Machine Learning
Implement and train a system capable of performing both binary and multi-class classification problem in binary analysis

Vannoli Marco matr.1860363[*]
Master's degree in artificial intelligence and robotics Department of computer engineering, automatic e management "Antonio Ruberti" University of Sapienza Roma
(Dated: February 14, 2020)

1. INTRODUCTION

The field of machine learning meets the world of malware more precisely that of binary analysis with the aim of optimizing processes regarding the analysis and individualization of malware. The tasks with which machine learning helps such processes are:

- Binary Similarity,
- Compiler Provenance,
- Function Naming.

The goal of the project is to classify a series of functions, each of which consists of assembly instructions (binary code) in two different point of view:

- from a binary point of view with purpose to identify whether the function belongs to hight or low optimizer.
- from a multi-class point of view, identifying the source compiler considering that the compiler classes are 3 or icc, clang and gcc.

2. DATASET

The dataset is composed of a series of functions consisting of three parameters:

- instructions composed of assembly instructions
- the value of the optimization which can be High (H) or Low (L)
- the compiler of membership that takes three class values (gcc, icc and clang)

[*] vannolimarco956@gmail.com

The dataset that was provided is formed by two files in format jsonl, one used for the training phase (train_dataset.jsonl) and one intended for the test phase (test_dataset_blind.jsonl). The JSONL file for training follows a structure of this type:

```
{
  "instructions": ["xor edx edx",.,"ret"],
  "opt": "H",
  "compiler␣":" gcc "
}
```

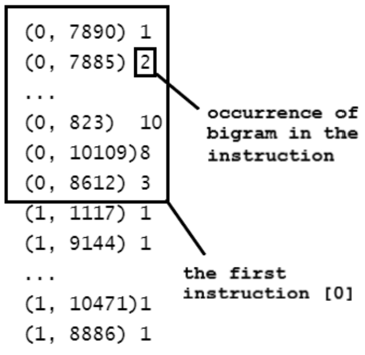
All the instructions inside the "instructions" tag include the input that, as will be explained in the preprocessing paragraph, will be processed to convert it into numbers. The last two "opt" and "compiler" tags are the outputs for our classification models, the first is a binary classification problem [hight, low] or [1.0] and the other a classification problem multi-class [gcc, icc, clang]. Instead, the dataset of test (test_dataset_blind.jsonl) only contains information related to the input in order to obtain a forecast for the outputs relative of classification problems.

3. PREPROCESSING

The preprocessing has been performed through an implementation of a class dedicated to it to expose the necessary methods both to manipulate the data to deliver them to the model and to process them during the prediction phase. Preprocessing plays a fundamental role in optimizing the accuracy of prediction results. For this homework different methods have been used to vectorize the features, each of them tested in order to obtain good accuracy and modeling about classification problems provided. The first method that has been adopted is to consider all of its mnemonics for each instruction and then assign to each of it the index of a vocabulary that has been created specifically for them. Then each instruction has been padded in such a way to have a shape suitable for the model. Thus the input is represented in this form:

[[300, 300, 300, ... , 64, 264, 309 0 0 0 ...max len]
[394 , 47, 185, 319 , 34, 309 0 0 0 ....max len]]

The numbers correspond to the index of mnemonics belong to the vocabulary. The padding influences strongly, as will been reported later, accuracy and the final score because they dirty the features especially if for padding we consider the maximum length that an instruction can have (in this case 22666). Such a method can be identified as a unigram representation with relative vocabulary. During the analysis of the dataset, the huge quantity about mnemonics belong to each instruction has been noted that might bring confusion during the training of the model. In order to avoid this issue, only the n-first and n-last (the best value tested is n=3) mnemonics have been taken focalizing the model just on relevant features. This kind of preprocessing, as will seen later, allow to obtain as accuracy more satisfactory rather than considering all mnemonics inside instruction. The second way of preprocessing adopted is the using of a method that allows to vectorize each instruction considering the bigrams and their counting inside each instruction. This method has been performed by a method given by sklearn library which exposes some method used for the vectorizing. The features going out are shown in the following figure (figure 1):



```
(0, 7890) 1
(0, 7885) 2
...
(0, 823)  10
(0, 10109)8
(0, 8612) 3
(1, 1117) 1
(1, 9144) 1
...
(1, 10471)1
(1, 8886) 1
```

occurrence of bigram in the instruction
the first instruction [0]

FIG. 1. Structure of outcome from CountVectorize method

Thus this method converts a collection of text documents to a matrix of token counts considering the bigram for each instruction. This preprocessing results a good choice because the bigrams allow to mark the position of each mnemonic with its next one and besides the counting of each bigram enable adding important information utils during training. Thus these two methods of preprocessing have been used in order to mark all possible important feature utils to the model in order to obtain a good probabilistic model. However will be shown all results arose by this two kind of preprocessing in the next sections [3]. The targets have been preprocessed by accounting for the task required, binary classification or multi-class classification. In the first case the targets have been considered with binary values, i.e 0 value for Low 'opt' case and 1 value for Hight 'opt' case. In the other case the targets were managed in order to have the 0 value for the 'gcc' compiler, the 1 value for the 'clang' compiler and 2 value for the icc compiler:

	values for targets in output	
	targets	value
optimizer	Hight	1
	Low	0
compiler	gcc	0
	clang	1
	icc	2

TABLE I. values for targets

The fetaures and targets were splitted in such way to obtain a 70% for training set and 30% for testing set:

Training/Test set after count vectorizing	
Set data	shape
X train	(21000,10751) 70%
Y train	(21000,1) 70%
X test	(9000,10751) 30%
Y test	(9000,1) 30%
features	(30000,10751) 100%
targets binary	(30000,1) 100%
targets multi-class	(30000,1) 100%

TABLE II. The Training/Testing set with relative shape

4. CLASSIFICATION ALGORITHMS

Each algorithm has been performed for the two different methods adopted for the extrapolation of features on preprocessing in such a way to have several confronts between all algorithms about different preprocessing methods. As it was done for preprocessing, a class has been built called Model() in order to call whenever the desired classification algorithm needs. The algorithms that were used to training are:

- **Naive Bayes classifiers**: as naive bayes algorithm,**the Multinominal** and **Bernoulli** algorithms have been taken in considering because the second is used when we have discrete data while the first assumes that all our features are binary such that they take only two values i.e Hight = 1 , Low = 0.

- **Support-Vector Machines (SVM)**: is a supervised machine learning algorithm which can be implemented for both classification or regression problems. During training two different models of SVM or SVC which concerns the classification have been used: **KRB svc** and **Linear svc**. Both models implements the same algorithm but with the difference on kernel used. Infact the first use a rbf kerner (based on squared Euclidean distance) while the second a linear kernel (single Line).

- **Decision tree**: it derives to the family of supervised learning algorithms. It aims to create a training model which can utils to predict class or value of target variables by considering a learning decision rules deduced from prior data(training data). A classical decision tree algorithm has been used with max depth equals to 2.

Each algorithms have been confronted among all other others noting that some algorithm performs in a better way rather other due to composing of features extracted. This will be clear when the outcomes will be shown about accuracy and confusion matrix for each algorithm adopted.

5. EVALUATION ACCURACY

The model evaluation phase and therefore the algorithm associated with the various preprocessing methods required the careful analysis of both the accuracy parameter and the parameter f1 obtained from a confusion matrix. This section will be exposed the evaluation from a data distribution on dataset point of view in order to analyze the accuracy for each algorithm. The accuracy has been computed for each algorithm, each of them performed with the different techniques of preprocessing feature. The following table show the computed accuracy during testing phase remembering that the 70% of the training set has been used to train and the 30% of testing set was used to test accuracy for both classification problem:

Accuracy obtained with each algorithms			
Algorithms	feature Prepr.	% Binary	% M-Class
Bernoulli	Unigrams(n=3 *)	61%	33%
	Count bigrams	72%	70%
Multinomial	Unigrams(n=3 *)	63%	44%
	Count bigrams	75%	75%
SVC rbf k.	Unigrams(n=3 *)	**76%**	62%
	Count bigrams	**80%**	65%
SVC linear k.	Unigrams(n=3 *)	66%	34%
	Count bigrams	**84%**	**86%**
DecisionTree	Unigrams(n=3 *)	**73%**	45%
	Count bigrams	72%	52%

TABLE III. The values of computed accuracy

* only the first and last three mnemonics for each instruction are been taken

The table above shows results that must be analyzed both from an algorithm point of view and from a preprocessing point of view:

- From the preprocessing point of view is important to note the huge impact that the bigrams count vectorizing had on final accuracy results. This factor is proven by the fact that the preprocessing carried by Countvectorize with bigrams on features take account of order from for each mnemonic with its one next, counting the recurrence for each bigram in each instruction. On the other hand, it should note that the first method takes into account only the firsts n-mnemonics and this offers a good result rather than taking the entire instruction from the dataset. This gap between the two kinds of preprocessing there will be also relieved in the confusion matrix obtained during evaluation (see figure 3 on page 6).

- From the performance of algorithms point of view, it should mark the behavior between Naive Bayes Multinominial and Naive Bayes Bernoulli because this last care about the frequency with which a word, or as in this case a mnemonic, appears in the instruction while the Bernoulli algorithm take into account if one mnemonic is into instruction or not. Thus, in this case, Multinominial has a good trend than Bernoulli especially if you use a preprocessing based on bigrams and their counting. The key point in this point of view is the performance of the SVM algorithm compared with the Naive Bayes algorithms used. The SVM algorithms perform the accuracy in a better way because they care about the interactions among features rather than the Naive Bayes which considers them independent of each other. Therefore SVM results a good choice for this kind of problem and it has been considered during the entire work [1]. Another important thing is the few difference about accuracy between Linear and RBF SVC kernels (Linear would seem better) and this last made bring us to note the linearity of dataset [2].

So the focus was on the SVM algorithm with linear kernel. During the evaluation a parameter tuning was carried out up to arrive at the best hyperparameters which are shown in the following table:

The best Hyperparameters for Linear SVC algorithm	
Hyperparameters	Values
loss function (loss)	'squared_hinge'
Tolerance for stopping criteria (tol)	0.0001
C of the error term	0.1
multiclass strategy(multi_class)	'ovr'
max number of iterations(max_iter)	2000

TABLE IV. The best hyperparameters of Linear SVC algorithm

In consideration of this case, we note that the metric about accuracy it is not a faithful parameter of the precision and goodness of the model and for this reason, in the next section the metric from the confusion matrix will be shown.

6. EVALUATION CONFUSION MATRIX

The confusion matrix is an important metric to evaluate the performance of the model because it takes account of the balance of dataset. In this case, the displacement is present to dataset relates to binary classification [4]. In order to evaluate the models through confusion matrix the parameters recall, precision and f1 have to consider:

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

- Recall: the ratio of correctly predicted positive observations to the total observations in actual class: Recall = TP/TP+FN

- Precision: the ratio of correctly predicted positive observations to the total predicted positive observations: Precision = TP/TP+FP

- F1: the weighted average of Precision and Recall. It considersboth false positives and false negatives into account. F1 Score = 2*(R * P) / (R + P)

From this concept emerge that having an uneven class distribution bring to having False Positive and False Negative not similar in value and therefore it is preferable to adopt an evaluation on the parameters belonging to the confusion matrix, i.e Recall, Precision and so F1. Returning to the Linear SVC algorithm-based model, the following matrix confusion emerges:

	Binary Classification Confusion Matrix			
	precision	recall	f1-score	support
0 (Low)	0.85	0.89	0.87	5445
1 (Hight)	0.82	0.77	0.79	3555
micro avg	0.84	0.84	0.84	9000
macro avg	0.84	0.83	0.83	9000
weighted avg	0.84	0.84	0.84	9000

TABLE V. binary class. confusion Matrix Linear SVC model

	Multi-class Classification Confusion Matrix			
	precision	recall	f1-score	support
0 (gcc)	0.85	0.85	0.85	2993
1 (clang)	0.88	0.86	0.87	2964
2 (icc)	0.84	0.86	0.85	3043
micro avg	0.86	0.86	0.86	9000
macro avg	0.86	0.86	0.86	9000
weighted avg	0.86	0.86	0.86	9000

TABLE VI. Multi-class confusion Matrix Linear SVC model

The tables 5 and 6 show the good performance of the Linear SVC model on recall, precision, and f1. Looking at the values for recall purchased for the binary classification problem we recognize a difference with those obtained for the problem of multi classes that are closer to each other and this because the dataset with respect to the binary task (opt: Hight or Low) is unbalanced (from a difference of 0,89-0,77 = 0.12 for binary to 0,86-0,85 = 0,1 for multi-task). Thus the F1 value achieved in this way is promising for future predictions. From the figure 2 in page 5 is possible look the confront between the Multinominal NB, SVC rbf kernel and SVC Linear kernel model about confusion matrix: the more you advance with the models, the more the confusion matrix reduces the values beyond the mandatory line, thus increasing the values that are on it. The outcomes are obtained through the preprocessing that takes into account the bigrams and their recurrence in the instruction. It is also important to note the behavior of the same model with the preprocessing of the unigrams without recurrence. The comparison just cited, is shown in Figure 3 on page 6: the results show an incorrect behavior from the sensitivity and precision of the model which of course did not come out of it by estimating accuracy only [4].

7. CONCLUSION

In conclusion about results achieved, the Linear SVC algorithm achieves good results both from an accuracy point of view and from an F1 score point of view thanks above all to preprocessing which has allowed us to join the features in a dependent manner through bigrams. On the other hand, the idea of extrapolating for training only the initial and final mnemonics for a single instruction has led to a reduction in the sparsity of the data, thus increasing the results (accuracy but not f1) even if smaller than those obtained through vectorization with bigrams, as is shown in the figure 3 on page 6 from the point of view confusion matrix. How future work could perform a more precise analysis on binary analysis problems by extrapolating features that would make the data more dependent on each other and therefore easier to interpret for the model.

8. REFERENCE

[1] Zubrinic, Krunoslav,Milicevic, Mario, Zakarija, Ivona International journal of computers 2013/01/01 *Comparison*

of Naive Bayes and SVM Classifiers in Categorization of Concept Maps

[2] Apostolidis-Afentoulis, Vasileios *SVM Classification with Linear and RBF kernels* 10.13140/RG.2.1.3351.4083 2015/07/08

[3] Hunter Heidenreich https://towardsdatascience.com/natural-language-processing-count-vectorization-with-scikit-learn-e7804269bb5e. Aug, 2019.

[4] Josephine S Akosa, Oklahoma State University Paper 942-2017 *Predictive Accuracy: A Misleading Performance Measure for Highly Imbalanced Data*

FIG. 2. **Comparison of the confusion matrix normalized (9000 samples) between algorithms trained with counter vectorizer of bigrams**

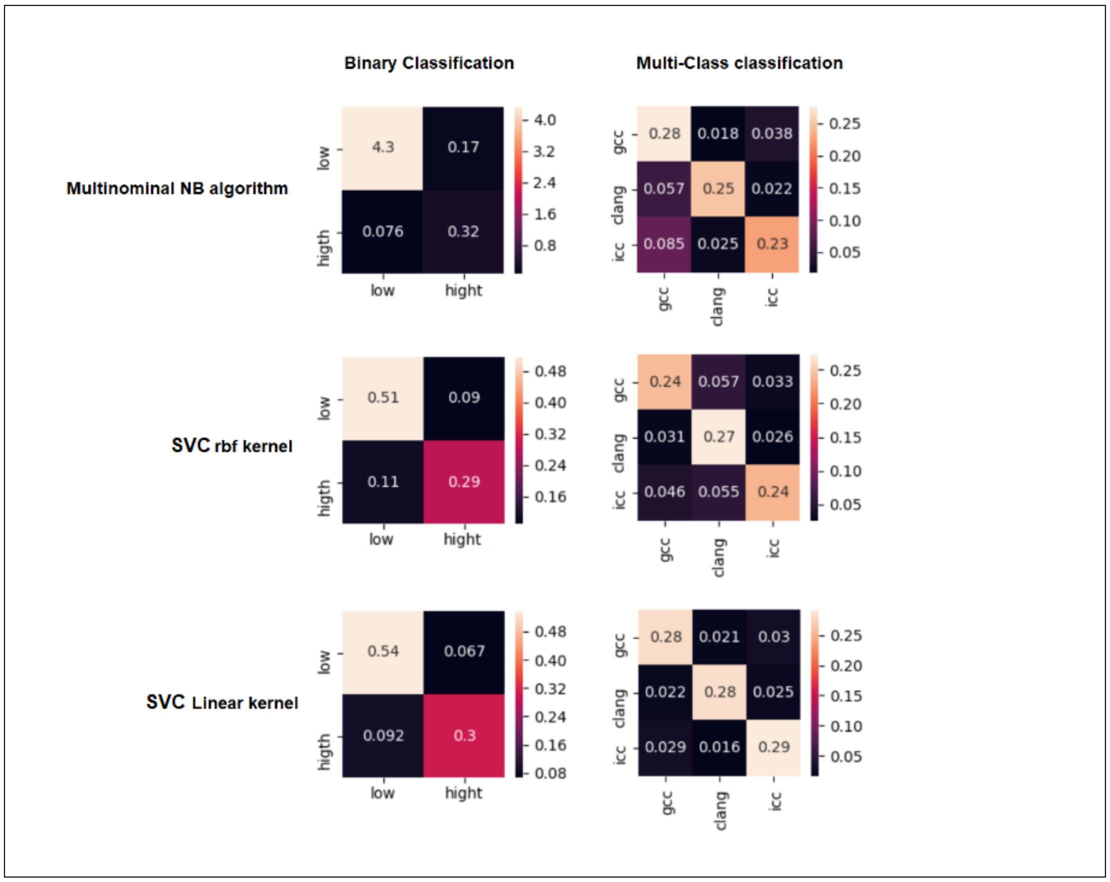
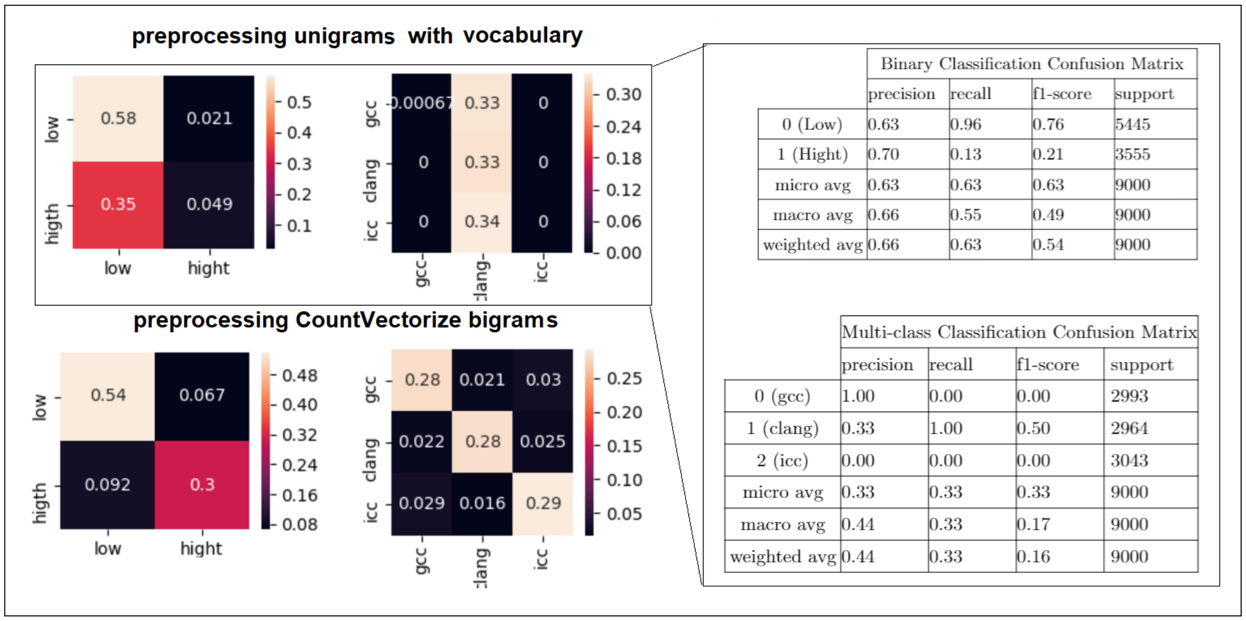


FIG. 3. **Comparison of the confusion matrix normalized (9000 samples) between the two preprocessing methods used both with SVC Linear model**



	Binary Classification Confusion Matrix			
	precision	recall	f1-score	support
0 (Low)	0.63	0.96	0.76	5445
1 (Hight)	0.70	0.13	0.21	3555
micro avg	0.63	0.63	0.63	9000
macro avg	0.66	0.55	0.49	9000
weighted avg	0.66	0.63	0.54	9000

	Multi-class Classification Confusion Matrix			
	precision	recall	f1-score	support
0 (gcc)	1.00	0.00	0.00	2993
1 (clang)	0.33	1.00	0.50	2964
2 (icc)	0.00	0.00	0.00	3043
micro avg	0.33	0.33	0.33	9000
macro avg	0.44	0.33	0.17	9000
weighted avg	0.44	0.33	0.16	9000