FUOYE Journal of Pure and Applied Sciences

Available online at https://fjpas.fuoye.edu.ng

Optimizing Support Vector Machine with Polynomial Kernel for Credit Card Fraud Detection

L. A. Odeniyi[1*], E. O. Oduntan[2], M. O. Balogun[3], M. A. Ogunrinde[4] and V. Omoniyi[5]

[1]Department of Mathematical and Computing Sciences, Koladaisi University, Ibadan, Oyo State, Nigeria.
[2]Department of Computer Science, The Federal Polytechnic, Ilaro, Ogun state, Nigeria.
[3]Department of Electrical and Computer Engineering, Kwara State University, Malete, Ilorin, Kwara State, Nigeria
[4]Department of Mathematical and Computer Sciences, Fountain University, Osogbo, Osun state, Nigeria.
[5]Department of Computer Sciences, Chrisland University, Abeokuta, Ogun State, Nigeria

ARTICLE INFO

Received: Aug. 2023
Accepted: Oct. 2023

Keywords:
Radial Basis Function, Optimization model, Sigmoid Kernel, Machine Learning Algorithms

Corresponding author:
latifat.odeniyi@koladaisiuniversity.edu.ng

DOI:**10.55518/fjpas.DYJF9012**

Abstract

The rate at which fraud relating to credit card is increasing is alarming and requires accurate detection to curb its regular occurrence. Discovery of such fraud especially in financial world is still a challenge despite several research using various machine learning algorithms. Though SVM has proved its efficiency in producing good detection parameters, there is still a need to improve the accuracy for optimal results. Hence, a Polynomial Kernel optimization method of Support Vector Machine (SVM) was adopted to identify Credit Card Fraud. The dataset used for this work was acquired from Kaggle and it presents transactions that occurred in two days with 492 frauds out of 284,807 transactions. The dataset was trained, tested and a parameter tuning using a polynomial kernel was performed, after which the performance of the model was evaluated. SVM was optimized with Polynomial kernel, Sigmoid kernel and Radial Basis Function (RBF) kernel and it was realized that sigmoid gave an accuracy of 76%, RBF gave an accuracy of 94% while Polynomial gave an accuracy of 100%. Hence, the result proved that SVM optimized with a Polynomial kernel outperformed other optimization techniques for detecting credit card fraud.

1.0

Introduction

Credit card fraud is the unauthorized possession and use of another person's credit card fraudulently. With the emergence of e-commerce, the use of credit cards for transactions experienced rapid growth for online purchases, which consequently led to an increase in the fraudulent use of credit cards. Research showed that the fraudulent cases of credit cards keep increasing as their use takes over as the preferred means of transaction for both online and offline shopping [1]. In 2019, it was discovered that the number of reported cases of credit card fraud detection increased by 72.4%, and this shows how alarming it becomes, which necessitates investigating the issue. Furthermore, according to credit card theft data, it was discovered that fraudulent use of credit cards was the most prevalent source of identity theft in 2019, with over 270,000 cases [2]. This study is very significant to the financial world, and it would help curb the rising number of credit card fraud and increase the accuracy of prediction systems.

There has been a lot of research on the use of machine learning techniques like support vector machines to detect credit card fraud, but there is a need to improve the accuracy to have an improved detection; hence, a polynomial kernel optimization method of support vector machines was adopted to identify credit card fraud [3]. This research work focuses on optimizing a support vector machine with a polynomial kernel for credit card fraud detection to serve as a means of improving the optimal result of credit card fraud detection. Several works have been done on credit card fraud detection to reduce fraudulent rates; some of them are discussed here.

Lakshmi *et al*. [3] proposed a comparative analysis technique among some machine learning algorithms such as logistic regression, decision trees, and random forest to determine which algorithm can be adopted for identifying fraudulent transactions. It was discovered from the research that the Random Forest Algorithm performed well in detecting fraud from credit cards with the best accuracy level. In 2019, Vaishnavi and Geetha [4] were able to identify an unfamiliarity in previous card transactions made by the customer, which is known as concept drift. Their research's objective was to provide a solution to the issue of notion drift for use in practical settings. The classifiers were shown to perform better than previously used ones after using SMOTE (Synthetic Minority Oversampling Technique), and they also found that the best algorithms were Random Forest, Decision Tree, and Logistic Regression.

Kartik *et al*. [5] presented a comparative analysis of the machine learning algorithms (Logistic Regression, Naive Bayes, Random Forest, K-Nearest Neighbour, Gradient Boosting, Support Vector Machine, and Neural Network) in 2021 to identify an optimal solution for predicting credit card fraud. It was discovered that the gradient boosting algorithm provided an edge over the other algorithms with an accuracy of 95.9%. Again in 2021, Chenglong *et al*. [6] took a different approach by optimizing support vector machines with the Cuckoo search algorithm and performing a comparative analysis with logistic regression, decision trees, naive bayes, random forests, and support vectors. It was discovered that support vector machines optimized with the Cuckoo search algorithm gave the best result.

It was discovered after a detailed analysis of the literature that many of the models created to test and identify credit card fraud are not being completely optimized; hence, this research seeks to address that by improving the accuracy level of the fraud detection algorithms. Table 1 below presents the summary of the related works reviewed. The model used in this research was further tuned with a number of kernels, and the most accurate kernel was selected for the purpose of detecting credit card fraud. In another work done by Bharti *et al*. [7] in 2022, the authors did a comparative study on various machine learning algorithms for detecting fraud in credit card usage. The result shows that SVM had a higher prediction rate than others. In 2022, Li *et al*. [8] worked on the optimization of support vector machines with the Firefly Optimization Algorithm (FFSVM) to detect fraud rates in financial institutions. The optimized algorithm had 85.65% accuracy, and it outperformed other non-optimization techniques.

2.0 Materials and Methods

2.1 Machine Learning Model

The process adopted for the optimization of the support vector machine with polynomial kernels for predicting credit card fraud detection is explained in this section, along with the steps that were involved in each stage. In order to create the machine learning model, the technique uses support vector machines (SVM). SVM is a method of supervised learning that may be used to solve both classification and regression issues [7] by improving the generated model's accuracy.

2.1.1 Data Collection and Preparation

The dataset used for this research was collected from the Kaggle site at https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud. Data preparation is necessary, which is done by visualizing the dataset to understand its structure and relationships between the variables and classes present. After the dataset was cleaned, it was divided into two groups: one for training and the other for testing. Figure 1 shows a screenshot of the dataset collected from Kaggle:

Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.1285
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.1671
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.3276
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.6473
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.2060

5 rows x 31 columns

Figure 1: Cross section of the dataset

2.1.2 Model Selection

This system employs the support vector machine model, due to its usefulness in solving classification problems. The code snippet for the model selection is shown in Figure 2.

```
In [18]: classifier = svm.SVC(kernel='linear')
```

Figure 2: Code snippet for applying the model

2.1.3 Model Training

For the purpose of model training, the prepared dataset was entered into the machine learning model during training. As a consequence, the model was able to learn from the data and complete the given job. The model improves in prediction over time with training.

2.1.4 Model Evaluation

Following the initial training and model development, test data was utilized to assess the system's correctness. Selection has the highest

accuracy margins, and the system does not need to be 100 percent correct.

2.2 Parameter Tuning

When an evaluation fails to meet a certain threshold, some parameters can be modified or altered to increase the probability of attaining the intended result. The cost function, interception values, and weights employed by the model in training are examples of these parameters. The method of developing the support vector model is being optimized by a polynomial kernel as part of the outcomes of the training and testing phases of the development process. A code snippet for visualizing the database is shown in Figure 3.

```
In [5]: df.describe()
```

Out[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	3.918649e-15	5.682686e-16	-8.761736e-15	2.811118e-15	-1.552103e-15	2.040130e-15	-1.698953e-15	-1.893285e-16	-3.147640e-15
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01

8 rows x 31 columns

Figure 3: A Code Snippet for Visualizing the database

The dataset is obviously skewed in this instance, as can be seen. This is because of the number of frauds, which is less than the number of non-fraud instances. As a result, the classifier gives the non-fraud class extremely little or no weight. Either the over-sampling approach or the under-sampling method can be used to tackle this issue. The under-sampling concept should only be applied if we are certain that the non-fraud class as a whole is accurately represented by the chosen few non-fraud tuples.

2.3 Optimizing the Model by Changing the Kernel

The Support Vector Machine (SVM) essentially operates on several kernels that are created for various types of data distribution. In other words, multiple kernels allow the SVM model to employ various hyperplane types on the dataset [9]. As a result, several of them are utilized, and the optimal SVM kernel for the dataset was the kernel that produced the least amount of error in the confusion matrix. Consequently, the SVM algorithm can perform at its peak level on the dataset.

3.0 Results

3.1 Evaluation of the Model

The model's accuracy was assessed during training to ensure its efficiency. To put this approach into practice, a confusion matrix, as shown in Table 1, is used to graphically represent how well a model performs. It is a method for summarizing a classification algorithm's performance. The results where the model properly predicted the positive class are considered to be true positive values. The model's outcome accurately predicted the negative classes, known as the real negative value.

The off-diagonal elements representing the false positive and false negative are those that are mislabeled by the classifier. The model's inaccurate prediction of the positive class results in misleading positive values. The false negative is a result where the model forecasts the negative class

wrongly. More accurate predictions are made when the diagonal values of the confusion matrix are greater.

Table 1: A visual representation of a confusion matrix

True Positive Values	False Positive Values
False Negative Values	True Negative Values

The model's accuracy level, weighted average, and macro average were all given careful consideration during the model evaluation. These numbers represent the average recall, F1 score, and accuracy scores, which together represent how well the model performs in categorizing the given data set. Accuracy is calculated by dividing the number of the model's accurate predictions by the total number of observations.

$$\text{Accuracy} = \frac{\text{number of correct predictions by the model}}{\text{total number of observations}}$$

The average of the grades for each individual class is known as the macro average, the average of the accuracy, recall, and F1 score. The mean of all the class scores is what is referred to as the macro average. As an illustration, the macro average for recall would be

$$\text{Average for Recall} = \frac{\Sigma recalls}{number\ of\ classes}$$

After multiplying the scores for each class by the class proportions, the weighted average is provided. As an illustration, the weighted average for recall would be

$$\frac{\Sigma(recall\ score \times support)}{total\ support}$$

A few different kernels were trained using the same dataset as the polynomial kernel and their accuracy was assessed alongside the polynomial kernel's accuracy to better understand the accuracy of the kernel applied.

3.2 SVM Optimized with various Kernels

Tables 2, 3, and 4 show the scores for the polynomial kernel, sigmoid kernel, and RBF kernel with two main classes: 0 for non-fraud and 1 for fraud cases of credit cards, considering recall and precision as main parameters. Table 5 displays the summary of the F1 scores for the polynomial kernel, sigmoid kernel, and RBF kernel with accuracy levels of 100, 94, and 76, respectively.

3.3 SVM Optimized with Polynomial Kernel

Figure 4 displays the result of a support vector machine optimized with a polynomial kernel, which recorded a 100% accuracy level.

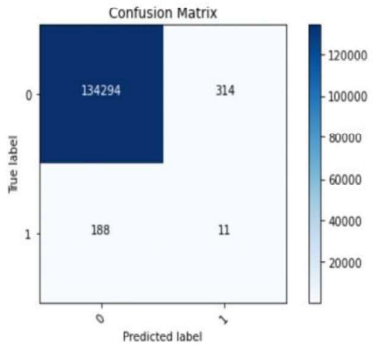
```
In [29]: classifier_b = svm.SVC(kernel='poly',class_weight={0:0.60, 1:0.40})

In [30]: classifier_b.fit(X_train, y_train)

Out[30]: SVC(class_weight={0: 0.6, 1: 0.4}, kernel='poly')

In [31]: prediction_SVM_b_all = classifier_b.predict(X_test_all)

In [32]: cm = confusion_matrix(y_test_all, prediction_SVM_b_all)
         plot_confusion_matrix(cm,class_names)
```




```
In [33]: print('We have detected {} frauds / {} total frauds.'.format((cm[1][1]) ,(cm[1][1]+cm[1][0])))
         print('Probability to detect a fraud is {}'.format(cm[1][1]/(cm[1][1]+cm[1][0])))
         print("the accuracy is : {} ".format((cm[0][0]+cm[1][1]) / (sum(cm[0]) + sum(cm[1]))))

         We have detected 11 frauds / 199 total frauds.
         Probability to detect a fraud is 0.05527638190954774
         the accuracy is : 0.9962761577662881

In [34]: from sklearn.metrics import classification_report,mean_absolute_error,mean_squared_error
         report= classification_report(y_test_all, prediction_SVM_b_all)
         print(report)

                       precision    recall  f1-score   support

                    0       1.00      1.00      1.00    134608
                    1       0.03      0.06      0.04       199

             accuracy                           1.00    134807
            macro avg       0.52      0.53      0.52    134807
         weighted avg       1.00      1.00      1.00    134807
```

Figure 4: Code snippet for applying polynomial kernel

Table 2: Box scores for the Polynomial kernel, describing performance about key parameters

Class	Precision	Recall
0 (non-fraud)	1.00	1.00
1 (fraud)	0.03	0.06

3.4 SVM optimized with Sigmoid Kernel

Figure 5 shows the sigmoid kernel with an accuracy score of 76% which is far less than the 100% accuracy recorded from the optimization with polynomial kernel.

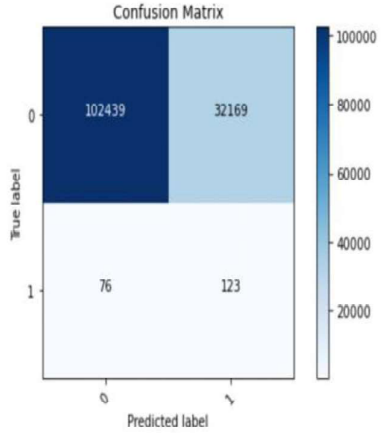
```
In [41]: classifier_b = svm.SVC(kernel='sigmoid',class_weight={0:0.60, 1:0.40})

In [42]: classifier_b.fit(X_train, y_train)

Out[42]: SVC(class_weight={0: 0.6, 1: 0.4}, kernel='sigmoid')

In [43]: prediction_SVM_b_all = classifier_b.predict(X_test_all)

In [44]: cm = confusion_matrix(y_test_all, prediction_SVM_b_all)
         plot_confusion_matrix(cm,class_names)
```




```
In [45]: print('We have detected {} frauds / {} total frauds.'.format((cm[1][1]) ,(cm[1][1]+cm[1][0])))
         print('Probability to detect a fraud is {}'.format(cm[1][1]/(cm[1][1]+cm[1][0])))
         print("the accuracy is : {} ".format((cm[0][0]+cm[1][1]) / (sum(cm[0]) + sum(cm[1]))))

         We have detected 123 frauds / 199 total frauds.
         Probability to detect a fraud is 0.6180904522613065
         the accuracy is : 0.7608061895895614
```

```
In [46]: from sklearn.metrics import classification_report,mean_absolute_error,mean_squared_error
         report= classification_report(y_test_all, prediction_SVM_b_all)
         print(report)

                       precision    recall  f1-score   support

                    0       1.00      0.76      0.86    134608
                    1       0.00      0.62      0.01       199

             accuracy                           0.76    134807
            macro avg       0.50      0.69      0.44    134807
         weighted avg       1.00      0.76      0.86    134807
```

Figure 5: Code snippet for applying sigmoid kernel

Table 3: Box scores for the Sigmoid kernel, describing performance about key parameters

Class	Precision	Recall
0 (non-fraud)	1.00	0.76
1 (fraud)	0.00	0.62

3.5 RBF Kernel

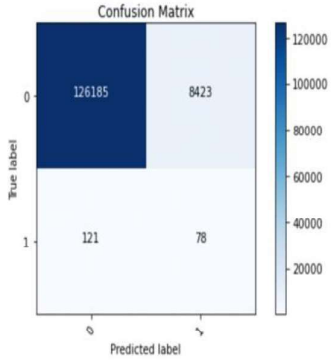
The result of SVM optimized with RBF kernel is shown in Figure 6. It was realized that the optimization recorded accuracy level of 94% which is less than the 100% of the Polynomial Kernel's accuracy score. It was discovered that SVM optimized with Polynomial kernel outperformed other optimization techniques.

```
In [35]: classifier_b = svm.SVC(kernel='rbf',class_weight={0:0.60, 1:0.40})

In [36]: classifier_b.fit(X_train, y_train)
Out[36]: SVC(class_weight={0: 0.6, 1: 0.4})

In [37]: prediction_SVM_b_all = classifier_b.predict(X_test_all)

In [38]: cm = confusion_matrix(y_test_all, prediction_SVM_b_all)
         plot_confusion_matrix(cm,class_names)
```




```
In [39]: print('We have detected {} frauds / {} total frauds.'.format((cm[1][1]) ,(cm[1][1]+cm[1][0])))
         print('Probability to detect a fraud is {}'.format(cm[1][1]/(cm[1][1]+cm[1][0])))
         print("the accuracy is : {} ".format((cm[0][0]+cm[1][1]) / (sum(cm[0]) + sum(cm[1]))))

         We have detected 78 frauds / 199 total frauds.
         Probability to detect a fraud is 0.39195979899497485
         the accuracy is : 0.9366205019027202
```

```
In [40]: from sklearn.metrics import classification_report,mean_absolute_error,mean_squared_error
         report= classification_report(y_test_all, prediction_SVM_b_all)
         print(report)

                       precision    recall  f1-score   support

                    0       1.00      0.94      0.97    134608
                    1       0.01      0.39      0.02       199

             accuracy                           0.94    134807
            macro avg       0.50      0.66      0.49    134807
         weighted avg       1.00      0.94      0.97    134807
```

Figure 6: Code snippet for applying RBF kernel (b) Classification report of RBF kernel

Table 4: Box scores for the RBF kernel, describing performance about key parameters

Class	Precision	Recall
0 (non-Fraud)	1.00	0.94
1 (fraud)	0.01	0.39

Table 5: Summary of F1-score of each kernel

	Accuracy	Macro Average
Polynomial Kernel	1.00	0.52
Sigmoid Kernel	0.94	0.49
RBF Kernel	0.76	0.44

4.0 Discussion

This research investigated the detection of credit card fraud by optimizing a support vector machine with a polynomial kernel, a sigmoid kernel, and a radial basis function kernel to determine the optimal accuracy level. The dataset used for this work was acquired from Kaggle and it presents transactions that occurred in just two days with 492 frauds out of 284,807 transactions. The dataset was trained, tested and a parameter tuning was performed on the dataset using a polynomial kernel after which the performance of the model was evaluated. SVM was optimized with Polynomial kernel, Sigmoid kernel and Radial Basis Function (RBF) kernel and it was realized that sigmoid gave an accuracy of 76%, RBF gave an accuracy of 94% while Polynomial gave an accuracy of 100%. It was realized that support vector machines, when optimized with a polynomial kernel, gave a higher accuracy level of 100 percent, which is far above optimization with the sigmoid kernel, which gave 94 percent, and the radial basis function (RBF) kernel, which gave 76 percent. Hence, the result proved that SVM, when optimized with a polynomial kernel outperformed other optimization techniques and could successfully detect credit card fraud which is in support of Mahduri and Purva, 2022 [10] and could therefore be used to predict credit card fraud detection.

5.0 Conclusion

The main reason for investigating the optimization level of various machine learning algorithms is to improve the detection of credit card fraud using the optimization method. The emergence of e-commerce exposes people to the use of credit cards, which consequently increases the number of credit card frauds at a geometric rate in society. Hence, the need to research on an approach to reduce the fraudulent act to the bearest minimum using Support Vector Machine (SVM) optimized with a polynomial kernel, a sigmoid kernel, and a radial basis function (RBF) kernel to improve the accuracy level of credit card fraud detection. It is obvious from the result obtained from this research that credit card fraud could be reduce to the bearest minimum using SVM optimized with polynomial kernel.

Declarations

Conflict of Interest

The authors hereby declare no conflict of interest.

Funding

This research did not receive any form of funding.

Authors Contribution

Authors LAO and EOO originated the idea, drafted the introduction and worked on the methodology. Authors MOB and MAO reviewed the manuscript and worked on evaluation of the model. Author

VO did the final correction on the manuscript. All authors approved the final copy of the manuscript.

Acknowledgment: Not Applicable

References

[1] Abinayaa S., Sangeetha, H., Karthikeyan, R. A., Saran, S. K. and Piyush, D. (2020). Credit Card Fraud Detection and Prevention using Machine Learning. *International Journal of Engineering and Advanced Technology*. 9(4): 1-3.

[2] Vikas P.1., Samruddhi, K., Hrushikesh, Z., Sajjad, A. K. K. and Shriviraj, A. (2020). Credit Card Fraud Detection using Isolation Forest. *International Research Journal of Engineering and Technology*. 7(3): 3538-3543.

[3] Lakshmi, S. V. S. S. and Selvani, D. K. (2018). Machine Learning for Credit Card Fraud Detection System. *International Journal of Applied Engineering Research*, 13(24): 16819-16824.

[4] Vaishnavi, N. D. and Geetha, S. (2019). Credit Card Fraud Detection using Machine Learning Algorithms. *Procedia Computer Science*. 65:631–641.

[5] Kartik M., Manthan, N., Preity, P., Riya, R. and Supriya, P. (2021) Credit Card Fraud Detection System. *International Journal of Recent Technology and Engineering*. 10(2): 158-162.

[6] Chenglong L., Ning, D., Haoyun, D. and Yiming, Z. (2021) Application of Credit Card Fraud Detection Based on CS-SVM. *International Journal of Machine Learning and Computing*. 11(1): 34-39.

[7] Bharti, K., Swapnil, B., Abhishek, H., Sameer, K. and Sunil, G. (2022). Credit Card Fraud Detection using Machine Learning. *International Research Journal of Modernization in Engineering Technology and Science.* 4(1): 1246-1249.

[8] Li C., Ding, N., Zhai, Y. and Dong, H. (2021). Comparative Study on Credit Card Fraud Detection Based on Different Support Vector Machines. *Intelligent Data Analysis*. 25(1):105 – 119.

[9] Ajeet S., Anurag, J. and Seblewonge, E. B. (2022). Financial Fraud Detection Approach Based on Firefly Optimization Algorithm and Support Vector. *Applied Computational Intelligence and Soft Computing*, 2022:1-10.

[10] Madhuri B.T. and Purva, D. N. (2022). Fraud Detection System using SVM. *International Journal of Research Publication and Reviews,* 3(11): 2864-2867.