Foundations of Data Science
Final Project

La Sapienza University, Computer Science

A.Y. 2020-2021

Binary Functions Classification

Andrea Gasparini, 1813486
Edoardo Di Paolo, 1728334
Cirillo Atalla, 1755033

Contents

1 Introduction

The task of this project was to build a *multiclass* classifier to determine whether an assembly function is an **encryption**, **math**, **string manipulation** or **sorting** function. Formally, the task was to approximate a mapping function:

$$f : X_{bin} \rightarrow \{encryption, math, string, sort\}$$

where X_{bin} is an assembly set of instructions that can be classified in one of the target classes of the f function's right-hand side. To be more precise, since we have to define a features list to train a Machine Learning model, X_{bin} is a list of measurable properties extracted from the dataset after a feature extraction procedure.

2 Motivations

The functions classification is a useful task, for example, in the *malware analysis*. Certain Machine Learning models are able to understand when a program is a malware or not doing a binary classification. These Machine Learning models are useful also in the process of *reverse engineering* because in an unknown software it is often useful to find specific functions.

3 Evaluation methods

In order to evaluate a solution based on a features list, it's necessary to train some models and compare the performances according to some metrics. The comparisons in this report are based on the results obtained from three models: **Decision Tree**, **Support Vector Machine** and **Logistic Regression**. We used different metrics to evaluate the performances of these models, like plotting the *confusion matrices* and comparing *precision*, *recall* and *f1-score*. Another metric that we took in consideration was the way of splitting the dataset; in fact the standard way is to reserve $\frac{2}{3}$ for the training and $\frac{1}{3}$ for the test. Decreasing the first one and therefore increasing the second one could get worse performances, but if we still get acceptable results it probably means that the chosen features are meaningful.

4 Dataset, preprocessing and features extraction

The initial dataset had **14397** assembly functions containing *duplicates*. Then we removed all the duplicates and we got **6073** functions. As we can see in Figure 1, removing the duplicates creates an unbalanced dataset and this can lead to problems such as predicting the minority class.

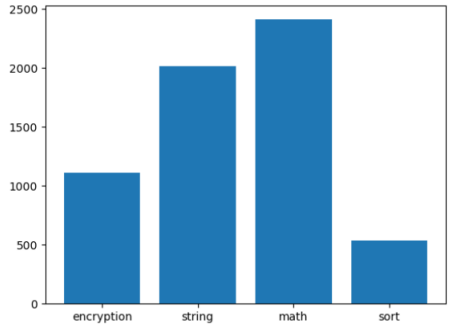


Figure 1: Dataset without duplicates

Each entry of the dataset, provided as a JSON, represent an assembly function and contains:

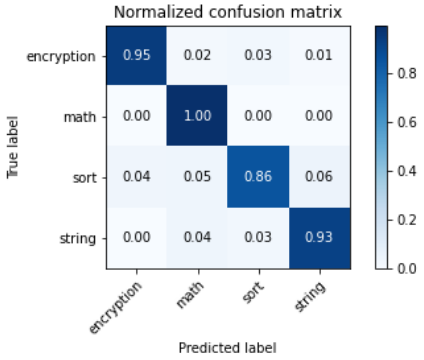
- the label of the function, that is one of the target classes;
- the linear list of his assembly instructions;
- the control flow graph (CFG), encoded as a NetworkX graph.

Every entry in the dataset has been processed and mapped in a list with these following features:

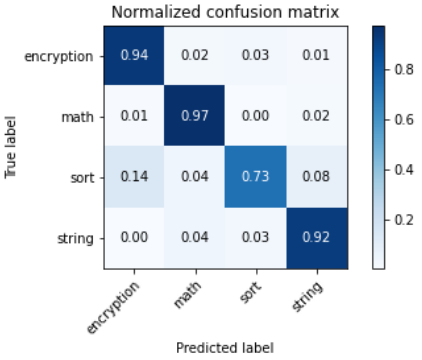
- *graph_nodes*: it is the number of nodes in the CFG;
- *graph_diam*: it is the maximum diameter value of the CFG's strongly connected components;
- *mov*: it is the number of mov instructions in the function;
- *bitwise*: it is the number of bitwise instructions in the function;
- *xmm*: it is the number of xmm registers in the function;
- *arithm*: it is the number of arithmetic instructions in the function;
- *cmp*: it is the number of cmp instructions in the function;
- *shift*: it is the number of shift instructions in the function;
- *calls*: it is the number of calls instructions in the function;
- *cyclomatic complexity*: the cyclomatic complexity of the CFG.

5 Results and conclusions

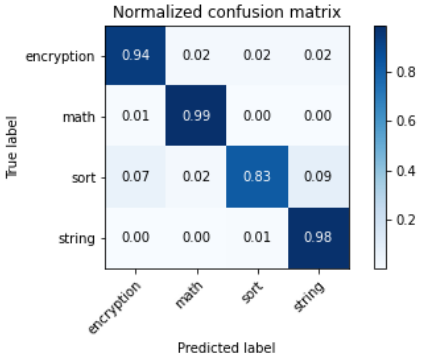
The *Decision Tree* model obtains the best results compared to *SVM* and *Logistic Regression*. The bigger issue was the misclassification of *sort* functions because in the dataset we have a small number of these functions. Incrementing the number of *sort* instances in dataset without adding duplicates would certainly improve the performances for these cases, but produce this new data could require a lot of effort.



(a) SVM with 33% test



(b) Logistic Regression with 33% test



(c) Decision Tree with 33% test

In these images there are the results that we obtain with each method with a test size equal to the 33% of the dataset. Here we have the tables with the different metrics we obtained for each method.

	precision	recall	f1-score	support
encryption	0.98	0.95	0.96	366
math	0.95	1.00	0.97	816
sort	0.83	0.86	0.84	173
string	0.98	0.93	0.95	650
accuracy			0.95	2005
macro avg	0.93	0.93	0.93	2005
weighted avg	0.95	0.95	0.95	2005

Table 1: Decision Tree metrics.

	precision	recall	f1-score	support
encryption	0.94	0.94	0.94	366
math	0.98	0.99	0.99	816
sort	0.92	0.83	0.87	173
string	0.96	0.98	0.97	650
accuracy			0.97	2005
macro avg	0.95	0.94	0.94	2005
weighted avg	0.96	0.97	0.96	2005

Table 2: SVM metrics.

	precision	recall	f1-score	support
encryption	0.91	0.94	0.92	366
math	0.95	0.99	0.99	816
sort	0.81	0.73	0.77	173
string	0.95	0.92	0.94	650
accuracy			0.93	2005
macro avg	0.90	0.89	0.90	2005
weighted avg	0.93	0.93	0.93	2005

Table 3: Logistic regression metrics.