Journal of Multidisciplinary Research at Trent

Volume 3 Issue 1

Source Code Similarity Indicators Based on Machine Learning

Ajinkya Kunjir[a], Jinan Fiaidhi[a*]

[a] *Department of Computer Science, Lakehead University, 955 Oliver Rd., Thunder Bay, P7B 5E1, Ontario, Canada*
***Correspondence to**: jfiaidhi@lakeheadu.ca*

Received: March 24, 2021; **Accepted:** December 19, 2021; **Published:** December 31, 2021

Abstract

This paper presents a hybrid system-driven machine learning research to classify plagiarized assignments in academics. Previous research has shown several similarities or clone detector engines developed to identify plagiarized source-code assignments and make things easy for the evaluators to determine the code thieves. The research presented in this paper is the successor of the former module in which our similarity detector engine labelled 'SimDec' detects plagiarism in C and C++ source codes. This paper will illustrate machine learning algorithms' work for solving prediction problems on the system data generated by the 'SimDec' software engine. Multi-class SVM, Logistic regression, and a Simple Neural Network in the supervised learning spectrum have been implemented on the attribute counting methods (ATM) generated data to predict the source codes' plagiarism severity. In this incremental process, the experimentation of applying unsupervised learning on the same data by discarding the target variable was performed to deliver a comparative study of ML algorithms. Finally, the directions for improving the system are pointed out. The proposed hybrid approach would reduce our similarity detector's time complexity and processing speed.

Keywords: *Machine learning, Multi-class SVM, Logistic Regression, Neural Networks, Accuracy, Unsupervised Learning*

Introduction

Plagiarism in programming language source codes is an extreme concern in professional and academics area. Zobel (2004) stated that students often plagiarize their source code assignments from their colleagues in the class and the colleagues often refer to the internet for solving their coding problem. There is a linear dependency observed in this process, making it complicated to track the homework's originality and detecting plagiarism in source codes. Relating to the situation in a class of more than 100 students where it is nearly impossible to check students' assignments for similarity manually. The idea of validating the first student's assignment with 99 others is time-consuming and impractical to do by human evaluators. Previous research describes the two ideal methods for plagiarism detection, such as the Structure-based method (SM) and the Attribute counting method (ATM). SM technique directly considers the source code document's structural characteristics when developing the model, whereas the ATM's extract information or

features from the source code documents. The researchers in their empirical approach for plagiarism detection, mentioned six levels for source code detection (Faidhi & Robinson, 1987). Level 0 represented the lowest level where the student copies someone else's work without any modification and level 6 is the highest level where the plagiarism is done smartly by non-novice programmers. (Arwin & Tahaghoghi, 2006) The authors stated did not favor the structured methods as it was proved that the plagiarized document's structural properties differ from their original state and further face complications in identifying similarities. The already existing similarity detectors using attribute counting metrics technique are not dependent on the documents' structural properties and, therefore, overcome the problem mentioned in the paragraph above. However, the systems implementing ATM methods are not delivering accurate results and failing to meet the academic requirement of identifying similar source code assignments.

Considering all the limitations and advantages of existing systems, we proposed a new system in our previous paper labelled as 'SimDec', a java-based forensic engine responsible for identifying plagiarism in C and C++ source codes. The system uses the ATM technique of extracting information from the comparison observation of applying mathematical similarity algorithms on the source codes. The source code dataset selected for performing the former experimentation was the IEEE homework programming dataset available at this web source. The designer of the academic dataset mentioned the contents of the dataset such as student's submitted assignments of courses A and B in two successive years 2016 and 2017 (Ljubovic, 2020). Subject 'A' was taught in 2016 in C programming language, and subject 'B' is in C++ programming language. This dataset involves thousands of C and C++ source codes subdivided into assignment folders. More details about the dataset is available on the web source mentioned above. The ATM driven system generates a dataset consisting of the features extracted from the documents. We have proposed a set of supervised machine learning algorithms for predicting low, average and high levels of plagiarism. The main intention of engrossing machine learning with our 'SimDec' system was to reduce the system's time complexity and increase the efficiency of the overall system. There is a famous quote around in the machine learning area saying, "*There is no single learning algorithm in any field that induces the most accurate learner"*. It is always safe to use more than one learning method to test your hypothesis, and therefore, we have used three supervised learning algorithms in this study and compared the results with two unsupervised learning methods. The data being adaptable could be modified for both the ML techniques, and hence we decided on delivering a comparative study on the selected data.

For the rest of the paper, Section II will present a short literature survey of all the plagiarism detection systems using ML and DL and the need for ML with software systems. In Section III and IV, we will describe the supervised and unsupervised learning algorithms and their implementation on our system generated data. We will discuss the comparative study's experimentation results and justify which technique outclasses the other and the reason in section V of this paper. The paper concludes the research with a conclusion and future work aided by the references at the end.

.

Review

Literature Survey

There has not been any exceptional research on similarity detectors or clone detectors using the attribute counting method, but there have been a few worth mentioning in

this section. The authors used ATM feature metrics to design their code plagiarism tool using machine learning algorithms (Bandara & Wijayarathna, 2011). They trained and tested three ML algorithms such as Naïve Bayes, Adaboost meta-learning and kNN to improve the system's accuracy. The dataset consisted of java source codes and hence the features generated were token frequencies, occurrences of characters and other relevant elements. The system was vulnerable to coding standards and would fail if it followed certain procedures. The author in their thesis on textual plagiarism detection using ML methods, compared the system with the results of the Plagiarism, Authorship and Social software misuse (PAN) workshop (Kalleberg, 2015). They decided to carry on the experimentation with random forest and decision trees and had the optimal solution compared to the previous PAN workshop participants. The PlagDet score of 0.9 was obtained in detecting plagiarism with obfuscation strategies. The researchers have proposed a novel source code comparability recognition strategy which uses source code metric histogram and hereditary calculation (Lange & Mancoridis, 2011). The exploration utilized a hereditary calculation to distinguish the improved arrangement of source code measurements. The authors recorded a total accuracy score of 55% for identifying each source code file's true authors.

Not to mention the use of deep learning techniques for detecting plagiarism. In their research, (Suleiman et al., 2017) proposed a word2vec model to detect similarities in Arabic language words. Word2vec is deep learning technique that represents words as features of vectors with high precision. The authors used the OSAC corpus to train the model and used the cosine similarity measure to link the word vectors to computer similarity. The author's system proved efficient in detecting plagiarism in arabic text even if the text is altered or replaced with synonyms. We will discuss the need of machine learning indicators in a software system in the following subsection.

Need of ML in Software Systems

Since the invention of LISP and FORTRAN, ML has played a major role in software systems. The tools for building low-level and high-level programming languages didn't change their layout or appearance but are essentially the same. Take a look at the fancy editors, the editors have the features such as colour highlighting, predicting the next word when typing the current, and different programming styles. A software system is a mixture of source codes that is formulated in a flow to get a series of output. Adding machine learning and pattern recognition to the software code can enhance the system in numerous ways by increasing its efficiency. As language changes and usage shifts, new elements are discovered and the neural network can be revisited and retrained on the new data. To discuss the scope of machine learning in our proposed research, it is mainly implemented to decrease the running operation's time complexity. The time taken by the program to build and run the program could be reduced gradually as the number of computations would be reducing as well. To dive into the details, the number of computations required to assign a plagiarism level label (such as low, average or high) to one comparison takes substantial time, starting from mathematical calculation to storing and loading from the dataset. Machine learning could easily omit the computing time as the model(s) could be trained on huge data and then can be used to predict the target label if the accuracy is reasonably well. This subsection will justify the above-mentioned hypothesis by covering the machine learning algorithms and relevant concepts. The dataset selected for this experimentation is gullible with supervised and unsupervised for predictive and descriptive analysis both. A comparative study observed in the below-given sections and the

best suitable technique for the system would be considered a final analytical decision in this software engineering process.

Implementing machine learning algorithms on the data gathered in the form of features from source code comparison. Our 'SimDec' system assigns a label for each comparison pair such as low, average and high. The main purpose of making the system do it is to make it compatible with the supervised learning algorithms for prediction. Because without the class label, the generated data will only be good for unsupervised learning and not suitable for classification or prediction. This information is stored in relational database systems and fetched by the users at the front-end. The labels generated in the dataset is required for applying supervised machine learning algorithms for achieving classification. In this second module of our research, the objective is to extract and infiltrate the output of the SimDec software system and forward as an input to the machine learning algorithms. The development environments and platforms for both the modules are different and the configuration is independent. Both of the modules are not directly interlinked with each other and separated by the dataset. Dataset is the middle layer between the software system and machine learning platforms. This module involves supervised and unsupervised learning, and the decision is concluded by observing and analyzing the results.

Methods

Supervised Learning Techniques

The name 'Supervised Learning' is self-explanatory to mention the need and importance of this technique. Supervised techniques is like a teacher guiding a student, and during the training process, the algorithms will search for patterns in the data that correlate with the desired outputs. The nature of a supervised learning algorithm is to train on a set of data to get the patterns right and then predict the label or target variable for the newly presented data. Regression and classification can be defined as two categories of supervised learning. The regression approach is undertaken if a prediction is made for a continuous variable such as numerical scores, amount, percentage etc. Regression can be further sub-divided into linear and logistic, where logistic regression is designed for categorical variable prediction and linear for numerical predictions. Classification algorithms predict categorical labels such as high/low/medium, 0/1/2 or true/false in the datasets. The job is to take the input and assign a class or category that fits the training data provided. A problem of classification can be tackled with plenty of calculations, for example, Support vector machines, Naïve Bayes, Decision trees, Neural Nets and K-Nearest Neighbor techniques. The authors implemented supervised ML algorithms such as Naïve Bayes, logistic regression, kNN and random forest for a credit card fraud detection task (Samidha Khatri et al., 2020). They classified fraudulent and genuine transactions for newly entered data after successful training and testing of their models. We have considered three popular and quality algorithms for conducting this experimentation: SVM's, Logistic regression, and Neural Networks. A brief description of each one of the above mentioned is given in the lower sections as follows:

Multi-class Support Vector Machines (SVM)

The authors developed support Vector Machines (SVM) algorithms at the AT & T Bell laboratories in 1992 (Cortes et al., 2011). Support vector machines are supervised learning algorithms that can solve a classification problem using two-class (high, low) and multi-class (high, medium, and low). According to the data formulated in our

system, a multi-class SVM was a favorable one as there are three class labels: high, average and low. The goal of a multi-class SVM is to discover a hyperplane in an n-dimensional space that isolates the information focuses as indicated by their classes. The information focuses closest to the hyperplane are called 'Support-Vectors'. Another justification SVM's to be called kernelized vectors is because they convert input information space into a higher-dimensional space. The number of classifications required for one vs one multi-class classification can be found out by the formula given below:

Kernel Functions:

Linear, polynomial, radial basis and sigmoid are popular kernel functions available in the scikit-learn kit. The equations of the four parts are given below:

- Linear Function - $k(x_i, x_j) = x_i * x_j$
- Polynomial function - $k(x_i, x_j) = (1 + x_i + x_j)d$
- Radial Basis function (RBF Kernel) - $k(x_i, x_j) = exp$
- Sigmoid function - $k(x_i, x_j) = tanh\ (ax^T y + c)$

We will not discuss SVM's root contents by explaining the mathematical concept as that knowledge is available on hundreds of websites online. The SVM algorithm performed efficiently on the SimDec data for multi-class classification and obtained an accuracy of 99% for both training and testing. The details of multi-class SVM results on our system's data are briefly mentioned in section V of this paper.

Logistic Regression

Logistic regression is the simplest type of supervised Regressor used only when the target variable is categorical. A famous statistician (Cox, 2020) invented logistic regression as a binary probability model. For numerical target variables, linear regression is recommended due to compatibility. An example would be to predict/classify whether the statement is true or false or yes or no. The function used in logistic regression is the sigmoid function.

There are several types of logistic regressions, such as binary logistic regression for predicting 0 or 1, multinomial logistic regression for classifying more than two labels such as high, average and low. A threshold needs to be set to predict the data class because the estimated probability is classified into targets. The cost function is considered in the central equation for linear regression and not logistic regression. MSE (Mean square error) is used with linear regression, but it is used with logistic function to create a non-convex function of parameters. The cost function for logistic regression is defined below in which, if y = 1, the output approaches to O as hΦ(x) approaches to 1. The cost to pay grows to infinity as hΦ(x) comes to O. The same situation applies when y = 0, where there are bigger penalties when the label is y = 0, but the algorithm predicts hΦ(x) = 1.

The other results of the logistic regression algorithm implementation on system data are described in section V of this paper, where the accuracy obtained with the same technique on system data is around 98% for training and testing the model. The below-given figure displays the code output on applying logistic regression from the SKLearn package on the training and testing data. The precision, recall, f1-score and other validation techniques for plagiarism level classes such as

low, average and high represented as 0, 1, and 2 (see Figure 1).

Simple Neural Networks

The professors at the University of Chicago invented neural networks derived from the calculus in nervous activity (McCullough & Pitts ,1943). To give a short and simple description, a neural network is constructed with thousands of neurons and one neuron is a basic unit of the network. Neurons simply take the input, process the computation and give the output. For example, in a three-neuron neural network, the three inputs are multiplied with weights and added with a bias 'b' as shown below:

*x1 -> x1 * w1, x2-> x2*w2, x3->x3*w3*
(where w1,w2,w3 are weights of the network)
Secondly, the inputs are added with a bias and passed through an activation function as follows:

*Y = f(x1 * w1 + x2 * w2 + x3 + w3 + b)*

The activation functions can be of 7 varying kinds: sigmoid, ReLu, Tanh, linear activation, non-linear activation, softmax and swish. A detailed explanation about all the activation functions can be found out at https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/.

The setup can be established and coded in python using Keras and TensorFlow framework, getting back to neuron-based neural networks. The neural network could be built of several layers and activation functions to get a satisfactory result. The neural net model should be trained and tested on a dataset before putting it to actual real-world testing. Standard loss and error computation evaluate the model like all other machine learning models. MSE (mean square error) is the loss function used for computing a neural network evaluation. Neural nets can be configured in three ways: feed-forward networks, backward propagation, and ensemble learning (hybrid NN's). Lastly, we would like to briefly discuss the optimizers and the optimizers available for balancing the weight and minimizing the loss. The learning rate encompassed in the optimizers sleeps up the training time of the model. All these parameters can be altered and played with for getting an appropriate score. The code for a model which we built for applying on our system's data is shown below as follows:

```
input_dim = len(data.columns) - 1
model = Sequential()
model.add(Dense(8, input_dim = input_dim , activation = 'relu'))
model.add(Dense(10, activation = 'relu'))
model.add(Dense(10, activation = 'relu'))
model.add(Dense(10, activation = 'relu'))
model.add(Dense(3, activation = 'softmax'))
model.compile(loss = 'categorical_crossentropy' , optimizer = 'adam' , metrics = ['accuracy'] )
model.fit(train_x, train_y, epochs = 10, batch_size = 20)
scores = model.evaluate(test_x, test_y)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

There has been an imposition of four dense model layers consisting of three ReLu activation layers and one softmax layer. The loss function for compilation is categorical cross-entropy with Adam optimizer for minimizing loss. The model is trained with training data (x,y) on ten epochs and a batch size of 20 and then evaluated on testing data parameters (x,y). The testing score achieved with 20 batch size is 93.22 %, with a minimum loss of 0.1619. The model looks good at 93.22% accuracy on the current system data, and therefore, new data for another student assignment folder is given to the model for testing. The trained and tested model is saved, loaded back on the

environment and tested on the data without the target variable. The one-hot encoding is done for the target variable 'Plagiarism' low, average and high as '0', '1', and '2'. The data is given to the model. predict(datafilename) and a numpy array is obtained as the predictions array. On printing the list, the predictions obtained for the data is shown in figure 2.

Unsupervised Learning Techniques

The machine learning technique where the target variable or class label is not required to train the algorithms for predictions is known as unsupervised machine learning. In other words, there is no need to supervise the classifier model in this type of learning. The data is fed to the model to discover the patterns and explore new information based on the learning. Unsupervised learning mainly deals with data that has no label/target. Clustering analysis or descriptive mining holds the major portion of unsupervised ML. Other existing ML techniques are given as association rules and variations of association rules. There are various clustering algorithms such as Hierarchical clustering, K-means, KNN classification, principal component analysis, single value decomposition and independent component analysis. To contradict the previous relevant research, (Kumar & Dwivedi, 2020) conducted an unsupervised learning approach on the credit card data for fraud detection. The group of existing methods such as auto encoder, isolation forest and k-means was used with the proposed unsupervised neural networks. The accuracy obtained with NN's was 99.98% and competed with the supervised learning results. The study conducted on the SimDec system data involves two types of unsupervised clustering such as K-means and PCA due to the data's simple nature. The coding for both techniques has been completed using the python programming language in this module. The purpose of conducting unsupervised studies is to compare it with supervised learning and conclude the best ML category for satisfying this research's intention.

K-means Algorithm

K-means clustering is an iterative algorithm that helps find the clustering's highest value by selecting a centroid / central point. The number of clusters 'k' are chosen by coding the elbow method curve where the value of bent curve is selected as the desired number of clusters for the k-means operation. The entire data is clustered into 'k' groups, and the algorithm's output is a group of 'labels'. The centroids are the hearts of the clusters and the bigger the cluster, the lower the granularity and the value of 'k'. The small clusters often have large granularity and a bigger value of 'k'. (Singh & Reddy, 2020) gave simple pseudocode for the k-means algorithm in their paper and recited the same as shown below:

The K-means Clustering Algorithm

1. Input data points 'D' and specify the number of clusters 'K'
2. Initialize central points or centroids randomly.
3. Each data point in the cluster should be associated with the nearest centroid. This will divide data points into ‘k’ clusters.
4. Repeat the above two steps after recalculating the position of centroids.
5. Represent data points with clusters

The program starts with declaring imports and reading the data file as 'data' variable and then dropping the categorical attribute from the pandas frame, then assigning the same categories to a label.

```
Labels = Data['plagiarism']
Data = Data.drop(['filenames', 'plagiarism'],
axis = 1)
Labels_keys = Labels.unique().tolist()
Labels = np.array(Labels)
```

```
print('Plagiarism levels: ' + str(Labels_keys))
```

The labels will print the plagiarism levels as high, medium and low. The data is then scaled and standardized using scalar. fit_transform (data). The optimal cluster 'k' values are checked using the elbow method, and the desired number is calculated (K=2 in this case). K-means function is computed in the cellblocks, and the related scores such as inertia, v-means, homo, ARI, AMI and silhouette score are outputted for the given data. With k=2 clusters, the system data's silhouette score is 0.349 and 0.21 for k=3.

Referring to figure 3, upon giving the data of the SimDec system to the k-means unsupervised clustering algorithm, it is observed that the optimal number of clusters for the operation would be 2 (k=2) for 15,344 records. The k-means performed in python cluster the data into two groups where 'Average' and 'High' labelled data (Label is dropped as shown in the python code above) is allocated to the second 'Low' and 'Average' is assigned to the first cluster. As you can see from the figure, the highest number of records are labelled as 'Average' are 6191 and therefore fluctuate in both the clusters. According to the algorithm's theory, unsupervised k-means are performing well in clustering with our system's data. Silhouette score ranges from -1 to 1, and any value closer to '1' means that cluster is well separated from each other and the value nearby '0' denotes that clusters are overlapping. The silhouette scores obtained with our data (0.349) indicate that clusters overlap and that records with 'Average' label tops the count supports the overlapping theory.

Principal Component Analysis (PCA)

Principal component analysis (PCA) is another clustering technique that acts as a dimensionality reduction mechanism (Pearson, 1901). PCA procedure could decrease a considerable variable arrangement into a little set that contains the vast majority of the data in the huge set. PCA for the same dataset as used for k-means has been conducted on the python environment. The first step is to find the optimal number of features for dimensionality reduction.

```
pca = PCA(random_state=123)
pca.fit(Data)
features = range(pca.n_components_)
plt.figure(figsize=(8,4))
plt.bar(features[:15],
pca.explained_variance_[:15],
color='lightskyblue')
plt.xlabel('PCA feature')
plt.ylabel('Variance')
plt.xticks(features[:15])
plt.show()
```

Looking at figure 4, 1-feature seems to be the best fit for our data as ‘0’ is default set to the maximum variance. If the PCA feature is equal to 1 and clusters = 2, the resulting components, including the silhouette score, are better than those of k-means. If we switch the PCA features from '1' to '2', the silhouette and the corresponding scores will decrease as the number of PCA components/features increases. The unsupervised clustering study shows that k-means perform exceptionally well for large data combined with the PCA technique (see figure 5).

Results and Discussion

The dataset generated by our proposed 'SimDec' system is typically designed for supervised learning as it contains a class label. It could still be used for unsupervised learning as all columns are numerical and the class label column is discarded. The three algorithms used for this study were logistic regression, multi-class SVM, and simple neural networks to discuss the result for supervised learning methods. The logistic

regression was performed on the SimDec data consisting of around 18K records and a class label with high, average and low for plagiarism levels. The records indicate the information obtained from a pair of student assignments under comparison. The data was split into 80:20 ratios, and the model was trained with three categories in the class label column. The precision, recall, f1-score and support obtained with logistic regression were 0.96, 0.98, 0.97 and 549. Due to the dataset's simple schema, the accuracy obtained with logistic regression was 0.98 or 98% with less than 60 false positives. The confusion matrix obtained for logistic regression is below (see figure 6).

Multi-class SVM was trained on the same data in this comparative study with six numeric features. At first, the predictor set was normalized for SVM training, and the library used to build the ensemble model was a random forest classifier. The hyperparameter tuning using grid search and cross-validation involved 'rbf' and 'linear' kernels varying C from 0 to 1000. The datasets for training and testing were provided as separate files to the SVM, and the scores obtained were 0.99 and 0.99 for training and testing, respectively. SVM performed better than logistic regression at classification in training and model evaluation. The latest algorithm to be discussed for supervised learning is the simple neural network algorithm. As mentioned in the solo subsection for neural networks, the model's architecture is sequential with three 'ReLU' activation layers and one softmax. The optimizer used for compiling the model here is Adam with categorical cross-entropy. The model is trained on ten epochs with a batch size of 20 and evaluated for the test data. The testing accuracy obtained with neural networks is 93.96% and could increase if epochs increase with varying batch sizes and different optimizers. The neural networks model was saved, loaded back and tested on another data type without the class label. The predictions for the new data were reasonable and paved the way of success for this research module. All three supervised techniques perform exceptionally well of the system data and would work fine for the new real-world generated data. This implementation of ML supervised algorithms with the SimDec system could reduce the time complexity and number of computations as the system won't be required to generate the class label. Unsupervised techniques help cluster and visualize the student assignments in groups but are less efficient for classification or time complexity reduction. The clustering visualization is not cumbersome to display on the web interface and is a good-to-have feature with the similarity detector system. Nonetheless, if it comes to deciding one out of the two ML experimentations, supervised learning algorithms are recommended to collaborate the SimDec system as they can predict the class label for a student assignment record without the need of the system to calculate the class label based on some criteria, indirectly resulting in reduced time complexity. To summarize this section, the accuracy obtained from all the supervised learning algorithms is shown in table I of this paper.

Conclusion

In this paper, we have emphasized supervised and unsupervised machine learning techniques for source code similarity detection. We preferred to perform a comparative study of the above-mentioned ML techniques to select one that outclasses another in performance and accuracy. After a thorough investigation, it was found out that supervised learning algorithms gave better results than unsupervised learning and therefore, we proceeded with the prediction for newly entered data.
Supervised algorithms such as logistic regression, multi-class SVM and simple

neural networks were trained and tested on the data generated by our 'SimDec' system. All three performed well, with testing accuracies ranging in 95 – 99%. The predictions obtained were efficient and satisfied the research's intention of reducing the processing time of the 'SimDec' system. This research can be extended in future by involving deep learning techniques and ensemble learners for faster and enhanced processing with minimum features. This hybrid approach emerging from two methodologies, such as software development and machine learning, breaks the barrier of coding standards and source code formatting tools. The system can detect plagiarism with level 6 severity, where the programmers change the structures and alter the lines by spamming whitespaces and comments for a quick escape. A few minor challenges, such as processing time and data handling, will be addressed later to improve the system.

Acknowledgements: The author has no acknowledgements to declare.

Author Contributions: Ajinkya Kunjir and Jinan Fiaidhi wrote the manuscript.

Conflicts of Interest: The author has no conflicts of interest to declare.

Funding: The author has no funding to declare.

References:

1. Zobel. J. (2004). *Uni cheats racket: A case study in plagiarism investigation*. In Proceedings of the Sixth Australasian Conference on Computing Education.

2. Faidhi, J.A.W, & Robinson, S.K. (1987). *An empirical approach for detecting program similarity and plagiarism within a university programming environment*, Computers & Education, Volume 11.

3. Arwin, C & Tahaghoghi, S.M.M. (2006). *Plagiarism detection across programming languages*. In Proceedings of the 29th Australasian Computer Science Conference.

4. Ljubovic, V. (2020). *Programming Homework Dataset for Plagiarism Detection*, IEEE Dataport, doi: https://dx.doi.org/10.21227/71fw-ss32.

5. Bandara, U & Wijayarathna, G. (2011). *A Machine Learning Based Tool for Source Code Plagiarism Detection*. International Journal of Machine Learning and Computing. 1. 337-343. 10.7763/IJMLC.2011.V1.50.

6. Kalleberg & Borge, R. (2015). *Towards Detecting Textual Plagiarism Using Machine Learning Methods*.

7. Lange, R & Mancoridis, S. (2007). *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). DOI:https://doi.org/10.1145/1276958.1277364

8. Suleiman,D , Awajan, A & Al-Madi, N. (2017). *Deep Learning Based Technique for Plagiarism Detection in Arabic Texts*. International Conference on New Trends in Computing Sciences (ICTCS), pp. 216-222, doi: 10.1109/ICTCS.2017.42.

9. Khatri, S, Arora, A & Agrawal, A.P. (2020). "Supervised Machine Learning Algorithms for Credit Card Fraud Detection: A Comparison," 2020 10th

International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2020, pp. 680-683, doi: 10.1109/Confluence47617.2020.9057851.

10. Cortes, C, & Vapnik, V. (1995). *Support-vector networks*, Machine Learning, 273-297.

11. Ben-Hur, A, David, H, Siegelmann, H, Vladimir, N. (2001). *Support vector clustering* .Journal of Machine Learning Research. 2: 125–137.

12. Cox, D. (2020). *The Regression Analysis of Binary Sequences*. Journal of the Royal Statistical Society. Series B (Methodological), 20(2), 215-242.

13. McCulloch, W & Pitts, W. (1943). *A logical calculus of the ideas immanent in nervous activity*. Bulletin of Mathematical Biophysics, 115–133. https://doi.org/10.1007/BF02478259

14. Rai, A.K & Dwivedi, R.K. (2020). *Fraud Detection in Credit Card Data using Unsupervised Machine Learning Based Scheme.* International Conference on Electronics and Sustainable Communication Systems (ICESC), pp. 421-426, doi: 10.1109/ICESC48915.2020.9155615.

15. Singh, D & Reddy, C. (2014). *A survey on platforms for big data analytics.* Journal of Big Data. 2. 10.1186/s40537-014-0008-6.

16. Pearson, K. (1901). *On lines and planes of closest fit to systems of points in space. T*he London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2:11, 559-572, DOI: 10.1080/14786440109462720

Table 1: Accuracies of all supervised learning algorithms

Supervised Algorithms	**Training Accuracy**	**Testing Accuracy**
Logistic Regression	98%	98%
Support Vector Machine	99%	99%
Neural Networks	93%	95%

	precision	recall	f1-score	support
0	0.96	0.98	0.97	549
1	0.99	0.99	0.99	2296
2	1.00	0.98	0.99	989
accuracy			0.99	3834
macro avg	0.98	0.99	0.98	3834
weighted avg	0.99	0.99	0.99	3834

accuracy_Score 0.9869587897756912

Figure 1: Model Evaluation parameters for logistic regression

```
finalresult[:50]
```

```
Out[38]: array([0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 2])
```

Figure 2: Predictions for new system data using Neural Networks

```
k_means(n_clust=2, data_frame=Data, true_labels=Labels)
```

orig_label clust_label	Average	High	Low
0	6191	0	4065
1	2504	2574	0

```
inertia  homo    compl   v-meas   ARI     AMI     silhouette
66926    0.303   0.464   0.366    0.213   0.303    0.349
```

Figure 3: K-means output for k=2 with related scores

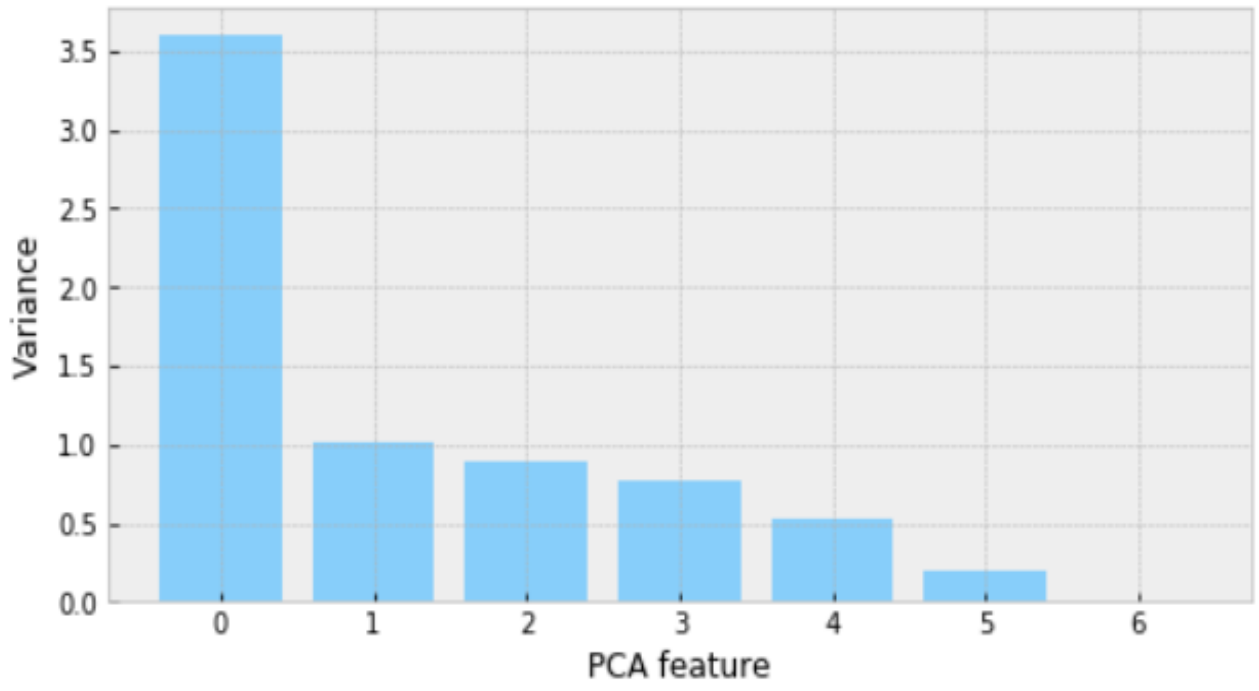

Figure 4: PCA features X variance for system data

```
pca_transform(n_comp=1)
k_means(n_clust=2, data_frame=Data_reduced, true_labels=Labels)
```

```
Shape of the new Data df: (15334, 1)
```

orig_label	Average	High	Low
clust_label			
0	2624	2574	0
1	6071	0	4065

```
inertia  homo    compl   v-meas   ARI     AMI     silhouette
15244    0.301   0.458   0.363    0.207   0.301    0.637
```

Figure 5: PCA transformation with two features

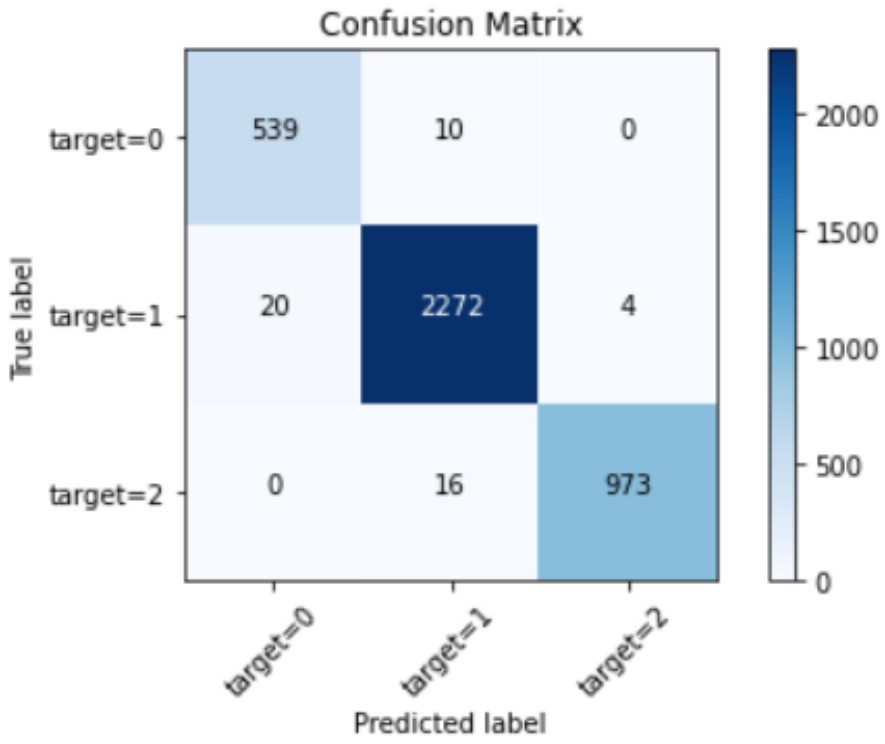

Figure 6: Logistic Regression Confusion Matrix