project 1, we import malware subset c [1] as the original input for class signature process, which contains 1,100 malwares. For next step, we will import the whole malware dataset into the class signature process, which will contain 11,000 malwares in total. In clustering part, we will handle tenfold malwares data comparing with project 1. The output will still be the 66 sub-clustered files. However, each file will contains 11,000 rows instead of 1,100 in project 1.

Besides that, we also need to develop a prediction model, which will import nine malware subsets as training set and the rest one as testing set. Through this way, we can get the confusion matrix, FPR-TPR, Recall-Precision, Threshold-Accuracy graphs.

The plan for the part 2 of the project is, among other things, to incorporate the clustering algorithms into the GUI. We intend to reconcile the differences in the python interpretations i.e to have everyone on the same page as regards to the python version to adopt for the project.

In term of T-SNE, we plan to work hard to see we can generate the same result, for example the same shape, and try to use different colors that we can easily indicate instances belong to a selected subclass. Beside that, we will work to generate other components of the Class Signatures such as confusion Matrix. we plan to write a script that get two names, the dataset and data labels and generate the TSNE.

In term of clustering part, we will create a program that can serve both our subsequent phases -- TSNE and Feature Ranking.

6. REFERENCES

[1] Krause, J., Perer, A., & Bertini, E. (2016, 06). Using Visual Analytics to Interpret Predictive Machine Learning Models. *WHI 2016*.

[2] G. H. Laurens van der Maaten. Visualizing data using t-sne. Journal of Machine Learning Research , 9, Nov 2008.

[3] L. van der Maaten. t-SNE Laurens van der Maaten. https://lvdmaaten.github.io/tsne/ , 2017.

[4] Bermingham, Mairead L.; Pong-Wong, Ricardo; Spiliopoulou, Athina; Hayward, Caroline; Rudan, Igor; Campbell, Harry; Wright, Alan F.; Wilson, James F.; Agakov, Felix; Navarro, Pau; Haley, Chris S. (2015). *"Application of high-dimensional feature selection: evaluation for genomic prediction in man"*. *Sci. Rep.* .

[5] Gareth James; Daniela Witten; Trevor Hastie; Robert Tibshirani (2013). *An Introduction to Statistical Learning*. Springer. p. 204.

[6] Wikipedia. (2017, April 10). *scikit-learn*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Scikit-learn

[7] Wikipedia. (2017, April 9). *k-means clustering*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/K-means_clustering

[8] Wikipedia. (2017, March 27). *Confusion matrix*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Confusion_matrix

[9] Ming Liang's blog. Selecting good features – Part IV: stability selection, RFE and everything side by side.

http://blog.datadive.net/selecting-good-features-part-iv-stability-selection-rfe-and-everything-side-by-side/