Machine Learning and Security Research: models for classification and prediction of binary functions

Matteo Facci
1597454

October 6, 2022

1 Introduction

This report provides a solution for the function classification problem described in the seminar "Machine Learning and Security Research", using a specific data set.
Starting from the manipulation of the latter, in order to prepare the environment where the program will be executed, new methods for training a classification model and predicting new results will be shown and compared.

2 Programming Language

The whole program is developed in python in order to make use of Scikit Learn Library, a useful library containing tools for Machine Learning including classification models such as Support Vector Classification, Random Forest, Naive Bayes and Logistic Regression.
Colab is used as a development environment. It allows to write and execute arbitrary python code through the browser, and it is especially well suited to machine learning and data analysis. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

3 Workflow model and features for data processing

During reversing of an unknown software it is often useful to find specific functions. In the present case, as mentioned above, a specific data set without duplicates is selected for the analysis. It contains 6073 binary functions belonging to one of four classes defined by the labels *Encryption*, *String*, *Sort*, *Math*. Each item is a dictionary $\{key1 : value1, key2 : value2, ...\}$. The keys are:

- ID : unique id of each function
- $semantic$: the label of each function in $\{math, sort, encryption, string\}$
- $lista\_asm$: the linear list of assembly instruction of each function
- cfg : the control flow graph encoded as a $networkx$ graph

	id	semantic	lista_asm	cfg
0	828	string	['jmp qword ptr [rip + 0x220882]', 'jmp qword ...	{'directed': True, 'graph': [], 'nodes': [{'id...
1	11786	math	['ucomisd xmm2, xmm2', 'jp 0x40', 'ucomisd xmm...	{'directed': True, 'graph': [], 'nodes': [{'id...
2	12621	encryption	['push rbx', 'mov r8d, ecx', 'mov qword ptr [r...	{'directed': True, 'graph': [], 'nodes': [{'id...
3	11166	math	['mov qword ptr [rsp - 0x10], rbx', 'mov qword...	{'directed': True, 'graph': [], 'nodes': [{'id...
4	10432	sort	['jmp qword ptr [rip + 0x200ba2]', 'jmp qword ...	{'directed': True, 'graph': [], 'nodes': [{'id...
5	4176	encryption	['push rbp', 'push r15', 'push r14', 'push r13...	{'directed': True, 'graph': [], 'nodes': [{'id...
6	8323	encryption	['push rbp', 'push r15', 'push r14', 'push r12...	{'directed': True, 'graph': [], 'nodes': [{'id...
7	14389	encryption	['push rbp', 'push rbx', 'mov r10, rdx', 'movz...	{'directed': True, 'graph': [], 'nodes': [{'id...
8	6501	sort	['push rbp', 'push r15', 'push r14', 'push r13...	{'directed': True, 'graph': [], 'nodes': [{'id...
9	1141	string	['jmp qword ptr [rip + 0x200b92]', 'jmp qword ...	{'directed': True, 'graph': [], 'nodes': [{'id...

Figure 1: First 10 items of the original data set.

A workflow model for data processing is established to compare Naive Bayes, Random Forest, Support Vector Machines, and Logistic Regression classifiers.

3.1 Data extraction

The main goal of the first part is to select only the required and related data fields to process the data and optimize memory usage. Only *semantic* (for the labels) and *lista_asm* (for the content) fields are taken from the input data set.

3.2 Pre-processing and data filtering

Pre-processing is useful to prepare the field *lista_asm* for the extraction of the features.
This stage is carried out by removing (filtering) irrelevant characters: simple text cleaning steps are done, such as removing white spaces and numbers, removing symbols and punctuations, converting text to lowercase, etc.
The aim is to have only relevant data for the identification of the features. This can be done by following the same procedures and logic of the word tokenization (to split a sentence into words) implemented for instance with the NLTK (Natural Language Toolkit Project) library.
In the present case, instead of finding and removing the so-called *stopwords* of the library, all the words that do not contain particular subwords are removed.
These subwords are contained in a list *characters of interest* defined as:

```
list = ['not','and','or','xor','sal','sar','shr','shl','cmp','test','mov','xmm','ucomis']
```

The latter recalls all the assembly commands which permit the identification of the functions described by the four aforementioned classes (*Encryption*, *String*, *Sort*, *Math*).
Analyzing in more detail, *Encryption* functions use many shifts and bitwise operations, *String* manipulation functions make many comparisons and swap of memory locations, *Math* functions use mainly floating-point instructions and special registers $xmm0...15$, *Sort* functions use compare and moves operations.

In the end, after the filtering step, a new column *label_id* with encoded categories is added, therefore each unique integer from 0 to 3 will identify a precise label.

	content	label	label_id
0	qword qword qword mov mov dword cmp dword mov ...	string	0
1	ucomisd xmm2 xmm2 ucomisd xmm0 xmm1 pxor xmm3 ...	math	1
2	mov mov qword mov qword xor word mov dword sar...	encryption	2
3	mov qword mov qword movsd qword xmm0 mov mov m...	math	1
4	qword qword qword mov xor mov xor mov cmp dwor...	sort	3
5	mov qword mov qword mov qword xor qword xor qw...	encryption	2
6	xorps xmm0 xmm0 movaps xmmword xmm0 movaps xmm...	encryption	2
7	mov movzx shl movzx shl or movzx shl or movzx ...	encryption	2
8	mov cmp movsxd mov mov qword mov qword mov xor...	sort	3
9	qword qword qword qword mov mov mov mov qword ...	string	0

Figure 2: First 10 items of the filtered data set.

3.3 Bags of words and feature extraction

The n-gram method as a sequence of words (instructions) is applied to construct bags of words and to transform each data into vectors.
It is a process to split the instructions into words and group them using a combination of n-grams. Unigrams, bigrams and trigrams are created from the *content* parts which have passed the previous stages of pre-processing. Basically, with this approach, one creates the input to the classifier with all features (tokenized terms, and term groups), and the classifier builds the model which assigns the class as accurately as possible.
These words are imported to a specially created hashing term-frequency vectorizer that counts the frequency in the set and assigns a unique numerical value for the next classification stage, as well as the weights needed for each word. In other words, a term-frequency vectorizer is identifying how important a word is to a function in the *content*, i.e. the key as the word and the value as the number of frequency in the given set.
$TfidfVectorizer$ class is a standard tool for words classification and TF-IDF stands for Term Frequency-Inverse Document Frequency. The vectorizer can be initialized with the following parameters:

- $min\_df = 5$: remove the words from the vocabulary which have less than 5 occurrencies.
- $sublinear\_tf = True$: scale the term frequency in logarithmic scale.
- $ngram_range = (1, 3)$: to indicate that unigrams, bigrams and trigrams will be considered.

Therefore, the feature vector transforms words in to the numerical value represented in the integer format, i.e. the numerical value to the given word and second - the value of frequency of the word.

```
==>  encryption :
  * Most Correlated Unigrams are: shr, ror, shl
  * Most Correlated Bigrams are: dword shr, movzx xor, xor dword
  * Most Correlated Trigrams are: xor dword xor, dword xor dword, xor dword mov

==>  math :
  * Most Correlated Unigrams are: xmm1, xmm0, movsd
  * Most Correlated Bigrams are: xmm0 qword, qword xmm0, movsd qword
  * Most Correlated Trigrams are: movsd qword xmm1, movsd xmm0 qword, movsd qword xmm0

==>  sort :
  * Most Correlated Unigrams are: movsd, ucomiss, word
  * Most Correlated Bigrams are: qword cmp, word mov, qword word
  * Most Correlated Trigrams are: cmp qword test, mov qword word, mov qword qword

==>  string :
  * Most Correlated Unigrams are: xmm0, movsxd, movsd
  * Most Correlated Bigrams are: test cmp, dword test, test mov
  * Most Correlated Trigrams are: mov dword test, test mov dword, mov qword test
```

Figure 3: Three most correlated instructions for each category.

3.4 Classification

A data splitting task is done with the help of the *train_test_split* function from *sklearn*.
The composition of the data set for training and testing is distributed like this: 75% for training and 25% for testing.
It must be noticed that the data set used for solving the problem is a largely imbalanced data set, in fact, there is not an equal number of items per class.
This aspect could cause problems in the prediction phase especially for the categories with fewer data available for training and testing. This involves additional considerations to be made.
False positives could be much more than false negatives. Variance in the minority set (*sort*) could be larger due to fewer data. The majority class (*math*) could dominate predictions.
Given the prevalence of the majority class, the algorithm will likely regress to a prediction of the latter. In this way, it can maximize its *Accuracy* by arbitrarily predicting that the majority class occurs every time. This result provides close to zero predictive power.
The *Accuracy*, a metric for evaluating the performance of a classifier, might not be appropriate when the data is imbalanced.

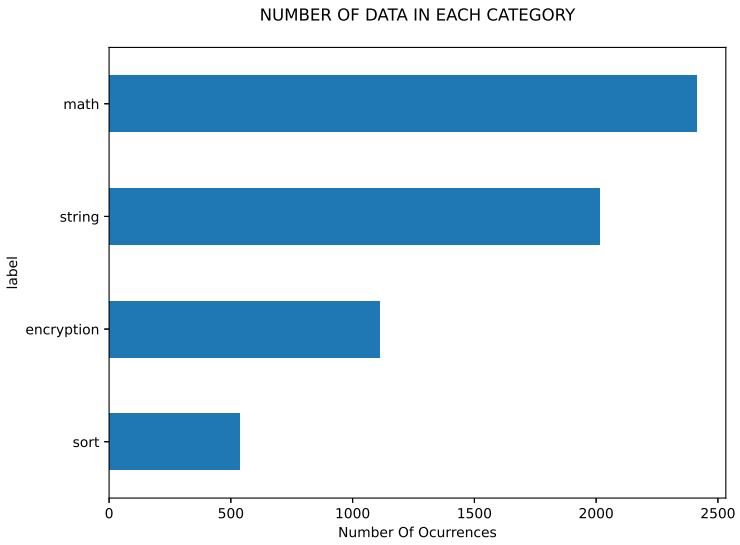

Figure 4: Each category does not contain the same amount of data.

A popular method for dealing with class imbalance is changing the performance metric. Metrics that can provide better insight include:

- $Confusion\,Matrix$: a table showing correct predictions (the diagonal) and types of incorrect predictions.
- $Precision$: the number of true positives divided by all positive predictions. It is a measure of a classifier's exactness. A low value denotes a high number of false positives.
- $Recall$: the number of true positives divided by the number of positive values in the test data. It is a measure of a classifier's completeness. A low value denotes a high number of false negatives.
- $F1\,Score$: the weighted average of precision and recall.

Generally, $Confusion\,Matrix$ and $F1\,Score$ are sufficient in order to evaluate the best model for solving the classification problem with an imbalanced data set.
As a starting point, different standard machine learning classifiers are used, and according to their performance with the data set, the best model among those tested is selected for the solution of the problem.
The classifiers chosen are:

- Random Forest Classifier
- Linear Support Vector Machine
- Multinomial Naive Bayes
- Logistic Regression

To get the performance of each model, a 5 fold cross-validation is made. Cross-validation is a resampling procedure used to evaluate machine learning on a limited data sample.
The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. Cross-validation is primarily used to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.
It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.
$k = 5$ usually yield test error rate estimates that suffer neither from excessively high bias nor from very high variance.
The cross-validation results are included in the following table where the models are sorted in descending order of performance.

Model Name	Mean Accuracy	Standard Deviation
LinearSVC	0.987818	0.015423
LogisticRegression	0.980079	0.018024
MultinomialNB	0.906967	0.015786
RandomForestClassifier	0.888032	0.017320

4 Evaluation

In this section are reported all the results obtained in building the four models and evaluating them with the test data.
In particular for each model are reported the training time in seconds, the values of the performance metrics, and the respective confusion matrices.

4.1 Linear Support Vector Machine

$Training\,time = 0.22133231163024902\,s$

Metric	Value
$Accuracy$	0.990125082290981
$Precision$	0.9857824337942886
$Recall$	0.9834662935524778
$F1\,Score$	0.9845973822168206

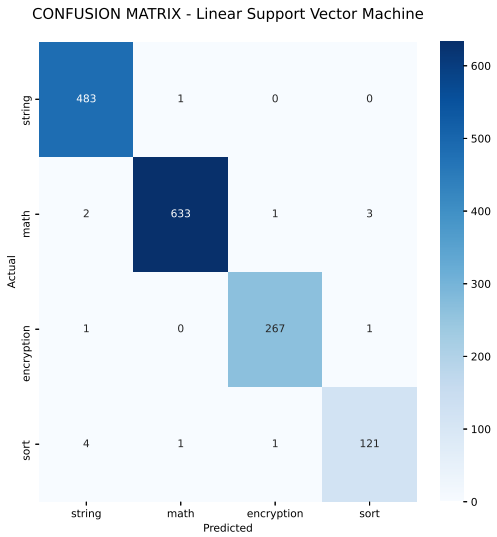

Figure 5: $Confusion\,Matrix$ for Linear Support Vector Machine.

4.2 Logistic Regression

$Training\,time = 7.9367969036102295\,s$

Metric	Value
$Accuracy$	0.978275181040158
$Precision$	0.9649008615091115
$Recall$	0.9593245150447627
$F1\,Score$	0.9619698914604277

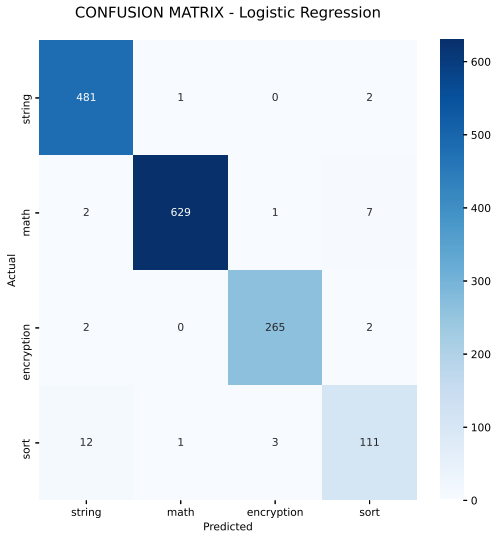

Figure 6: $Confusion\,Matrix$ for Logistic Regression.

4.3 Multinomial Naive Bayes

$Training\,time = 0.11235594749450684\,s$

Metric	Value
Accuracy	0.9078341013824884
Precision	0.886528253109682
Recall	0.7790894278669656
F1 Score	0.789238665636196

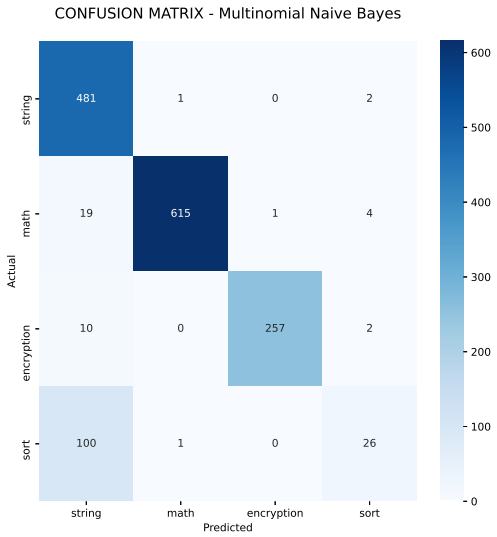

Figure 7: *Confusion Matrix* for Multinomial Naive Bayes.

4.4 Random Forest Classifier

$Training\,time = 3.3990097045898438\,s$

Metric	Value
Accuracy	0.9881500987491771
Precision	0.9812527599618346
Recall	0.9800518926601103
F1 Score	0.980631185215694

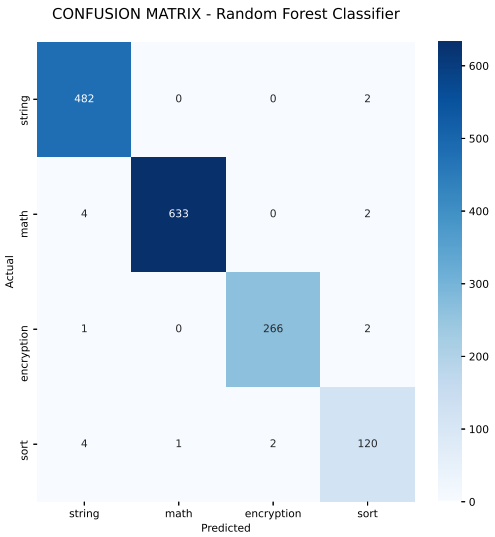

Figure 8: *Confusion Matrix* for Random Forest Classifier.

5 Conclusions

Concerning the considerations made about the classification of an imbalanced data set, all the values aimed at decreeing the best model for the present classification problem were collected.

Looking at the confusion matrices and at the $F1\,Score$ of each model, it is evident that Linear Support Vector Machine and Random Forest Classifier have almost the same performance in terms of correct predictions, since their scores are of about 0.984 for the first and 0.980 for the second.

Immediately behind is the Logistic Regression classifier, with a score of about 0.961 and more wrong predictions especially in the *sort* category, the one with the least amount of data available.

Insufficient performance with respect to the other models on the part of the Multinomial Naive Bayes classifier, which with a score of about 0.789, especially for the categories with fewer data available, is clearly the model that would lead to more prediction errors.

Therefore the choice for the best model in order to solve this classification problem is between Linear Support Vector Machine and Random Forest Classifier.

Since the first has slightly higher values also on the other metrics and has also a reduced $Training\,time$ (0.221 seconds vs. 3.399 seconds), it is definitely the best choice among the four tested models.

Therefore, with the Linear Support Vector Machine model, a solution for the multiclass classification problem is successfully built.