Authorship Identification of Source Code Segments Written by Multiple Authors Using Stacking Ensemble Method

Parvez Mahbub
Department of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada
parvezmrobin@dal.ca

Naz Zarreen Oishie
Department of Computer Science
University of Saskatchewan
Saskatoon, Saskatchewan, Canada
naz.oishie@usask.ca

S.M. Rafizul Haque
CSE Discipline
Khulna University
Khulna, Bangladesh
rafizul@cse.ku.ac.bd

***Abstract*—Authorship identification of source code segment identifies the author of a source code segment through supervised learning. It has vast importance in plagiarism detection, digital forensics, and several other law enforcement issues. However, when multiple authors write a source code segment, typical author identification methods no longer work. Here, an author identification technique capable of predicting the authorship of source code segments, even in the case of multiple authors, has been proposed. This proposed technique uses a stacking ensemble classifier built upon several deep neural networks, random forests and support vector machine classifiers. It has been shown that a single classification technique is no longer sufficient for identifying the author group. Using a deep neural network-based stacking ensemble method can significantly enhance the accuracy. The performance of the proposed technique has been compared with some existing methods, which only deal with the source code segments written precisely by a single author. Despite the more challenging task, our proposed technique has achieved promising results evident by the identification accuracy and f1-score, compared to the related works, which only deal with code segments written by a single author. This work is previously published at International Conference on Computer and Information Technology (ICCIT) 2019.**

***Index Terms*—Source Code Authorship Identification, Multiple Author, Deep Neural Network, Random Forest, Support Vector Machine, Stacking Ensemble**

I. Introduction

Author identification of source code segment is an important research topic in the field of software forensics. It has many uses such as plagiarism detection, law enforcement, copyright infringement etc. [8], [12]. Frantzeskou [7] mentioned that source code author identification is useful against cyber attacks in the form of viruses, trojan horses, logic bombs, fraud, credit card cloning, and authorship disputes or proof of authorship in court. There are specific patterns that developers subconsciously reflect in their codes based on their particular coding style while still following the guidelines, standards, rules, and grammars of a language or framework [12]. These pieces of information can be used to identify the author of the source code segment.

In recent years, open-source software development has entered a new era. Many big companies like Google, Microsoft, and many others are maintaining their projects open source. Alongside, small and mid-level projects are being written by a group of authors. In these cases, trivial author identification schemes no longer work. When someone contributes to an open-source project, the writing style of the original author of the source code segment is no longer unique, and it makes the author identification task harder. Even worse case is when several authors equally contribute a project. The writing style is then the aggregation of all the authors. We aimed to solve this problem and proposed an approach to identify the author of a source code even when more than one author contributes it. This paper has proposed an author identification technique using a stacking ensemble method composed of several deep neural networks(DNN), random forests, and support vector machines(SVM).

A. Problem Definition

Authorship identification is the task of having some samples of code for several programmers and determining the likelihood of a new piece of code having been written by each programmer [9]. As the name suggests, authorship identification of source code segments written by multiple authors identifies the author-label when the number of authors of the source code segment is more than one. These contributions can be of two types. The source code segment can be written by mostly one author and has small contributions from several other authors. Another is that a source code segment can be directly written by a group of authors and have a roughly equal contribution from each of them. Both of these happen in open-source software and projects, which are very popular nowadays.

B. Motivation

Authorship identification of source code segment has a vast application area including plagiarism detection, authorship dispute, software forensics, malicious code tracking, criminal prosecution, software intellectual property infringement, corporate litigation, and software maintenance [8], [13], [16], [19]–[21]. In the case of an authorship dispute, authorship

identification can be a solution. Given the source code segment and the candidate owners, the likelihood of each candidate being the author of the source code can be determined [12]. Again, Kothari [12] identified that author identification helps to detect the author of the malicious code. Software companies can also use an authorship identification system to keep track of programs and modules for better maintenance [21]. Though source code segments are much more restrictive and formal than spoken or written language, they inhibit a significant degree of flexibility [7]. According to Shevertalov [18], using differences in the way programmers express their idea, their programming style can be captured. This programming style, in turn, can be used for author identification. Although a large number of works have already been done regarding author identification of source code segment, according to Frantzeskou [8], the future of author identification of source code segment is in collaborative projects to which we aimed at.

The remaining sections are organized as follows. Section II contains a briefing on background topics regarding this work. Section III contains a summary of the related works. In section IV, we discuss our author identification technique for multiple authors. In section V, the experimental results of our proposed technique are analyzed and compared with that of some related works. Finally, in section VI, the conclusion is stated with the possible future direction of this work.

II. Background

A. Ensemble Method

By combining several methods, ensembling method helps to improve the results of machine learning. An ensemble is often more accurate than any of the single classifiers in the ensemble. According to Maclin [14], an ensemble consists of a set of individually trained classifiers whose predictions are combined while classifying instances by the ensemble method. This meta-algorithm combines several machine learning techniques into one predictive model. In our work, we used a stacking ensemble in order to improve our prediction performance.

B. Random Forests

Random forest is an ensemble learning method where each classifier in the ensemble is a decision tree classifier. This collection of classifiers is called a forest. During classification, each decision tree gives its vote, and the result is based on the majority of the votes.

III. Related Works

Numerous works are available on source code segment author identification using a variety of features and classifiers. However, very few of them use machine learning techniques to identify the author of source code segments.

According to Ďuračík [6], there are several approaches to identify the author of the source code segment. The first one is text-based and uses plain text as an input. The second level is token or metric-based.

A. Text Based Approaches

The first approach, which treats source code segment as plain text, is a form of natural language processing. This approach cannot make use of the programmatic structure of the source code segment.

Frantzeskou et al. [8] proposed a technique called Source Code Author Profiles(SCAP) for author identification. They generated a byte-level n-gram author profile and compared it with previously calculated author profiles. Burrows [4] mentioned, the SCAP method truncates the author profiles greater than the maximum profile length causing a bias towards the truncated profiles.

Burrows et al. [3] proposed an approach using information retrieval. They generated n-gram tokens from the source code segments and indexed them in a search engine to query the author of source code and return a ranking list of authors which matched the n-gram token of the source code segment with 67% accuracy.

B. Metric Based Approaches

Frantzeskou [8] pointed out that metric-based author identification is divided into two steps. The first part is extracting the code metrics that represent the author's style. The second part uses those metrics to generate a model capable of labelling a source code segment by corresponding author name. However, a significant amount of time is required to gather all possible metrics and examine to choose only the metrics responsible for differing the authors' style.

Lange and Spiros [13] assumed that the code metrics histogram should vary from author to author as of their coding style. An optimum set was selected using genetic algorithms(GA) and used as input for the nearest neighbour (NN) classifier from several source code metrics. This method achieved 55% accuracy. According to Yang [20], some of the features of this paper are unbounded, for example, the indentation category.

Shevertalov et al. [18] proposed a technique based on GA. The metrics are extracted from the source code segment to make a histogram which is sampled using GA. The author profile is produced using categorized histogram samples. For files, they achieved 54% accuracy, and for projects, they achieved 75% accuracy. Yang [20] mentioned that the details of the final feature set are not mentioned in this paper. So, the feature set is non-reproducible.

Bandara and Wijayarathna [1] used the deep Neural Network for source code segment author identification. The converted source code metrics they used to feed a neural network are identical to that of Lange et al. [13]. Their deep neural network consisted of three restricted Boltzmann machines (RBM) layers and one output layer. They achieved 93% accuracy.

Zhang et al. [21] used SVM to identify the author of the source code segment. They categorized their feature into four groups, namely – programming layout feature, programming style feature, programming structure feature and programming logic feature. They used sequential minimal optimiza-

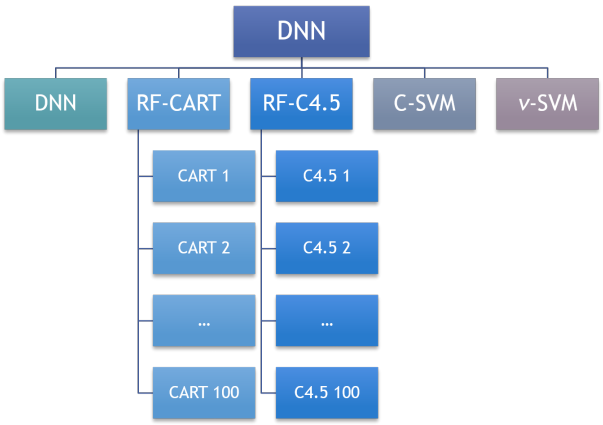

Fig. 1. Block diagram of the architecture of the stacking ensemble method

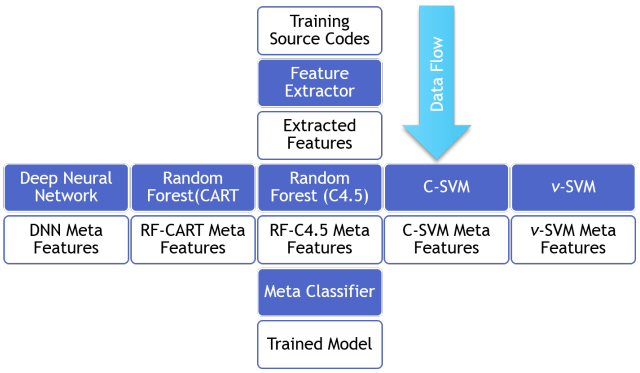

Fig. 2. Block diagram of proposed author identification approach

tion(SMO) as the classifier for SVM and achieved 98% and 80% accuracy for two different datasets.

IV. Author Identification of Source Codes Written by Multiple Authors

Our developed author identification approach consists of four phases. Firstly, source code metrics are extracted from the source code segments in the training set. These extracted metrics are then converted to feature vectors.

Secondly, these feature vectors are fed to five individual base classifiers and corresponding class labels to train the author signatures to the base classifiers. In the case of open source contribution, class-label means the owner of the source code segment, and in the case of a group of authors, the whole group is considered the class label. By author signature, the coding style of a particular class label is meant. Caruana [5] showed that, in general, for the classification problem, random forest, DNN, decision tree, and SVM are the top four algorithms. Hence, our chosen classifiers are DNN, random forest with CART decision trees [2], random forest with C4.5 decision trees [10], C-SVM and ν-SVM.

Thirdly, each of the classifiers generates the posterior class probability according to their predictions. These outputs are called meta-features. Meta-features are used as the input for a meta-classifier. Then the meta-classifier is trained based on the meta-features and output. This approach is known as stacking-ensemble. Another deep neural network is used as the meta-classifier. Figure 1 shows a block diagram of the architecture of the stacking ensemble method we have designed.

Finally, to identify the author of a new source code segment, from the test set, the same metrics are extracted from the test source code segments and converted to feature vectors. These feature vectors are fed to the meta-classifier via the base classifiers. Using the experience from the training, the meta-classifier, along with the base classifiers, predicts the class labels of the test source code segments. Figure 2 shows the block diagram of the proposed approach for author identification of source codes written by multiple authors.

In the following sub-sections, the building blocks of the author identification approach are described.

A. Dataset

Some careful considerations are needed while choosing the dataset. Data must be collected from a diverse population of programmers. They should provide enough information about the authors so that a clear distinction can be computed from author to author and a valid comparison of their programming style can be made. In addition, the dataset must be close to real-world data as well as open for academic study [13].

In our study, we have generated our dataset based on open-source codes from github.com. All the source codes have a permissive license like MIT or BSD. The dataset contains 6063 python source code segments from 8 authors/ author groups considered individual classes. Each source code segment contains roughly 226 lines on average. Each author's source code segments are roughly split into a 2:1 ratio to make the training and testing set.

Each class label consists of authors and contributors. By author, we mean the true owner of the projects. This could be a single author or a group of authors. By contributors, we mean a group of people who are not the project-owner but willingly contribute to the project by writing or editing a segment of it. The number of authors and the number of contributors per class label is listed in table I.

B. Metric Extraction

Previously, Shevertalov, Lange, Bandara, and Zhang [1], [13], [18], [21] used source code metrics for author identification. Lange selected the optimal set of code metrics using the genetic algorithm from a set of probable code metrics. Bandara used almost the same set of source code metrics. We have used the same set of metrics for our author identification approach only except the access modifier metric. The access modifier feature is present only in a limited number of programming languages and makes the whole system language-dependent. Table II shows the set of metrics to be used and corresponding descriptions.

After extracting the metrics, we have counted the number of occurrences for each possible value for each of the metrics. For

example, for underscore metrics, we have counted the number of words with no underscore, one underscore, two underscores etc. These counts have been fed to the base classifiers.

C. Base Classifiers

There are a total of five base classifiers in our author identification system. They are – DNN, random forest based on CART, random forest based on C4.5, C-SVM and ν-SVM. Each of the base classifiers is described below:

1) Deep Neural Network: The DNN model used as the base classifier consists of 14 layers. Data are fed to the DNN as batches of 32 entries. They are one input layer, followed by eight fully connected layers, a dropout layer, a fully connected layer, a dropout layer, a fully connected layer and finally, the output layer.

In the fully connected layers, *ReLU* activation function and in the output layer *softmax* activation function are used. *Categorical cross-entropy* is chosen as the loss function. *Adam* [11] optimizer is used to optimize the network.

2) Random Forest: The second base classifier is a random forest with one hundred decision trees. Classification and Regression Tree(CART) [2] algorithm is used to build the trees which select the split node based on Gini impurity.

The third base classifier is another random forest with one hundred decision trees. Decision trees in the third base classifier are built with the C4.5 [10] algorithm. This algorithm chooses the split node based on the entropy ratio.

3) Support Vector Machine: The fourth base classifier is a C-support vector classifier. It is a support vector machine where C is a penalty parameter for the error term.

The fifth base classifier is a ν-support vector classifier. It is a support vector machine where ν is the upper bound of training error and the lower bound of the number of support vectors.

D. Meta Classifier

We have used another deep neural network as the meta-classifier. The outputs of the base classifiers (meta-features) are fed to the meta-classifier to learn the mapping from the meta-features to the actual output.

TABLE I
NUMBER OF AUTHORS AND CONTRIBUTORS FOR EACH CLASS

Class Label	Number of Authors	Number of Contributors
Azure	3	136
GoogleCloudPlatform	33	820
StackStorm	2	147
dimagi	2	101
enthought	9	224
fp7-ofelia	1	4
freenas	2	126
sympy	2	712

TABLE II
SET OF CODE METRICS AND DESCRIPTIONS

Metric Name	Metric Description
Line Length	This metric measures the number of characters in one source code line.
Line Words	This metric measures the number of words in one source code line.
Comments Frequency	This metric calculates the relative frequency of line comment, block comment and optionally doc-comment used by the programmers.
Identifiers Length	This metric calculates the length of each identifier of programs.
Inline Space-Tab	This metric calculates the whitespaces that occur on the interior areas of non-whitespace lines.
Trail Space-Tab	This metric measures the whitespace and tab occurrence at the end of each non-whitespace line.
Indent Space-Tab	This metric calculates the indentation whitespaces used at the beginning of each non-whitespace line.
Underscores	This metric measures the number of underscore characters used in identifiers.

The neural network consists of 19 layers. They are one input layer, followed by eight fully connected layers, a dropout layer, two fully connected layers, a dropout layer, a fully connected layer, a dropout layer, a fully connected layer, a dropout layer, a fully connected layer, a dropout layer, and finally, the output layer. The output from this output layer is the final output of our author identification system for the source code segment written by multiple authors.

The activation functions of the network are *ReLU* for fully connected layers and *softmax* for the output layer. The loss function used in the meta-classifier is *categorical cross-entropy*. *Stochastic Gradient Descent(SGD)* is used as the optimizer of the meta-classifier.

E. Training

We have implemented our author identification system for source code segment written by multiple authors in multi-class classification category. Here, a unique list of authors(or groups of authors) of the source code segments in the training set is treated as classes. The author identification system produces confidence for each class of being the actual class of the given

1. Extract code metrics from the training set
2. Convert the code metrics to feature vectors
3. For each model in {DNN, RF-CART, RF-C4.5, C-SVM, *v*-SVM}:
    1. Train model based on the training features
4. Stack the outputs of each model to form meta features
5. Train the meta classifier based on the meta features
6. Predict the authors of unkown samples using the classifiers

Fig. 3. Steps for training the stacking ensemble system

TABLE III
PARAMETER VALUES OF THE CLASSIFIERS

Classifier	Parameter	Value
C-SVM	C	1.0
ν-SVM	ν	0.15
Base DNN	learning rate	0.01
Adam optimizer	β_1	0.9
	β_2	0.999
Meta DNN	learning rate	0.001
SGD optimizer	momentum	0

TABLE IV
ACCURACY OF THE BASE CLASSIFIERS

Classifier Name	Accuracy
Deep Neural Network	82%
CART Based Random Forest	83%
Random Forest	83%
C-Support Vector Machine	79%
ν-Support Vector Machine	79%

source code. The actual author is expected to have the highest confidence.

Roughly, 67% source code segments from each class formed the training dataset, and the rest are used for testing. The training set contains 4034 files, and the test set contains 2039 source code segments.

The training stage of our system is divided into three phases – feature extraction from the source code segments, training the base classifiers and training the meta-classifier. Figure 3 shows the steps followed in our author identification system for source code segment written by multiple authors.

First of all, the source code metrics mentioned in table II are extracted from source code segments. Then the extracted metrics are converted to feature vectors as mentioned in section IV-B. These feature vectors are fed to each of the base classifiers as input.

The base classifiers run according to their learning algorithm to learn to identify the writing style of each class. During this training phase, several configurations of each of the base classifiers, specially DNN, are used to determine which configuration works best for the training set.

After completing each base model's training, the posterior probability for each input in the training set is generated. This produced a $5 \times |classes|$ sized feature vector for each input feature vector where $|classes|$ is the number of classes. These feature vectors are known as meta-features. Meta features are fed to the meta-classifier and the class labels through which the meta-classifier learned to predict the actual class from the meta-features.

V. EXPERIMENTAL RESULTS

A. Experimental Setup

While implementing our author identification system for source code segment written by multiple contributors, we have used Keras as the framework for deep neural networks and Scikit Learn [17] as the library for general-purpose machine learning. For data pre-processing and visualization, we have used NumPy and pandas [15] library. We have developed a feature extractor that extracts the features mentioned in table II from the source codes.

For C-SVM, the parameter C is a penalty for the error term. For ν-SVM, the parameter ν is an upper bound to the training error and lower bound to the number of support vectors. During the experiment, we found that for both the random forests, a hundred trees were sufficient to converge to the highest accuracy. After numerous iterations, we reached a decision that the set of values stated in table III classifies the source code segments most accurately.

Accuracy and *f1-score* were used to evaluate the accuracy of our method. *Accuracy* is the ratio between the number of correctly identified samples and the number of total samples. *F1-score* is the harmonic mean of *precision* and *recall*. *Micro averaging* was used to compute the *f1-score*.

B. Results of The Base Classifiers

Table IV contains the accuracies for the five base models of our stacking ensemble method.

C. Results of The Meta Classifier

After training the meta-classifier by the meta-features, we have achieved 87% accuracy with f1-score 0.86. Identifying the authorship of source codes is more difficult when the number of authors is more than one, as the writing style of the source code is then inconsistent from segment to segment. Table V shows a comparison between the type of features, language independence, the capability of handling multiple authorship, the number of classes and the total number of source code segments used in training and testing. From that table, we can see that even after dealing with source code segments written by multiple authors, our method has achieved an accuracy that is pretty close to that of the methods that deal with single authors. Our chosen set of metrics is compact and is still able to achieve satisfactory accuracy. Alongside, several works suffer from choosing a set of metrics that are not language-independent. So, the main contribution of this work is the identification of multiple authors using a language-independent set of metrics.

VI. CONCLUSION

Here, we have proposed a new approach for identifying the author of the source code segment where the number of authors of the source code segment is more than one. The main challenge of this work is to select the base estimators from a large number of possible combinations. Again, as several classifiers need to be trained, each classifier needs to be fine-tuned individually to produce an excellent final result. On the

TABLE V
COMPARISON AMONG THE METHODS FOR SOURCE CODE SEGMENT AUTHOR IDENTIFICATION

Method Name	Features	Language independent features	Multiple author	Number of classes	Total source code segment	Accuracy
Information retrieval approach [3]	Character level n-gram	Yes	No	100	1640	67%
Code metric histogram [13]	7 code metrics	Yes	No	20	4068	55%
Genetic algorithm [18]	4 code metrics	Yes	No	20	N\A	75%
Deep neural network [1]	9 code metrics	No	No	10, 10, 8, 5, 9	1644, 780, 475, 131, 520	93%, 93%, 93%, 78%, 89%
Support vector machine [21]	46 code metrics	No	No	8, 53	8000, 502	98%, 80%
Stacking ensemble method	8 code metrics	Yes	Yes	8 (group of authors)	6063	87%

other hand, identifying the authorship of source code segments is more complicated when the number of authors is more than one.

We have developed a stacking ensemble classifier consisting of five base classifiers and a meta-classifier that uses a relatively small set of code metrics that are relatively easy to compute and language-independent.

Even though our stacking ensemble method achieved satisfactory accuracy, this still can be improved. Even though our code metrics are language-independent, we only tested with python source code segments. Future works may test on other languages and check how the set of metrics works for other languages. Other sets of metrics can also be examined to see how they contribute to the writing style of source code segments.

REFERENCES

[1] Bandara, U., Wijayarathna, G., "Deep neural networks for source code author identification," In: M. Lee, A. Hirose, Z.G. Hou, R.M. Kil (eds.) Neural Information Processing, pp. 368–375, Springer Berlin Heidelberg, Berlin, Heidelberg (2013)

[2] Breiman, L., Friedman, J., Stone, C.J. Olshen, R., "Classification and Regression Trees," CRC Press (1984)

[3] Burrows, S., Tahaghoghi, S.,"Source code authorship attribution using n-grams," In: A.T. Amanda Spink, M. Wu (eds.) Proceedings of the Twelfth Australasian Document Computing Symposium, pp. 32–40, School of Computer Science and Information Technology, RMIT University (2007)

[4] Burrows, S., Uitdenbogerd, A., Turpin, A.,"Comparing techniques for authorship attribution of source code," Software: Practice and Experience 44 (2014)

[5] Caruana, R., Karampatziakis, N., Yessenalina, A., "An empirical evaluation of supervised learning in high dimensions," In: Proceedings of the 25th International Conference on Machine Learning, ICML '08, pp. 96–103, ACM, New York, NY, USA (2008). DOI 10.1145/1390156.1390169. URL http://doi.acm.org/10.1145/1390156.1390169

[6] Duracik, M., Krsak, E., Hrkut, P., "Current trends in source code analysis, plagiarism detection and issues of analysis big datasets," In: TRANSCOM 2017: International scientific conference on sustainable, modern and safe transport, Elsevier: Procedia Engineering, vol. 192, pp. 136–141 (2017)

[7] Frantzeskou, G., Stamatatos, E., Gritzalis, S., " Supporting the cybercrime investigation process: Effective discrimination of source code authors based on byte-level information," In: J. Filipe, H. Coelhas, M. Saramago (eds.) E-business and Telecommunication Networks, pp. 163–173, Springer Berlin Heidelberg, Berlin, Heidelberg(2007)

[8] Frantzeskou, G., Stamatatos, E., Gritzalis, S., Katsikas, S., "Source code author identification based on n-gram author profiles," In: I. Maglogiannis, K. Karpouzis, M. Bramer (eds.) Artificial Intelligence Applications and Innovations, pp. 508–515. Springer US, Boston, MA (2006)

[9] Gray, A., Sallis, P., MacDonell, S., "Identified: A dictionary-based system for extracting source code metrics for software forensics," In: Procceedings of SE: EI&P, pp. 252–259, IEEE Computer Society Press, Washington, DC (1998). DOI 10.1109/SEEP.1998.707658

[10] JR, Q., "C4.5: Programs for Machine Learning," Morgan Kaufmann, San Mateo (1993)

[11] Kingma, D., Ba, J., "Adam: A method for stochastic optimization," In: Procceedings of 3rd International Conference for Learning Representations, San Diego (2015)

[12] Kothari, J., Shevertalov, M., Stehle, E., Mancoridis, S., "A probabilistic approach to source code authorship identification," In: Proceedings of International Conference on Information Technology: New Generations, IEEE (2007)

[13] Lange, R.C., Mancoridis, S., "Using code metric histograms and genetic algorithms to perform author identification for software forensics," In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07, pp. 2082–2089. ACM, New York, NY, USA (2007). DOI 10.1145/1276958.1277364. URL http://doi.acm.org/10.1145/1276958.1277364

[14] Maclin, R., Opitz, D., "Popular ensemble methods: An empirical study," Journal Of Artificial Intelligence Research 11, 169–198 (1999)

[15] McKinney, W., "Data structures for statistical computing in python," In: S. van der Walt, J. Millman (eds.) Proceedings of the 9th Python in Science Conference, pp. 51–56 (2010)

[16] Mirza, O., Joy, M., "Style analysis for source code plagiarism detection," In: Proceedings of International Conference on Plagiarism across Europe and Beyond, pp. 53–61. Brno, Czech Republic (2015)

[17] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., "Scikitlearn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011)

[18] Shevertalov, M., Kothari, J., Stehle, E., Mancoridis, S., "On the use of discretized source code metrics for author identification," In: M.D. Penta, S. Poulding (eds.) Proceedings of 1st International Symposium on Search Based Software Engineering, pp. 69–78, IEEE Computer Society, Cumberland Lodge, Windsor, UK (2009)

[19] Tennyson, M.F., Mitropoulos, F.J. "A bayesian ensemble classifier for

source code authorship attribution," In: A.M.T. et al. (ed.) SISAP, LNCS, vol. 8821, p. 265276. Springer International Publishing, Switzerland (2014). DOI 10.1007/978-3-319-11988-525

[20] Yang, X., Xu, G., Li, Q., Guo, Y., Zhang, M., "Authorship attribution of source code by using back propagation neural network based on particle swarm optimization," PLOS ONE 12(11), 1–18 (2017). DOI 10.1371/journal.pone.0187204. URL https://doi.org/10.1371/journal.pone.0187204

[21] Zhang, C., Wang, S., Wu, J., Niu, Z., "Authorship identification of source codes," In: C. L., J. C., S. C., Y. X., L. X. (eds.) Proceedings of Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data, Lecture Notes in Computer Science, vol. 10366, pp. 282–296, Springer, Cham, Switzerland (2017)