Leandro Maglianella 1792507

Machine Learning: Homework 1.

Machine Learning and Security Research:

Function classification

1. Abstract

In this homework, we are provided with a dataset and a blind test: the goal is to be able to train a machine-learning model using the set of samples contained in the dataset and to classify, through this model, the instances contained in the blind test. In particular, each sample represents an assembly code running on a computer and it can be classified into one of four classes of algorithms: “encryption”, “math”, “sort” and “string”.

Being able to achieve an accurate classification of this kind represents an important basis in the field of cyber security to recognize, for example, a malware encrypting personal data stored on a PC.

2. Pre-processing

Two datasets were provided: the first one containing duplicate samples and the second one without duplicates. I chose to use the second one (no duplicates) to train my model. I made this choice assuming that the duplicate samples were chosen randomly and not as an oversampling of some data category. Based on this assumption, if I had used the dataset with duplicates, then the generated model would have been subject to a strong and harmful overfitting towards the duplicated samples.

The dataset is given in the form of a file with the ".json" extension, in which each line contains a sample with its "id", its class ("semantic"), the sequence of assembly instructions running in the computer ("lista_asm") and the control flow graph ("cfg") related to the execution of these instructions. For instance:

{"id": "828", "semantic": "string", "lista_asm": "['$instr_1$', '$instr_2$', …, '$instr_n$']", "cfg": {…*long structure*}}

The first phase to carry out is the pre-processing of the data in order to extract useful features, used to train the model and, in general, to organize the homework. The other goal of this phase is generating a dataset in a normal form, like $D = \{(x_n, t_n)^N_{n=1}\}$. I therefore saved the dataset on my Google Drive and loaded it on Google Colab, where I then continued to carry out the homework.

The lines of the json file are read one at a time and 10 features are extracted from each: 4 related to the control flow graph and 6 to the sample structure. I worked on the graph with the Python NetworkX package and the 4 features extracted are the number of nodes belonging to the graph, the diameter of the graph, the number of simple cycles in the graph and its cyclomatic complexity.

Regarding the remaining features, I have divided the assembly instructions in 6 categories:

A: instructions that move data between memory and registers.

B: arithmetic.

C: floating point instructions.

D: bitwise instructions.

E: calls.

F: jumps.

```
groupA = ['mov', 'push', 'pop', 'lea']
groupB = ['inc', 'dec', 'neg', 'add', 'sub', 'imul', 'idiv', 'sar', 'shl', 'shr']
groupC = ['xmm']
groupD = ['or', 'not', 'and']
groupE = ['call', 'leave', 'ret']
groupF = ['jmp', 'je', 'jn', 'js', 'jg', 'jl', 'ja', 'jb']
```

After that, I manipulated the lista_asm string by removing the special characters (“[”, “]”, “,”, “’”) and by splitting it on the space characters. At this point, I created six counters, one for each group, that were incremented every time a command of the respective group was invoked in the list.

Finally, I created a “pandas.Dataframe” in which I stored the id, semantic and the 10 features for each sample. The result of this phase is a data structure of this kind:

	Id	Semantic	contA	contB	...	Nodes num	Diameter	Cycles num	Complexity
0	828	string	54	5	...	23	9	8	53
1	11786	math	10	3	...	15	6	6	35
2	12621	encryption	215	154	...	18	7	16	33
3	11166	math	7	1	...	2	1	0	3
4	10432	sort	25	7	...	22	6	11	45
...	...	...	...	...	...	...	...	...	...
6068	12484	math	17	4	...	8	4	3	18
6069	7809	sort	52	21	...	22	6	13	18
6070	9806	encryption	282	95	...	113	11	87	115
6071	421	math	29	2	...	11	6	2	23
6072	2008	sort	39	22	...	10	4	5	18

[6073 rows x 12 columns]

3. Data splitting

The pre-processed dataset was first of all divided into two dataframes: X (containing the 10 features of each sample, I simply dropped the "Id" column as it was useless for training) and y (containing the class of each sample). Both were in turn divided into two parts obtaining X_train, X_test, y_train and y_test (respectively, Figure 1, 2, 3 and 4): this division was done with a proportion of 2/3 for the train datasets and 1/3 for the test datasets. Using this split, 4050 out of the 6073 samples were used for training and 2023 for testing.

	contA	contB	contC	contD	...	Nodes num	Diameter	Cycles num	Complexity
4087	46	10	0	16	...	25	8	13	61
4932	28	7	0	4	...	12	6	5	16
5904	10	1	2	8	...	2	1	0	3
1301	2	1	12	4	...	5	3	1	10
4832	11	1	0	2	...	15	8	0	28
...	...	...	...	...	...	...	...	...	...
905	50	4	0	16	...	15	4	12	41
5192	38	9	3	35	...	19	8	4	37
3980	44	4	109	78	...	37	8	16	89
235	5	1	3	3	...	2	1	0	3
5157	47	69	0	52	...	10	4	7	26

[4050 rows x 10 columns]

Figure 1

4087	string
4932	sort
5904	math
1301	math
4832	string
	...
905	string
5192	string
3980	math
235	math
5157	encryption

Name: Semantic, Length: 4050,

Figure 3

	contA	contB	contC	contD	...	Nodes num	Diameter	Cycles num	Complexity
997	84	7	0	51	...	21	5	9	34
3829	16	2	20	20	...	8	4	2	17
4268	38	14	9	27	...	14	7	4	31
5968	83	13	0	31	...	47	12	24	117
3708	79	28	0	28	...	28	7	15	22
...	...	...	...	...	...	...	...	...	...
3761	47	2	59	57	...	27	8	6	59
522	34	5	0	25	...	13	5	7	28
1020	48	15	0	17	...	12	4	9	12
1545	19	1	10	15	...	2	1	0	3
4517	9	2	0	6	...	15	7	6	35

[2023 rows x 10 columns]

Figure 2

997	encryption
3829	math
4268	math
5968	string
3708	sort
	...
3761	math
522	string
1020	sort
1545	math
4517	string

Name: Semantic, Length: 2023,

Figure 4

I tested other subdivision proportions (using a decision tree model to evaluate them) and increasing the test dataset over 1/3, the accuracy slightly decreases, albeit always remaining above 90%.

4. Model creation and fitting

Various models based on different concepts have been created and trained on X_train and y_train, here I am reporting only the two that have achieved the greatest results: classifiers based on Decision Trees and Support Vector Machines (with a linear kernel and C = 1). Both models were used without specifying any additional parameters.

5. Classification evaluation

Decision tree Model:

The model has been trained in 0.595s.

Testing it on X_test and y_test gives the following performance metrics:

	precision	recall	f1-score	support
encryption	0.95	0.94	0.95	370
math	0.99	0.99	0.99	831
sort	0.87	0.88	0.87	185
string	0.97	0.98	0.97	637
accuracy			0.97	2023
macro avg	0.95	0.95	0.95	2023
weighted avg	0.97	0.97	0.97	2023

In addition, the following confusion matrix is computed:

352	5	6	7
4	819	3	5
11	1	162	11
2	1	10	624

Support Vector Machines Model:

The model has been trained in 4.902s.

Testing it on X_test and y_test gives the following performance metrics:

	precision	recall	f1-score	support
encryption	0.97	0.95	0.96	370
math	0.97	0.99	0.98	831
sort	0.87	0.83	0.85	185
string	0.96	0.96	0.96	637
accuracy			0.96	2023
macro avg	0.94	0.93	0.94	2023
weighted avg	0.96	0.96	0.96	2023

In addition, the following confusion matrix is computed:

350	10	7	3
2	820	5	4
8	5	154	18
1	14	12	610

At this point, I tried to change some of the model's parameters; however this did not lead to an increase in metrics, but on the contrary the training time increased considerably: for this reason I decided to stick to the initial model parameters formulation. I also evaluated the accuracy of both models with a K-fold cross validation algorithm using various K values and the results have practically always been identical to the accuracy values already given above.

Based on these results, I have chosen to use the model based on decision tree for predicting on the blind test because it is able to reach higher value of performance in nearly all the metrics for every feature.

6. Prediction on the blind test

I reapplied the same pre-processing phase to the blind test (which in this case contains duplicates) and got the following *newdataset*:

	Id	contA	contB	contC	...	Nodes num	Diameter	Cycles num	Complexity
0	10351	264	52	34	...	93	9	68	101
1	14513	10	2	2	...	7	5	0	13
2	1767	8	0	0	...	9	5	0	17
3	9384	65	9	0	...	38	11	15	90
4	3018	15	3	26	...	6	4	2	13
..	...	...	...	...	...	...	...	...	...
752	9661	9	5	30	...	10	5	1	20
753	6933	18	10	0	...	13	6	8	21
754	11954	31	18	4	...	5	3	2	11
755	5769	228	32	432	...	110	19	34	253
756	7957	10	3	9	...	6	4	1	12

[757 rows x 11 columns]

Dataframe “new_X” were created by dropping the Id column and I used the trained model to predict the classes of the samples in it contained.

The final results are in “1792507.txt”

Of the 757 blind samples:

- 136 were classified as “encryption”
- 232 were classified as “math”
- 222 were classified as “sort”
- 167 were classified as “string”

7. Conclusion

The homework ends here, not knowing the actual classes of the blind test with which to compare the output of my classifier.

My expectations of the accuracy of the classification of this blind test are quite high, being rather high (97%) in the evaluation of the model: if I had to quantify my prediction, I would foretell that this model could achieve an actual accuracy of at least 80-85%.