First Homework Machine Learning

Edoardo Di Paolo 1728334

November 2020

Contents

1 Introduction

The main purpose of this homework is to apply the machine learning techniques to the problem given. In this homework the problem is to classify a set of given functions in different classes: *math, encryption, sort and string.*

2 Problem and data

The problem given is a supervised learning task. As I wrote in the introduction, we have to classify different assembly functions in different classes: *math, encryption, sort and string.* We have 4 different datasets: two datasets have the function's type (for the learning), and the other two have only the function without the class. We have two types of dataset: one with duplicates and the other one without.

3 Data preprocessing

The first step is data preprocessing. First of all I converted the dataset from *json lines* to *json*. Initially I thought that the problem was like a *text classification* problem, so I did some test similar to the *spam classification* exercise we did in class. From this tests I didn't get great results, so I changed the approach. We have some hints to classify correctly the functions:

1. *encryption*: these functions have a lot nested for and if (so a complex CFG, that is given in the dataset), they use a lot of xor, shifts and bitwise operation and they are very very long;
2. *sorting*: these functions have one or two nested loops and use some helper functions. They use also compare and moves operations;
3. *math*: these functions use a lot (obviously) of math commands and they use also xmm* registers;
4. *string*: these functions has a lot of swap of memory locations and comparison.

In the next sections I'll write about the evaluation methods and techniques.

4 Evaluation methods

I used, for the test, four different methods: the *decision trees*, the *logistic regression gaussian naive bayes* and *SVC*. I used the *sklearn* library for this homework.

Gaussian Naive Bayes The Naive Bayes methods are a set of supervised learning algorithms, where we assume that there is conditional independence between every pair of features given. In particular the

Guassian Naive Bayes is given by this formula:
$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2})$ where the σ_y and μ_y are estimated using maximum likelihood.

Decision trees Decision Trees are a non-parametric supervised learning used for classification. They are simple to understand and can be visualised. This method doesn't need special techniques on the data, just a little preparation. A problem is the overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid overfitting problem. For this homework I used the *DecisionTreeClassifier* that is a class capable of performing multi-class classification on a dataset.

SVC This method has some advantages: effective in high dimensional spaces (in some tests I had large spaces) and we can define some different kernel functions. I used SVC method.

Logistic Regression This is another method that I used for this homework. I used the LogisticRegresion method to fit the dataset. This is a machine learning classification algorithm that is used to predict the probability of a categorical dependent variable.

5 Results

I used different metrics to check my results. They are *precision, recall, f1-score and the accuracy.* I used also the *confusion matrix.*

I need to introduce some notation before explain the metrics:

1. *True Positive (TP)*: the actual class of the data point was positive and the predicted is also positive;

2. *True Negative (TN)*: the actual class of the data point was negative and the predicted is also negative;
3. *False Positive (FP)*: the actual class of the data point was negative and the predicted is positive;
4. *False Negative (FN)*: the actual class of the data point was positive and the predicted is negative.

The different metrics are defined as follow:

1. *Accuracy*: it is the number of correct predictions made by the model over all kinds predictions made.

$$Accuracy = \frac{TP + TN}{TP + FP + FN - TN}$$

2. *Precision*: it is the proportion of positive predicted that are actually positive.

$$Precision = \frac{TP}{TP + FP}$$

3. *Recall*: it is the proportion of actual positive predicted as positive

$$Recall = \frac{TP}{TP + FN}$$

4. *F1-score*: it is the harmonic mean between precision and recall

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall}$$

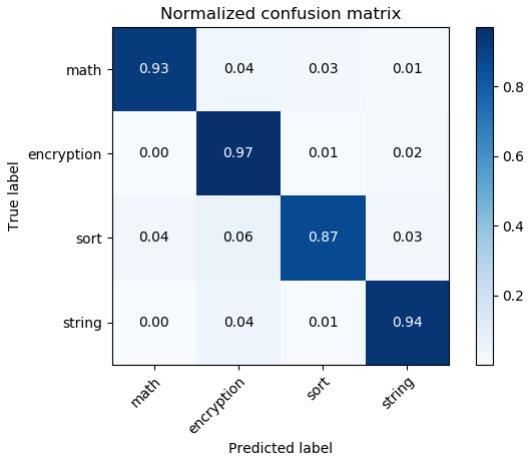
In the attributes list I count different things: all the x64 instructions and also the cyclomatic complexity of the CFG with the edges count and vertices count.

```
nx_graph = json_graph.adjacency_graph(func['cfg'])
css = nx.components.number_strongly_connected_components(nx_graph)
complexity = len(nx_graph.edges()) - len(nx_graph.nodes()) - css
```

Listing 1: Cyclomatic complexity count

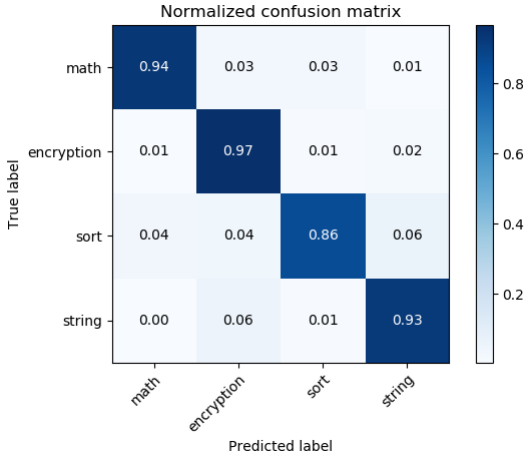
In this way I got a better accuracy on the non duplicate dataset. In fact with the duplicate dataset I got always an accuracy like 0.99 and it was like overfitting.

Figure 1: SVC Results no-dup dataset.



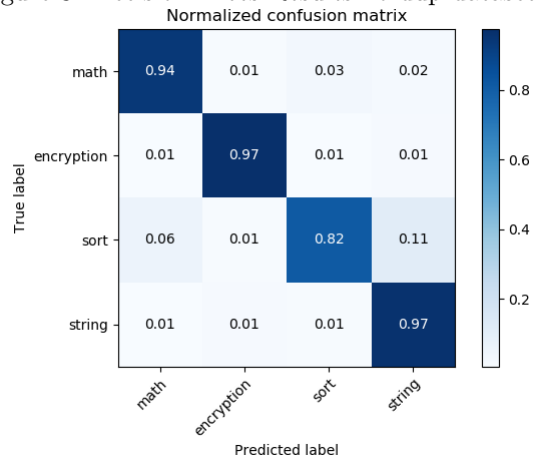
These are the results with SVC model. As we can see, the sort function is the less predictable. This happens because in the no-dup dataset we have less sort function.

Figure 2: Regression Results no-dup dataset.



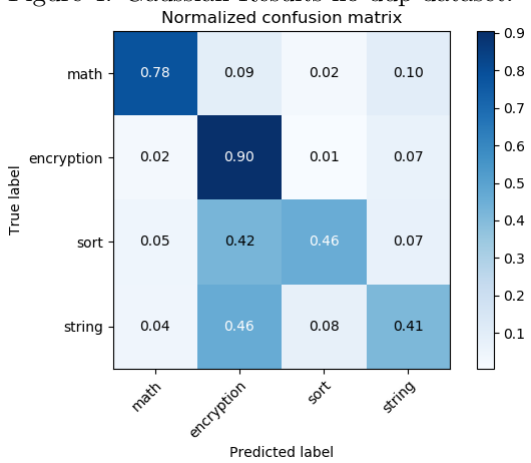
These are the results with the Linear Regression.

Figure 3: Decision Trees Results no-dup dataset.



These are the results with the Decision Trees classifier. As we can see it is less accurate about the sort functions.

Figure 4: Gaussian Results no-dup dataset.

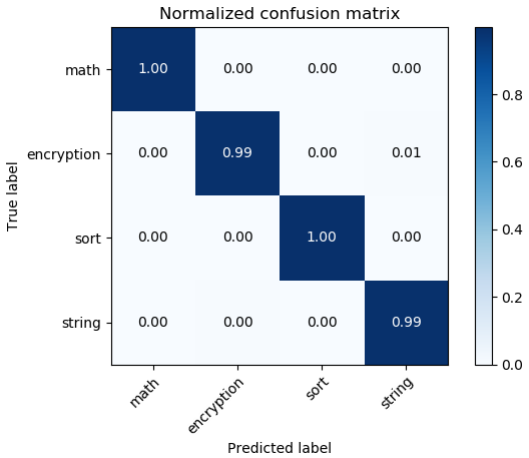


These are the results with the Gaussian Naive Bayes. As we can see it doesn't predict very well except for the encryption function (they are very long, and with many loop!) and the math functions that use xmm* registers that I count.

About the overfitting problem on the duplicate set, we can see

the big difference in this image:

Figure 5: Decision Trees Results dup dataset.



As we can see it seems predict all with 1.00 of Accuracy, that is almost impossible.

The test size is set to 0.33% of the dataset that we have. However, I tried also other data's configuration. For example I passed only some categories (without the graph data) like:

```
["movement", "binary", "comparison", "call", "shift", "unary"]
```

Listing 2: Attributes examples

these are defined in the instructionsx64 dictionary.

Here there are some results.

Figure 6: Regression Results no-dup dataset and no graph data.

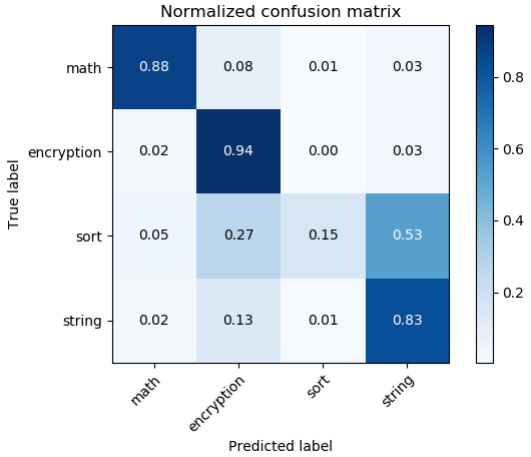
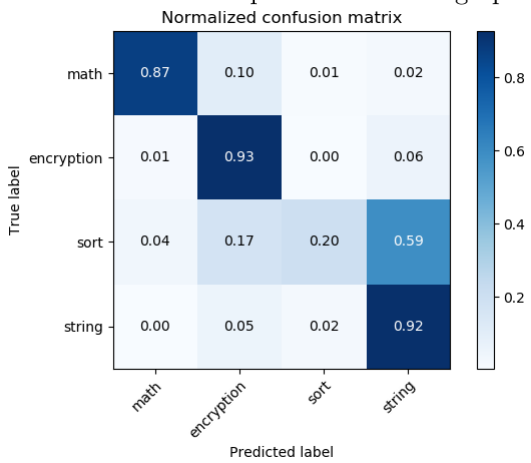


Figure 7: SVC Results no-dup dataset and no graph data.



It is really interesting. As we can see the sort functions are not predicted, as expected. It is expected because in the dataset without duplicates we have less sort functions.

About the blindtest I used Decision Trees classifier to fit the data. When the code is executed in the console there is a print about the accuracy, f1-score, recall and the precision and the *K-Cross Validation*.