Deductive learning approaches for Face Recognition by using Gabor Feature

* M.Shanmuganathan [1], T.Nalini [2]

[1] "Research Scholar., Dept of C.S.E, Bharath Institute of Higher Education and Research(BIHER)",
"Chennai"-600073, Tamil Nadu, India.
[1*]shanmca75@gmail.com

[2] "Professor., Dept of C.S.E, Dr. M.G.R. Educational and Research Institute,
"Chennai-600095, Tamil Nadu, India".
[2]drnalinichidambaram@gmail.com
* "Corresponding Author"

Abstract

Artificial Intelligence (AI) offers a vital role in refining the well- being of human in many areas. Recently in the field of face recognition, these have some limitations, exclusively when faces are in different poses or have different levels of illumination, or when the face is blurred. This research focuses to produce the optimal solutions for the face recognition. The parameter k is 6 the model produces the highest accuracy value which is 84.67% accuracy level. The highest precision value is 84.67% while applying the parameter k is 1.The highest recall value is 86.68% which is produced by while applying the parameter k=6, the lowest recall value is 68.77% while applying the parameter k=9. The model takes more time to build the model when applying the parameter k=10 and very low time consumption model is k=8 which is recommended this system for optimal solutions.

Keywords: Weka, Gabor filter, K NN, ARRF format, FEP.

I INTRODUCTION

In this section presents introduction of this research work. Inadequate software testing was estimated to cost the United States economy alone $59 billion in the year 2002. Software testing costs have been estimated by various researchers as up to 50% of total project cost with regression test cost being one-third of the overall testing cost. [1, 2]Given the importance of regression testing, it is not surprising that academic researchers have overwhelmingly focused on test case minimization, selection, and prioritization. As pointed out by Antonio Bertolino software testing research has become almost synonymous with these approaches [3,4]. However, the uptake of these methods by industry has been very limited and exploratory testing approaches are still preferred .

Test case minimization uses heuristics to reject unnecessary test cases across all versions of the software under test (SUT), while the selection problem deals with methods to minimize the test suite size for a particular version of the SUT. A key difference between the above two methods is that test case minimization attempts to once and for all minimize the test suite size (across all versions) and is an NP-complete issue requiring heuristic methods. The reduction is done keeping in mind certain criteria of "test adequacy" like statement coverage or additional statement coverage adequacy, branch coverage or additional branch coverage adequacy, etc. with methods such as the Graph-Walk approach, program slicing, etc. In the literature both minimization and selection are subsumed under the category of test case selection [5,6,7,8]. These methods can also have drawbacks ---for e.g., minimization can adversely affect the ability to detect faults [9,10,11].

Test case prioritization is involved with sequencing or ordering test cases such that some test performance criteria are better met, for e.g., detect faults at a faster rate as compared to the unordered original set using methods such as hill-climbing algorithm, genetic algorithm, greedy algorithm, etc. using coverage or FEP (Fault Exposing Potential) criteria .

In this paper presents section 2 of this paper explains the detail on the related works. In section 3 presents the materials and methods adopted and section 4 presents the details of the experiments and discussions. Finally section 5 concludes the paper by sharing our inferences and future plans.

II RELATED WORKS

In this section presents focuses the related works of this research work. A limitation of research in regression testing is that the majority of empirical studies have been done with programs less than 10K LoC in size and test suites are also correspondingly small with less than 1,000 test cases. A further limitation is that a majority of empirical studies draw upon a single repository, viz. the SIR. The number of empirical evaluations of various approaches is still very limited, although recent years have seen greater focus in this area. Also, since many of the approaches especially in Test case minimization rely on heuristic methods such as the greedy algorithm, there is no basis to state that anyone's approach is better than another[12,13,14].

Although research in these areas dates back to 1977, it is only recently that attention is being focused on not only the benefits such as improved test results according to certain criteria but also the costs involved in adopting these procedures. Execution time was one of the first cost parameters to be studied [15,16]. Harman recommends the inclusion of other criteria such as Data Access Costs, Human Sensitive Factors (subjective preferences of customers, managers, testers, etc.), Business Sensitive Factors (test features related to revenue or market share potential), criteria based on the relevance of fault models, etc.. However, studies have not yet factored in the cost of developing Test Oracles [17,18,19,20,21].

Given the very many criteria listed above along with other test adequacy criteria, it is tempting to model the choice of regression testing approaches related to testing case inclusion or exclusion as the MCDM (multi-criteria decision-making problem) or even a multi-criteria problem with multiple objectives. Indeed, several researchers take precisely such a route to tackle the complexity of the problem [1,8,17,18]. However, at present, there are no guiding principles to help select a more efficient regression test selection procedure.

The MCDM (Multi-Criteria Decision-Making) methodology, addressed by T.L. Saaty the Analytical Hierarchy Process, is one such method used by several researchers. Here a complex issue is reduced into a level hierarchy consisting of objective, criteria, sub-criteria, along with alternatives. This methodology identifies a set of matrices and gives priority by comparing requirements and alternatives together. While it is widely used, several facets of AHP appear to be contentious, in nature. The issue of reverse ranks in general if an alternative is introduced or removed has been highlighted [2,9,12,13].

III MATERIALS AND METHODS

In this section presents the materials and methods of this research work. In this research using CMU Face Images Data Set which is borrowed from UCI repository. This data consists of 640 black and white face images of people. This below table clearly shows that the specification of this image dataset.

Table 1: Specifications of Images

S.No	Category	Specifications
1	Pose	straight
		left
		right
		up
2	Expressions	neutral
		happy
		sad
		angry
3	Eyes	Wearing sunglasses
		Not Wearing sunglasses
4	Size	Different

- In this research implements in Weka 3.8.1. It has 10 different image filters are presented.
- Here apply the filter to be applied for transforming the input image.
- These filters generate set of features in numerical format and presented in the ARFF format.
- The data can be used for further data pre-processing or classification.
- **KNN**: This instance based nearest neighborhood algorithm is implemented as KNN and is found in Lazy category classifier

IV RESULTS AND DISCUSSIONS

In this section focuses the results and discussions of this research work. The below table clearly demonstrates that the Accuracy levels of K- Nearest Neighbor classifier.

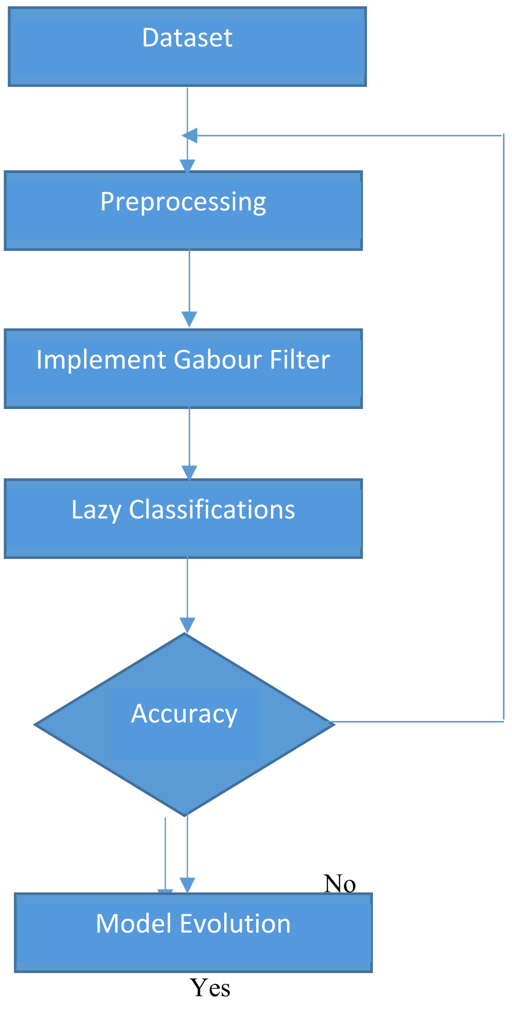

Figure 1: Proposed System

The above diagram clearly represents that the work flow of the proposed system to get the optimal solutions in the given face images dataset.

Table 2: Instance based classifier Vs Accuracy

S.No	K	Accuracy
1	1	82.48%
2	2	82.43%
3	3	77.80%
4	4	80.42%
5	5	76.68%
6	6	84.67%
7	7	68.59%
8	8	79.18%
9	9	80.42%
10	10	76.68%

The above table clearly shows that the instance based classifier is producing the accuracy levels are different while doing the parameter tuning like K parameter. In this dataset we implemented the lazy classifier while K=1 then the system gets an accuracy value is 82.48%, while K=2 then the system gets an accuracy value is 82.43%, while K=3 then the system gets an accuracy value is 77.80%, while K=4 then the system gets an accuracy value is 80.42%, while K=5 then the system gets an accuracy value is 76.68%, while K=6 then the system gets an accuracy value is 84.67%, while K=7 then the system gets an accuracy value is 68.59%, while K=8 then the system gets an accuracy value is 79.18%, while K=9 then the system gets an accuracy value is 80.42%, while K=10 then the system gets an accuracy value is 76.68%.

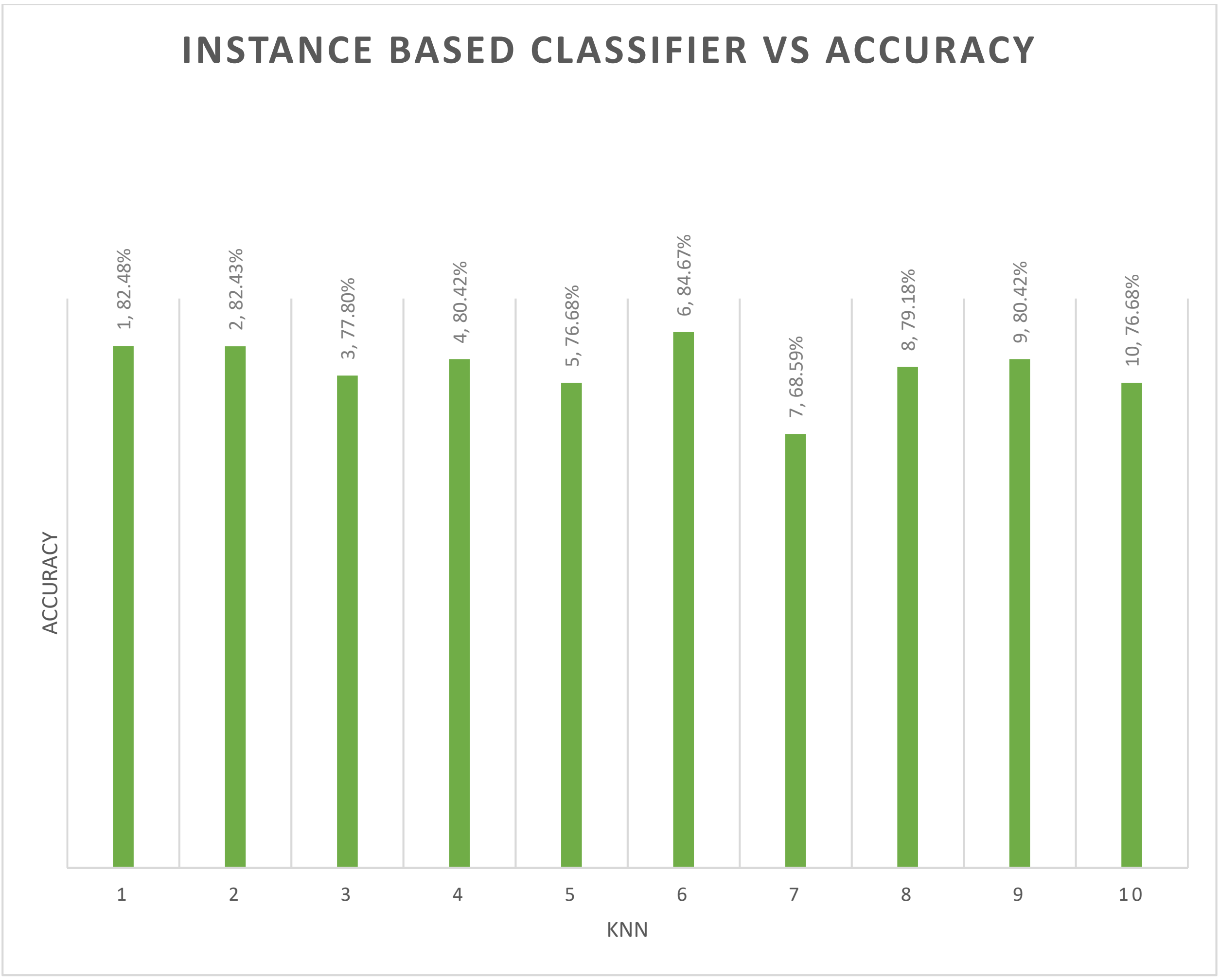

Figure 2: Instance based classifier Vs Accuracy

The above diagram shows that the highest accuracy value is 84.67% which is while applying the parameter k=6 and the lowest accuracy value is 68.59% when apply the k=7

Table 3: Instance based classifier Vs Precision

S.No	K	Precision
1	1	84.67%
2	2	76.68%
3	3	81.13%
4	4	82.31%
5	5	82.48%
6	6	83.43%
7	7	80.79%
8	8	80.42%
9	9	76.68%
10	10	77.80%

The above table clearly shows that the instance based classifier is producing the precision value levels are different while doing the parameter tuning like K parameter. In this dataset we implemented the lazy classifier while K=1 then the system gets a precision value is 84.67%, while K=2 then the system gets a precision value is 76.68%, while K=3 then the system gets a precision value is 81.13%, while K=4 then the system gets a precision value is 82.31%, while K=5 then the system gets a precision value is 82.48%, while K=6 then the system gets a precision value is 83.43%, while K=7 then the system gets a precision value is 80.79%, while K=8 then the system gets a precision value is 80.42%, while K=9 then the system gets a precision value is 76.68%, while K=10 then the system gets a precision value is 77.80%.

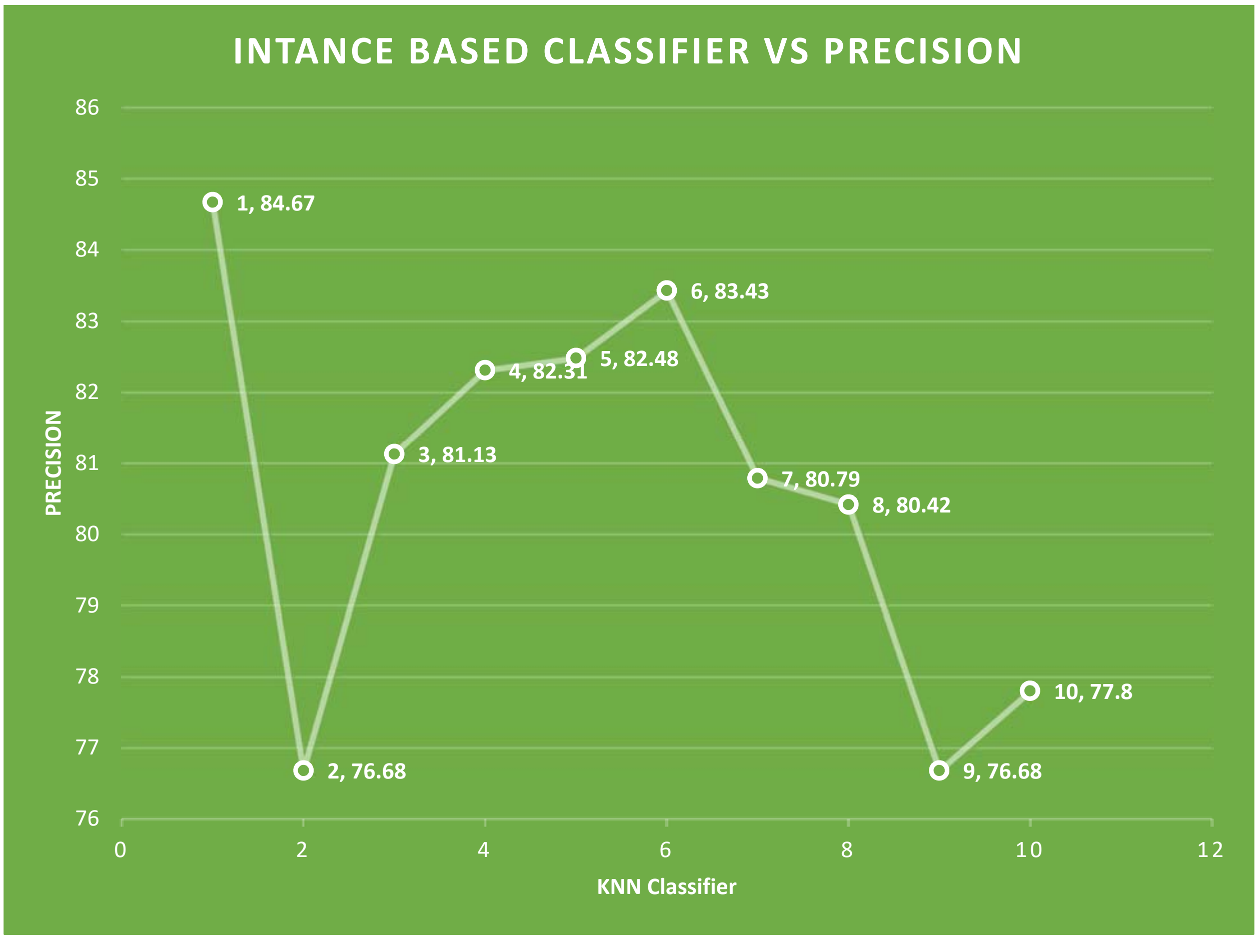

Figure 3: Instance based classifier Vs Precision

The above diagram clearly shows that the highest precision value is 84.67% while applying the parameter k=1 and very lowest precision value is 76.68% lies on k=2 and k=9.

Table 4: Instance based classifier Vs Recall

S.No	K	Recall
1	1	82.42%
2	2	74.68%
3	3	83.67%
4	4	78.59%
5	5	78.18%
6	6	86.68%
7	7	77.79%
8	8	84.66%
9	9	68.77%
10	10	75.88%

The above table clearly shows that the instance based classifier is producing the recall value levels are different while doing the parameter tuning like K parameter. In this dataset we implemented the lazy classifier while K=1 then the system gets a recall value is 82.42%, while K=2 then the system gets a recall value is 74.68%, while K=3 then the system gets a recall value is 83.67%, while K=4 then the system gets a recall value is 78.59%, while K=5 then the system gets a recall value is 78.18%, while K=6 then the system gets a recall value is 86.68%, while K=7 then the system gets a recall value is 77.79%, while K=8 then the system gets a recall value is 84.66%, while K=9 then the system gets a recall value is 68.77%, while K=10 then the system gets a recall value is 78.88%.

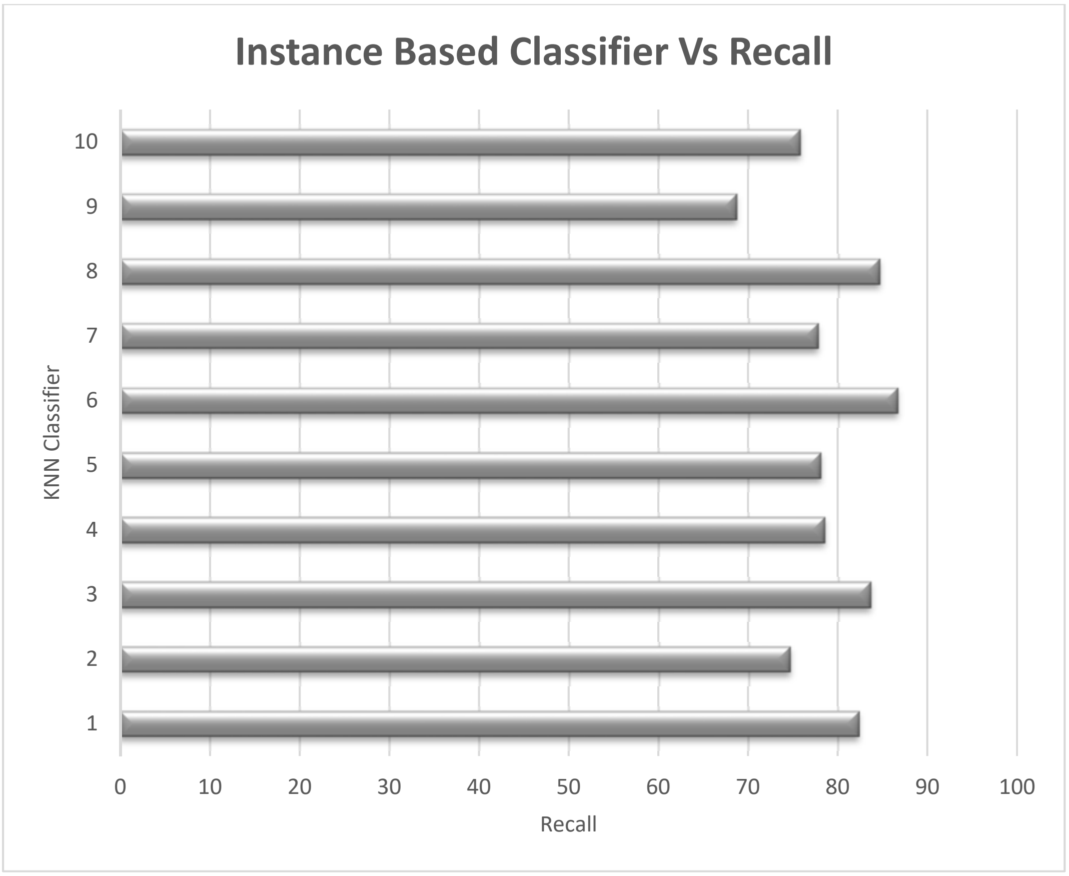

Figure 4: Instance based classifier Vs Recall

The above diagram the highest recall value is 86.69% which is produced by while applying the parameter k=6, the lowest recall value is 68.76% while applying the parameter k=9.

Table 5: Instance based classifier Vs Time

K	Time taken to build model(In Seconds)
1	0.16
2	0.18
3	0.46
4	0.22
5	0.20
6	0.88
7	0.21
8	0.11
9	0.33
10	1.21

The above table clearly shows that the instance based classifier is taking the time consumption to build the model is different while doing the parameter tuning like K parameter. In this dataset we implement the lazy classifier while K=1 then the system takes a time consumption to build the model is 0.16 seconds, while K=2 then the system takes a time consumption to build the model is 0.18 seconds, while K=3 then the system takes a time consumption to build the model is 0.46 seconds, while K=4 then the system takes a time consumption to build the model is 0.22 seconds, while K=5 then the system takes a time consumption to build the model is 0.20 seconds , while K=6 then the system takes an a time consumption to build the model is 0.88 seconds, while K=7 then the system takes an a time consumption to build the model is 0.21 seconds, while K=8 then the system takes an a time consumption to build the model is 0.11 seconds, while K=9 then the system takes an a time consumption to build the model is 0.33 seconds, and finally while K=10 then the system takes an a time consumption to build the model is 1.21 seconds.

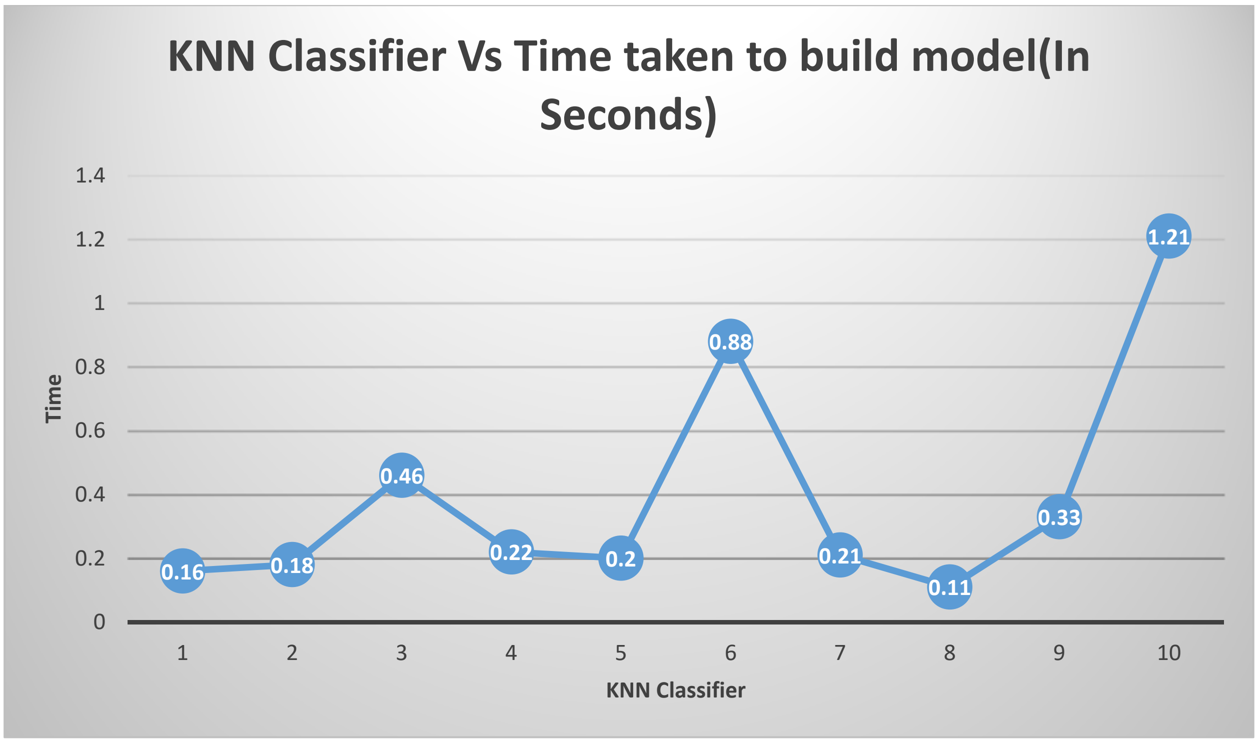

Figure 5: Instance based classifier Vs Time Consumption

The above diagram shows that the K=10 model takes more time to build the model is which is 1.21 seconds and when k=8 the model takes the less time consumption which is 0.11 seconds.

IV CONCLUSIONS

The System concludes that the when the parameter k is 6 the model produces the highest accuracy value which is 84.67% accuracy level. The highest precision value is 84.67% while applying the parameter k is 1.The highest recall value is 86.68% which is produced by while applying the parameter k=6, the lowest recall value is 68.77% while applying the parameter k=9. The model takes more time to build the model when applying the parameter k=10 and very low time consumption model is k=8 which is recommended this system for optimal solutions.

REFERENCES

[1] Vijayan T , Sangeetha M , A. Kumaravel , Karthik B , Gabor Filter and Machine learning Based Diabetic Retinopathy Analysis and Detection, Microprocessors and Microsystems (2020)
[2] Belton, V., & Gear, T. (1983). On a short-coming of Saaty's method of analytic hierarchies. Omega, 11(3), 228-230.
[3] Bertolino, A. (2003, March). Software testing research and practice. In International Workshop on Abstract State Machines (pp. 1-21). Springer, Berlin, Heidelberg.
[4] Donegan, H. A., Dodd, F. J., & McMaster, T. B. M. (1992). A new approach to AHP decision-making. Journal of the Royal Statistical Society: Series D (The Statistician), 41(3), 295-302.
[5] Elbaum, S., Malishevsky, A., &Rothermel, G. (2001, May). Incorporating varying test costs and fault severities into test case prioritization. In Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001 (pp. 329-338). IEEE.
[6] Engström, E., Runeson, P., &Skoglund, M. (2010). A systematic review of regression test selection techniques. Information and Software Technology, 52(1), 14-30.
[7] Engström, E., &Runeson, P. (2010, June). A qualitative survey of regression testing practices. In International Conference on Product-Focused Software Process Improvement (pp. 3-16). Springer, Berlin, Heidelberg.
[8] Harman, M. (2011, March).Making the case for MORTO: Multi-objective regression test optimization. In 2011 IEEE Fourth International Conference on Software Testing, Verification, and Validation Workshops (pp. 111-114). IEEE.

[9] Maleki, H., &Zahir, S. (2013). A comprehensive literature review of the rank reversal phenomenon in the analytic hierarchy process. Journal of Multi-Criteria Decision Analysis, 20(3-4), 141-155.
[10] Ayyappan.G et al. Knowledge Structure for Fraudulent Firm Classification Approaches, Indian Journal of Computer Science and Engineering (IJCSE), Volume No.10 Issue No.4 Aug-Sep 2019, Page No: 74-82, e-ISSN: 0976-5166, p-ISSN: 2231-3850.
[11] Rothermel, G., Harrold, M. J., Ostrin, J., & Hong, C. (1998, November). An empirical study of the effects of minimization on the fault detection capabilities of test suites.In Proceedings.International Conference on Software Maintenance (Cat. No. 98CB36272) (pp. 34-43). IEEE.
[12] Saaty, T. L. (2008). Decision making with the analytic hierarchy process. International journal of services sciences, 1(1), 83-98.
[13] Saaty, T. L. (2000). Fundamentals of decision making and priority theory with the analytic hierarchy process (Vol. 6). RWS publications.
[14] Triantaphyllou, E., & Mann, S. H. (1994). A computational evaluation of the original and revised analytic hierarchy process. Computers & industrial engineering, 26(3), 609-618.
[15] Whittaker, J. A. (2002). How to Break Software: A Practical Guide to Testing with Cdrom. Addison-Wesley Longman Publishing Co., Inc.
[16] Yoo, S., & Harman, M. (2007, July). Pareto efficient multi-objective test case selection.In Proceedings of the 2007 international symposium on Software testing and analysis (pp. 140-150).
[17] Yoo, S., Harman, M., Tonella, P., & Susi, A. (2009, July).Clustering test cases to achieve effective and scalable prioritization incorporating expert knowledge.In Proceedings of the eighteenth international symposium on Software testing and analysis (pp. 201-212).
[18] Harman, M., &Yoo, S. (2007).Regression-Testing Minimisation. Selection and Prioritisation: A Survey, software testing, verification, and reliability.
[19] Arafeen, M. J., & Do, H. (2011, November). Adaptive regression testing strategy: An empirical study. In 2011 IEEE 22nd International Symposium on Software Reliability Engineering (pp. 130-139). IEEE.
[20] Ayyappan.G et al. Novel Ensemble Approaches To Model Macroscopic Material Behavior Using Micromechanical Simulations, International Journal of Scientific & Technology research (IJSTR), Volume No.8, Issue 11, November 2019, Page No: 2560-2564, e-ISSN: 2277-8616.
[21] Rooksby, J., Rouncefield, M., &Sommerville, I. (2009). Testing in the wild: The social and organizational dimensions of real-world practice. Computer Supported Cooperative Work (CSCW), 18(5-6), 559.