ISSN 2278-3091
Volume 9, No.3, May - June 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse48932020.pdf
https://doi.org/10.30534/ijatcse/2020/48932020

A Comparative analysis of usage of Inheritance Metrics on software fault prediction using Machine Learning Technique

A. Brihathi[1], B. Satwika[2], D. Meghamala[3], Dr. S. Srinivasa Rao[4]
[1]Student, Computer Science and Engineering Department, K L deemed to be University, Andhra Pradesh,India,brihathiklu@gmail.com
[2]Student, Computer Science and Engineering Department, K L deemed to be University, Andhra Pradesh,India,satwikabana3@gmail.com
[3]Student, Computer Science and Engineering Department K L Deemed to be University, Andhra Pradesh, India, meghamdosa@gmail.com
[4]Associate Professor, Computer Science and Engineering Department, KL deemed to be University, Andhra Pradesh, India,srinu1479cse@kluniversity.in

ABSTRACT

In any software development lifecycle, efficient software is the output. Now-a-days software faults are occurring more and more after the deployment of the software, which are leading to severe problems that cannot be solved. So, there is an efficient need to calculate the software fault prediction before the deployment stage itself. In the process of developing a software project, we use some important concepts of programming languages such as Inheritance, Encapsulation, Polymorphism, etc.. Among these concepts, inheritance is widely used because it provides us some facilities like maintainability, reusability, data hiding, extensibility and overriding. Code reusability plays a vital role because with the help of this reusability property But, if this is used unnecessarily, it may create ambiguity and complexity in the code and also may produce some bugs or errors. So, in this project, we are going to evaluate how far we can use the inheritance metrics in a dataset and find the accuracy of these datasets with inheritance and without inheritance and write the comparative results. In this paper, up to 65 public datasets are collected from the public repository, and the comparative results of CK metrics with inheritance and CK metrics without inheritance have been observed. A Naïve Bayes algorithm has used for SFP, and accuracy has been mainly observed to know how efficiently the Naive Bayes is suitable and to say that whether the inheritance metrics usage is necessary or not.

Key words: Ck metrics, Inheritance, Inheritance Metrics, Naïve Bayes algorithm

1. INTRODUCTION

Software testing plays a vital role in the quality of the software, as in this phase, we detect the bugs present in the software. A bug is a fault or a failure that diverts the output of the project from the actual output to be produced. If bugs are found after the deployment of the software, it is tough to recover the faults [2]. So software tester should test the software project in order to predict the defects. In INDIA, cost of software development as of 2018 ranges from $13-50 USD per hour. Approximately 29% of costs for the testing of software projects. If there are some faults after the deployment, it leads to more cost of the project. So there is a considerable need to predict the faults before the deployment of the software.

To develop any software project, we use certain concepts of the programming language. These days the use of OO concepts has increased a lot. In that especially, the inheritance concept is employed very much due to its ability to provide maintainability, reusability, extendibility, and overriding. In order to predict the faults in case of object-oriented metrics, we use a suite of metrics called CK metrics as it supports the OO concepts. CK metrics consist of the following DIT, WMC, NOC, CBO, and RFC. In this paper, we are focusing on inheritance property because of the advantages stated above. This inheritance concept can be checked by using inheritance metrics. Inheritance metrics have the following metrics like Fan-in, Fan-out, NOAI, NOMI, IC, and MFA [11] [12].
In this, we are going to predict the software faults and examining whether the accuracy is more with inheritance metrics and without inheritance metrics and also to say to software testers whether to use the inheritance metrics with CK metrics or not. Many experiments have been conducted for this purpose by using datasets that are having the

above-stated metrics. In our experiment, we have used some datasets that are downloaded from the Zenodo website [1], which are having the above metrics. These datasets can also be downloaded from a tera-promise repository. For this experiment, different types of Machine Learning algorithms can be used, and we have chosen the Naïve Bayes Algorithm. We can say this efficient use of inheritance metrics by calculating the Accuracy, Precision, Recall, and F1score of these datasets.

In the following sections, we are going to explain. In section-II, we are describing CK metrics, Inheritance metrics, Concept of Inheritance, and Naïve Bayes algorithm. In section-III, we explain the working of the experiment that is filtering and dividing of the datasets and explanation of the code to implement for our experiment. In the section-IV, the coding algorithm will be explained. In section-V, the results of the code have been explained. As well in the following section-VI and VII, the conclusion and future work have been mentioned. [5][6]

2. RELATED WORKS

2.1 Description of CK Metrics

CK metrics is called as the Chidamber and Kemerer Object-Oriented metrics suited for examining the OO-concepts. As stated above, these metrics consist of six different metrics, which are explained, and in the following, where each metric has its own explanation and importance. [3][8]

1) WMC *(*weighted methods per class*):*
   WMC is defined as how many methods that a class consists of.

2) DIT *(*Depth of inheritance tree*):*
   DIT is the maximum length from node to the root of the tree in the hierarchy.
3) NOC *(*Number of children*):*
   NOC is the total sum of immediate sub-classes from the down of the base class or the main class that are to be derived.
4) CBO *(*Coupling between the object classes*):*
   CBO is to know the digits of classes are connected.

5) RFC *(*Response for a class*):*

   It is defined as the total methods in a class, and that is called by any other method.

6) LCOM *(*Lack of cohesion of methods*):*

   It always calculates the non- similar methods in a class.

2.2 Concept of Inheritance

Inheritance is a property of the java programming where it is defined as gaining the methods, variables from base class to the sub-class. This inheritance is divided as SINGLE, MULTI-LEVEL, MULTIPLE, HIRARICAL, and HYBRID INHERITANCE. And these types of inheritance are used based on our requirements. These different types are explained in the down stated as

1) Single Inheritance: It is defined as the classes inherit properties from only one superclass or parent class.

2) Multi-level Inheritance: A mechanism in which the subclass inherits the properties from a derived class.

3) Multiple Inheritance: A mechanism in which a subclass inherits from one or more subclasses.

4) Hierarchical Inheritance: In this, the main class is inherited by many secondary classes.

5) Hybrid Inheritance: The combination of multiple inheritances and multi-level inheritance is called hybrid inheritance.

2.3 Inheritance Metrics

Inheritance metrics are the metrics that can help to test if the presence of inheritance metrics gives the more accuracy or absence of inheritance metrics gives more accuracy, and these metrics are stated as below: [2].

1) Fan In – Super subordinate modules

2) Fan out- Immediate subordinate modules

3) NOAI - A count of attributes that are inherited

4) NOMI - Indicates methods that are inherited

5) IC - Inherited classes

2.4 Naïve Bayes Classifier [4]

It is one of the technique or algorithm to predict true or false conditions problems that is a supervised learning algorithm. It is mainly used for text classification and for performing predictions of faults [6][7]. The main reason for using this algorithm is that it is easy to implement and also requires less training data and can also handle missing values requires less time to implement it also can be used for large attributes are present we can update this algorithm as and when required easy to understand [9] [10] [13] [14].

3. METHODOLOGY

The following steps have been taken into consideration to know the accuracy, precision, recall, and F1score.

a) We collected the datasets from the zenodo website that are having both CK and inheritance metrics. these datasets are available on other websites such as tera promise repository and some of them are available even in websites where we can find other commonly used machine learning datasets such as:

b) After collecting the datasets, we removed the datasets that are having additional metrics. These additional metrics are not required for performing the experiment and are hence considered unnecessary and were removed during filtering of the datasets. Thus keeping only those that are required for the experiment.

c) We removed the duplicate values from the dataset with the help of predefined options that are available in Microsoft excel and also verified for redundancy in the datasets. Because if the datasets are not having enough redundancy then they may produce incorrect results.

d) Now next comes the division phase.as the main aim of this project is to know the impact of impact of inheritance metrics on software fault prediction we divided each dataset into 2 further datasets with inheritance and with CK metrics as type 1 and the other case as type 2.

e)Now that the datasets division phase is finished , we need some language to code and conduct the project. We have choosen R language to apply naïve Bayes on the datasets and calculate the accuracy, recall, precision, and F1score values and plotted the graphs for those values to show a clear comparison of the results. The main reason behind choosing R language for this purpose is that it is easy to implement and also provides some built in functions that are required to implement the NB algorithm, the coding part also becomes easy as we need not write the functions separately just have to import the libraries and install them and then can use them as and when required, also the changes can be made easily without any complexity.

.

4. ALGORITHM

It is a kind of classifier which uses Bayes theorem. It predicts membership probabilities for each class such as the probability that given record or data point belongs to a particular class the class with the higher probability is considered as the most likely class. Assumes that all the features are related to each other presence or absence of one feature does not influence the other features. the main disadvantage of this NB is that it is it can be used only for text classification. So for performing this experiment we have converted the o's to true and which are greater than 0 to false.

1. Load the dataset and read it using read.csv
2. Print the structure by str() function and find its dimensions with dim()
3. Divide the data for training and testing and print the training and testing data
4. Print their dimensions
5. Install the required packages.The required packages are caret and e1071using install.packages().then load the libraries using library()
6. Use the NB algorithm here
7. Print its summary using summary()
8. Print the summary and draw the graph for the same.the command used for drawing the graphs are plot()
9. Calculate the confusion matrix.this confusion matrix gives the required values of accuracy,f1 score,recall and precision.
10.the confusion matrix is printed using cm command
11. Apply accuracy, recall, precision, and F1 score values by applying the formulas and compare the values.

5. RESULTS AND ANALYSIS

This is an important section of a research paper since the experimental results will be discussed and a clear analysis will be given on them. Initially, we have started with 45 datasets that are having some unwanted metrics duplicate values, then applied required operations in ms excel and divided each of the datasets into two datasets as a result of filtration. After applying the code for the required values, it is observed that for some datasets, the values of precision, recall, and f1 has exceeded one which is not a good sign since the values should strictly lie between 0 and 1. Hence these datasets were also removed. As a result, 24 datasets remained and these are now each single dataset is divided into 2.comparision is made in order to decide which method works best. It is learnt that the existing one is more accurate than proposed NB.

Table 1: Experimental Results [4]

DATASET NAME	Inheritance +CK				$CK^{-Inheritance}$			
	Accuracy	Precision	Recall	F1score	Accuracy	Precision	Recall	F1score
Arc	0.125	0.5434783	0.5222727 3	0.1234783	0.75	0.6666667	0.6666667	0.6666667
Berek	1	1	1	1	1	1	1	1
Camel-1.6	0.7938144	0.6565934	0.5555195	0.5558608	0.7835052	0.6967391	0.5548485	0.5482369
Ckjm	1	1	1	1	1	1	1	1
Forrest-0.7	1	1	1	1	1	1	1	1

Log4j-1.0	0.9285714	0.9545455	0.875	0.9047619	0.8571429	0.9	0.8333333	0.8444444
Log4j-1.1	1	1	1	1	0.9090909	0.9375	0.875	0.8952381
Lucene-2.4	0.6764706	0.8103448	0.656625	0.621074	0.6176471	0.7678571	0.6578947	0.5888372
Poi-1.5	0.6666667	0.6888889	0.6783217	0.6643357	0.375	0.5	0.5	0.3466425
Poi-2.5	0.4102564	0.5290176	0.5192308	0.3959596	0.5128205	0.6444444	0.6553846	0.5076412
Prop-2	0.8272138	0.6117707	0.5892727	0.5978382	0.8195164	0.5894647	0.5485515	0.5547096
Prop-3	0.8267254	0.5794384	0.5383219	0.5418377	0.8189233	0.5369276	0.5167198	0.5124917
Prop-4	0.8678261	0.6202216	0.5559261	0.5681135	0.8483685	0.6316282	0.5682216	0.5816537
Prop-5	0.7786116	0.5521739	0.5132422	0.4898611	0.7802691	0.5826128	0.5269153	0.5144419
Prop-6	0.15	0.5446429	0.5363636	0.1497638	0.8	0.5142431	0.5185185	0.5148248
Serapion	0.8	0.875	0.75	0.7619048	1	1	1	1
Sklebagd	1	1	1	1	0.5	0.5	1	0.6666667
Szybkafucha	0.6666666	0.75	0.75	0.6666667	0.3333333	0.3333	0.75	0.45
	7					333		
Velocity-1.5	0.2727273	0.5789474	0.5789474	0.2727273	0.1818182	0.5714286	0.5263158	0.175
Velocity-1.6	0.6521739	0.5833333	0.5291667	0.4888889	0.8695652	0.9285714	0.7	0.747252
workflow	0.75	0.8333333	0.75	0.7333333	0.75	0.8333333	0.75	0.7333333
wspomaganiepi	1	1	1	1	0.5	0.5	1	0.6666667
Xalan-2.4	0.8493151	0.6612554	0.6182524	0.6338349	0.8219178	0.5858095	0.5252732	0.5170483
Xalan-2.7	0.6263736	0.5142857	0.8111111	0.4113394	0.3626374	0.7844828	0.3626374	0.4876055
Xerces-1.3	0.9130435	0.7984496	0.6878049	0.7261905	0.8478261	0.5634146	0.5773781	0.5689424
Xerces-init	0.5882353	0.75	0.65	0.5641026	0.4705882	0.4833333	0.4930556	0.3952569333333
Zuzel	1	1	1	1	1	1	1	1
Sum	20.1696918	20.4356514	19.6654599	17.87187	19.5099701	19.054256	19.1066754	1.964199333
average	0.7470256 2	0.7568759 8	0.7283503 7	0.6619212 2	0.7225914 9	0.7057131 9	0.7076546 4	0.65473311
median	0.8	0.75	0.6783217	0.6338349	0.8	0.6444444	0.6578947	0.5816537

From the above table 1, that is obtained as result of filtering the datasets by removing the duplicate values and unnecessary metrics. this has 28 datasets and also the results of sum, standard deviation, median and average values of the 4 parameters that are being calculated. the table is divided into 3 parts 1 part is for denoting names of datasets, second part is for datasets with both the metrics whereas the other part is for those remaining datasets that have ck metrics alone.

5.1 Figures of The Outputs

These pictures show the results of accuracy for one of the dataset prop-4, which indicates the accuracy of 87%, and the precision, recall, and f1score values are approximately 0.61, 0.56, and 0.57, respectively.

```
> accuracy=sum(diag)/n
> accuracy
[1] 0.8713043
```

Figure 1: This is the output of the accuracy

Above figure1 is obtained by calculating sum of diagonal values in the confusion matrix and then dividing it by total values 'n' and can see that the result thus obtained is 81%.

```
> macroPrecision = mean(precision)
> macroRecall = mean(recall)
> macroF1 = mean(f1)
> data.frame(macroPrecision,macroRecall,macroF1)
  macroPrecision macroRecall  macroF1
1      0.6186962   0.5658172 0.579479
```

Figure 2:A result of values macroprecison,macrorecall,macrof1

The above figure2 is the values of precision, recall, and f1score of each individual class.after finding these values they are printed using data.frame function.

5.2 Graphical Representation:

The corresponding graphs of the four values are as follows

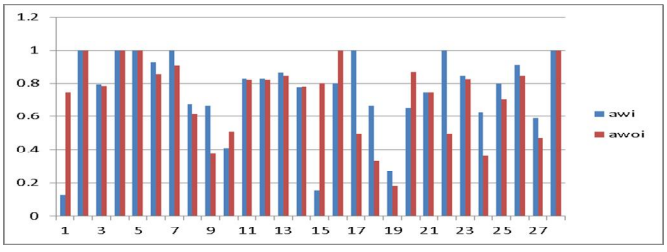

Figure 1: Plot of individual dataset accuracy of the two different types of datasets.

From the above figure 1 red color indicates without inheritance datasets and blue color indicates datasets with inheritance. values are almost similar in both cases.

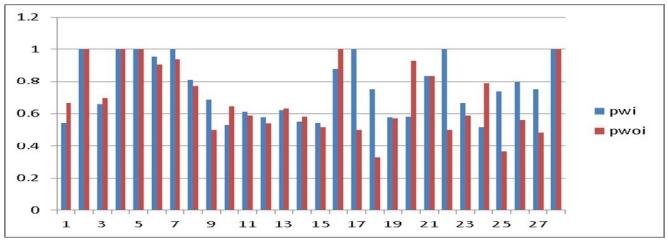

Figure 2: Graph of single dataset of precision values.

From above figure 2 we can describe result as red is for precision without inheritance and the other is for with inheritance. we can see that almost all the datasets are having same values irrespective of metrics only values are varying with respect to metrics.

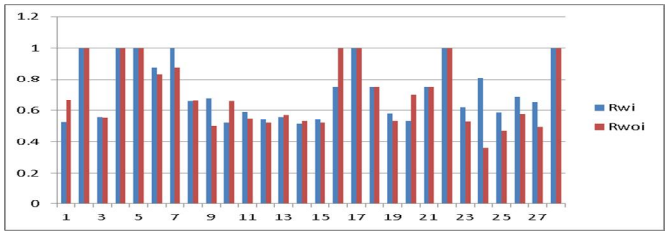

Figure 3: Diagram of individual dataset of recall values

From the above figure 3 .we can clearly observe that the values are varying with respect to metrics unlike precision. They are dependent on presence or absence of metrics.

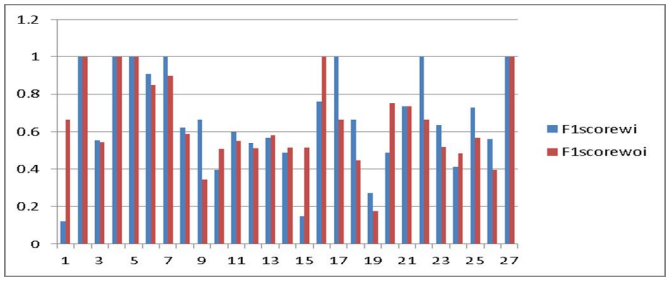

Figure 4: A plot of datasets for F1score values.

From figure 4 we can say that datasets 15th, 16th, are showing a clear variation. The datasets with inheritance have more f1 value than without f1.whereas 19th, 20th datasets are showing quite opposite results.

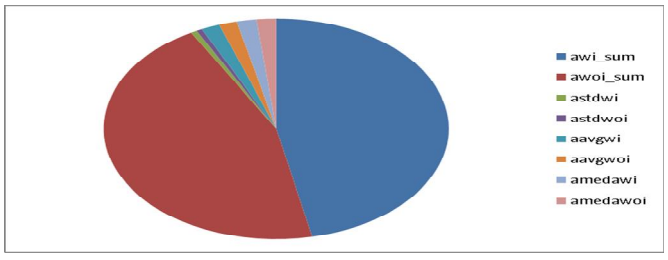

Figure 5: A pie graph of sum,standard deviation,average and median values of the accuracy.

From above figure 5 for the.sum value without inheritance is very high compared to others.

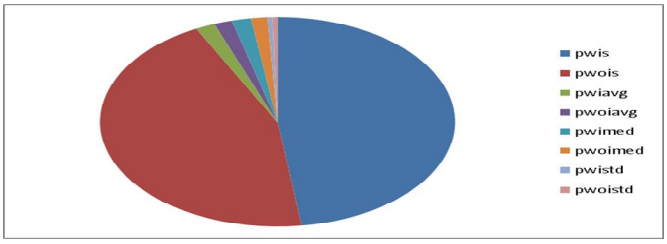

Figure 6: A similar pie graph for precision .

From the above figure 6 it can be seen that the sum and standard deviation are having almost the same values.

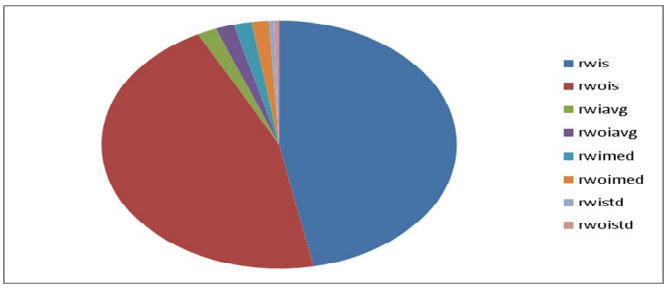

Figure 7: similarly the pie graph is drawn for recall .

From the figure 7 we can learn that sum without inheritance and with inheritance share almost the same part of the graph.

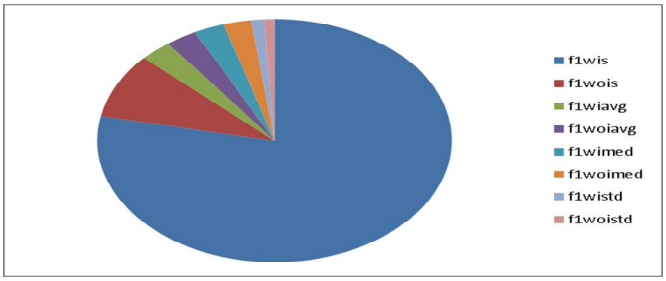

Figure 8: Pie graph for f1 score.

From the above result (figure 8) it is seen that the case is a bit different here we can see that only sum value occupies the most part of pie graph and the remaining part is shared by the other values.

6. CONCLUSION

In this paper we used the NB algorithm since it is suggested in the base paper that we can either improve the performance of already existing algorithm or by also can apply different ML techniques. We have chosen this technique to calculate and compare the values of the 4 parameters being measured and found out that the existing algorithm and NB show almost the same result but they vary only 2% from each other in terms of accuracy .Similarly we would suggest that a new ML technique can be applied for the same datasets or same algorithm with different one's. And compare the results.

ACKNOWLEDGEMENT

We would like to thank our guide Dr. S. Srinivasa Rao, for guiding us throughout the project and our HOD MR. Hari Kiran Vege. Finally, the University for providing the facilities required to finish the project.

REFERENCES

1. Datasets available at https://zenodo.org/
2. S. R. Aziz, T. Khan and A. Nadeem, "**Experimental Validation of Inheritance Metrics Impact on Software Fault Prediction**," in IEEE Access, vol. 7, 2019,pp. 85262-85275. https://doi.org/10.1109/ACCESS.2019.2924040
3. CKmetrics-https://sdlab.naist.jp/members/camargo/presentations/CKMetrics.ppt
4. Arasteh, Bahman, **Software Fault Prediction Using Combination Of Neural Network And Naïve Bayes Algorithm**, Journal Of Networking Technology Volume 9, No 3, 2018. https://doi.org/10.6025/jnt/2018/9/3/94-101
5. Srinivasarao Sabineni and Dr.Kurra Rajashekhara Rao, "**A comparative Study On Reliability And Cost Estimation For Architecture Based Software",** International Journal Of Applied Engineering Research, ISSN:0973-4562,Volume 10,No 8 May 2015,pp 19729-19736.
6. Srinivasarao Sabineni and Dr.Kurra Rajashekhara Rao, **Estimation of Reliability Allocation on components Using a Dynamic Programming**, International Journal Of Computer science Issues, ISSN: 1694-O814,volume 10,Issue 3,No 1 ,2013,pp.78-81, May
7. Manoj D. Shelar, S. Srinivasa Rao, **Malicious Threats Detection of Executable File** International Journal Of Innovative Technology and Exploring Engineering,2020, Volume-9, Issue-3, ISSN: 2278-3075 ,pp.3257-3262. https://doi.org/10.35940/ijitee.C8918.019320
8. Sabbineni Srinivas Rao, Inuganti Nava Sahitha, Godithi Sireesha, Palem Manoj **Evaluating Software System Reliability Using Architecture Based Approach** International Journal of Intelligent Information Systems, 2018;7(1):1-4 https://doi.org/10.11648/j.ijiis.20180701.11
9. Kuchibhotla,S., Bonthu,N.R., Kavuri,B.S.K., Datla, P.K. **Autism detection and subgrouping using machine learning algorithms**, International Journal Of Emerging Trends in Engineering Research,2019. https://doi.org/10.30534/ijeter/2019/407112019
10. Bhanu Prakash Dudi, Dr.V.Rajesh **Medicinal Plant Recognition Based On CNN And Machine Learning**, International Journal Of Advanced Trends In Computer Science And Engineering, ISSN:2278-3091,Volume 8,No 4,2019, pp August . https://doi.org/10.30534/ijatcse/2019/03842019

11. Calving ng, Alvin Chua. **Training Of A Deep Learning Algorithm For Quadcopter Gesture Recognition**, International Journal Of Advanced Trends In Computer Science And Engineering, ISSN:2278-3091,Volume 9,No 1,2020,pp January to February . https://doi.org/10.30534/ijatcse/2020/32912020

12. Praveen Kumar Kollu, R.Satya Prasad **Effective Intrusion Detection Using Deep Recurrent Neural Networks**, International Journal Of Advanced Trends In Computer Science And Engineering, Volume 8,no 4,2019,pp July To August.

13. Mothe, Saiteja. (2020). "**A Model for Assessing the Nature of Car Crashes using Convolutional Neural Networks**." International Journal of Emerging Trends in Engineering Research. 8. 859-863. doi:10.30534/ijeter/2020/41832020

14. Mothe, Saiteja & Tata, Ravi Kumar. (2020). "**Load Balancing Analyzer: A Recommendation System using Machine Learning**". International Journal of Emerging Trends in Engineering Research. 8. 2085-2090. doi:10.30534/ijeter/2020/99852020.