Categorizing Code Review Comments using Machine Learning

Submitted By

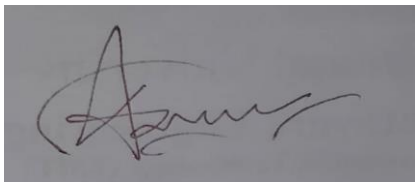
Yeasir Arafat
ID: 161-35-1501

Supervised By
Ms. Syeda Sumbul Hossain
Lecturer

Department of Software Engineering
Daffodil International University

DECLARATION

I hereby declare that I have taken this thesis under the supervision of **Ms. Syeda Sumbul Hossain**, **Lecturer**, **Department of Software Engineering**, **Daffodil International University**. I also declare that neither this thesis nor any part of it has been submitted elsewhere for award of any degree or diploma.



Yeasir Arafat
ID: 161-35-1501
Batch: 19th
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Certified by:

Ms. Syeda Sumbul Hossain
Lecturer
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Table of Contents

Abstract

Code review turns into a progressively mainstream method to find out early defects in source code. These days experts are going for peer-investigating their codes by their co-developers to make the source code clean. Chipping away at a circulated or scattered team, a code review is required to check the patches to consolidate. Code looking into can likewise be a structure of approving practical and non-useful necessities. In some cases, analysts don't put organized remarks, which turns into a bottleneck to developers for tackling the discoveries or recommendations remarked by the reviewers. For making the review support progressively successful, organized also, productive survey remarks are compulsory. Mining the repositories of five commercialized projects, we extricate 2185 review comments. We have utilized 6 machine learning classifiers to prepare our model. Among those Stochastic Gradient Descent (SGD) vector machines, the procedure accomplishes a higher exactness of 63.89%. This examination will assist the specialists with building up organized and viable code review by categorizing and culture among worldwide programming engineers.

Chapter 1: Introduction

In the worldwide software engineering period, code evaluating turns into an increasingly well-known method to find out early imperfections in source code. These days experts are going for peer-looking into their codes by their co-engineers to make the source code clean. Clean source code expands the decipherability of the code which is pivotal for worldwide software development. These days, software businesses are receiving agile software development methodology [44], and agile software development is done over the appropriated and scattered team when the task on an enormous scale. While chipping away at an appropriated or scattered task, each fix should be evaluated by different developers for reviewing, not just the source code just as to check if the fix meets the necessities or not. In some cases, analysts don't put organized remarks, which turns into a container neck to engineers for settling the discoveries or proposals remarked by the commentators. To make the review interest progressively successful, organized and productive survey remarks are compulsory.

In present-day code review [45], the tool-based review is getting increasingly prevalent. Distinctive tools [46,47] are utilizing for tool-based code review. An automated code review tool [48] is developed by Google. There are numerous apparatuses for evaluating source code, be that as it may, peer-exploring is one of the code quality confirmation activities other than tool-based reviewing [49]. In this manner, the source code should be investigated by human reviewers.

The purpose of this study is to help the developers to classify and prediction of review comments based on the analysis of software repositories. By the end of this case study, it is expected that the developer will get automated review categories. For this case study, we depict an intercession in a code-review classification using machine learning. In certifiable projects, code reviews a fundamental piece of our development lifecycle. Basically, it’s required huge time for brainstorming. A little issue is occurring on coding on projects dependent on some misconceptions and the real things, we took a shot at it.

In this examination, we present the successful order of AI procedures to find and anticipate the automated review of any corpus reviews. A code review is a typical practice received in software advancement to improve software quality dependent on static code examination by peers. Notwithstanding all these immense duties to the field of code review, it is basic to grasp the components that affect the practicality of code study. Not in any way like the standard code appraisals of the past, present-day code study is lightweight and adaptable.

To guarantee a significant level of software quality, Frequentis built up a code survey process and presented code review tooling into their advancement procedure to agree to the DO-278/ED-109 standard [22]. Its study provided evidence that reviewers are less rigorous and find fewer defects [23] on files with a high incidence of defects in the past, focusing on shallow perspectives, for instance, coding models rather than utilitarian viewpoints.

This paper is organized as pursues: Background study is done in Section 2 that pursued by Research Methodology and Result and Discussion at Section 3, Section 4 individually. The last Section 5 abridged our commitment and outfits the conclusion.

Chapter 2: Background Study

There is an enormous assemblage of literature on code reviews and investigations. Basically, Code review is a key tool for quality affirmation in software development[1]. During the procedure of statically assessing code, developers of a codebase can cooperatively identify conceivable code defects, just as use code reviews as a method for transferring information to improve the general comprehension of a system[2]. A few formal and less conventional audit approaches have been developed[24, 25]. One of the most legitimate review procedures is the item assessment was a review bunch takes a gander at a work thing in a couple of stages and each examiner fulfills a particular activity.

In Rigby's and Bird's recent study [3],they look at peer review procedures from a few projects and note assembly towards a typical procedure. This procedure is lightweight, adaptable and what's more, founded on the continuous review of little changes.
Stein et al. [26] and Mayer [27] worked on distributed asynchronous code reviews that are also properties of this work.

Regardless, manual code overview has a couple of shortcomings[28,29]: a) the opportunity has arrived consuming, b) results are passionate and depend upon the aptitudes of investigators. An electronic system for helping code reviews is as such incredibly alluring[30].

Utilizing machine learning in certifiable creation systems is entangled by the host of issues not found in little toy models or even enormous offline research experiments[4,5]. In light of long stretches of related knowledge utilizing Machine Learning at Google, in systems, for example, advertisement click forecast [6] and the Sibyl Machine Learning stage [7], they have developed a great deal of best practices for utilizing machine learning systems.

To support machine learning to construct evidence-based arguments[31], both types of legal annotation would be used to develop semantic-pragmatic analytics that receive input from a DeepQAsystem like Watson[32]. By developing the framework and machine learning model further in future we can get rid of the non-required audit cycle completely[34,35]. That burdens computer programs getting from information, is utilized to create check models which at that point can be utilized to total information[36].

The field of machine learning has been developing quickly, making a wide gathering of learning figurines for various applications[38]. Logical advancement consistently expands on existing distributions and strategies[38,39]. Different inquires about with respect to abscond models incorporate relapse models [40,41], factual models and AI-based models [42,43].

(DL) is also becoming popular in software repository mining research. For detecting redundant comments in code, [Louis et al.(2018)Louis, Dash, Barr, and Sutton] developed a tool using DL. They introduced a framework which can help the developers in writing better and informative programming comments. Using topic model and n-grams, [Movshovitz-Attias and

Cohen(2013)] presented a techniques to predict programming comments. In [White et al.(2016)White, Tufano, Vendome, and Poshyvanyk], they used DL in detecting code clone by mining software repositories.

Chapter 3: Research Methodology

In this segment, we briefly depict the approach of our methodology that we have embraced for the investigation. We are roused to structure our research by our earlier work [10]. So as to break down the review comments, at first, we have extricated code review comments from five popularized iOS projects. At that point, we marked the data for deciding suppositions of that reviews and categorized the gathered datasets. Subsequent to preprocessing the categorized datasets, we prepared our models. Figure 1 shows the general procedure of our research work.

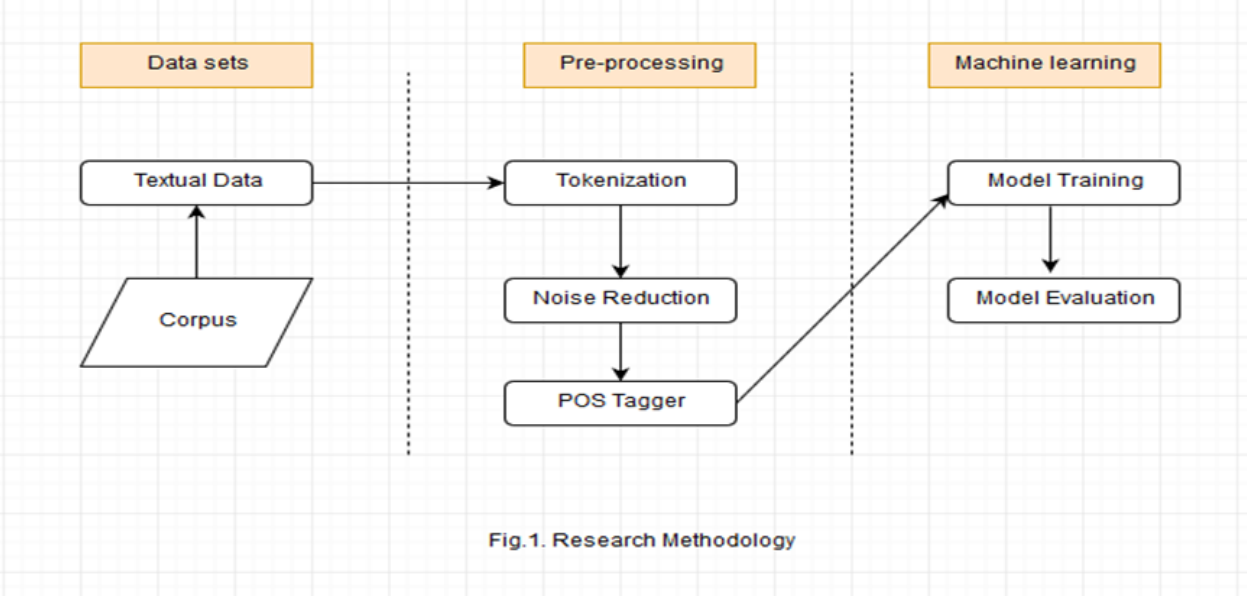


Fig.1. Research Methodology

3.1 Data Accumulation:

To accumulate information for our investigations, we have picked five promoted projects of an esteemed association. These five projects are extensively used and accept a critical role in their particular spaces. fundamentally, we use scripting systems as a data collection mechanism to creep the important reviews of these five projects. All projects are iOS App development extends and is kept up in Gerrit[37]. Utilizing the Gerrit API, we slithered the repositories and extricated the raw data. For extricating these data, we set the timeline for 6 months long. Table 1 is indicating the subtleties of the dataset extricated from the various data sources.

Projects Name	#No of Members	#No of Review Comments
Alpha	20	800
Beta	15	600
Gamma	10	400
Delta	4	185
Epsilon	3	200

Table 1: Summary of Datasets

3.2 Data Labeling**:**

Table II represents the classes of review comments that have a place with the various classification's dependent on our datasets. We decompose this categorization process from Bosu et al.(2015)bosu [11], they present the distribution of 13 categories of code review comments. That we utilized our data labeling process.

Category	**Description**
Documentation	Code structure & organization matter. Clean & consistent code.
Visual Representation	Make code more readable. Blank lines indentation.
Organization	How functionally is divided into methods.
Solution Approach	A problem solution approach. Dealing with key aspects of swift.
Resource	Reference or resource indentation.
Validation	Check char's, handle parameter, validation etc.
Logical	All about logic codes.
Synchronization	Dealing with thread, perform code & about properties.
Supporting Library	All About library programs.
Defect	Defect management & correction.

API Calls	All about relate with API.
False Positive	Using something, after that, it appears as unused.
Others	Question from reviewer. Praise for the implementation. Suggestion for a terminate output or error message. Discussion on design or new features.

Table 2: Categories of Reviews [11]

3.3 Preprocessing:

Corpus dataset gained from different sources regularly should be preprocessed before propelling a complete review comments classification. As a matter of first importance, we start from corpus textual data to tokenization. Then lemmatized the texts and removed the noise words utilizing noise reduction method.

3.3.1. Tokenization:

By and large, tokenization is utilized to break a sentence into words, expressions, symbols or other important tokens by expelling accentuation marks. We segment our text data into words using word-tokenize which is from NLTK library. We make sure all the short forms like don’t, she'll remain as one word.

“word_tokenize(str(review).lower()”

We use this method of code for tokenized our corpus datasets.

3.3.2 Lemmatization:

Lemmatization decreases the curved words guaranteeing that the root word has a place with the language. In lemmatization, the root word is called lemma. Utilizing Wordnet lemmatizer we query the lemmas of words. The grammatical form of a word is resolved in lemmatization like V as a verb word, A as an adjective word, N as a noun. Table 3 is giving some example lemma words from our dataset.

“review=[lemma.lemmatize(w,pos ='v')”

We use this method of code for lemmatized our corpus datasets.

Words	Lemma
Cryptography	Crypt
Reform	Form
Legalities	Legal
Normality	Norm
Claustrophobia	Phobia

Table 3: Lemmatization Process

3.3.3 Noise Reduction:

We have evacuated all the accentuation and regular expression from the content as these can't be utilized to investigate characterization by “re.sub” method of python programming language.

“review=re.sub('[^a-zA-Z]',' ',review)
review = re.sub('[0-9]+', '', review)
review = re.sub(r'[?|!|\'|"|#]',r'',review)
review = re.sub(r'[.|,|)|(|\|/]',r' ',review)
review = re.sub(r'http\S+', '', review, flags=re.MULTILINE)”

We use this method of code for noise reduction.

3.4 Training Model:

Like the prior examination [15], we get the quantifiable strategy of Harrell Jr. [16]. Whereas past work utilizes multiple linear regression models [15], we use logistic regression models considering the degree of sections with different post-release gives up are irrationally low for checking models or straight fits..For training our model, we have used 6 machine learning algorithms. Used algorithms are:

1. Multinomial Naive Bayes (MNB) [Rennie et al.(2003)Rennie, Shih, Teevan, and Karger].

2. Logistic Regression [Ho et al.(1994)Ho, Hull, and Srihari], Stochastic Gradient Descent (SGD) [Bottou(2010)],

3. Bernoulli Naive Bayes [McCallum et al.(1998)McCallum, Nigam, et al.],

4. Support Vector Machine(SVM) [Gunn et al.(1998)],

5. k-Nearest Neighbors (KNN) classifier [Scholkopf et al.(2000)Scholkopf, Smola, Williamson, and Bartlett], and

6. Stochastic Gradient Descent (SGD).

We have trained our review comment corpus data sets with our choosing train models. Figure 2 shows the training process of our models.

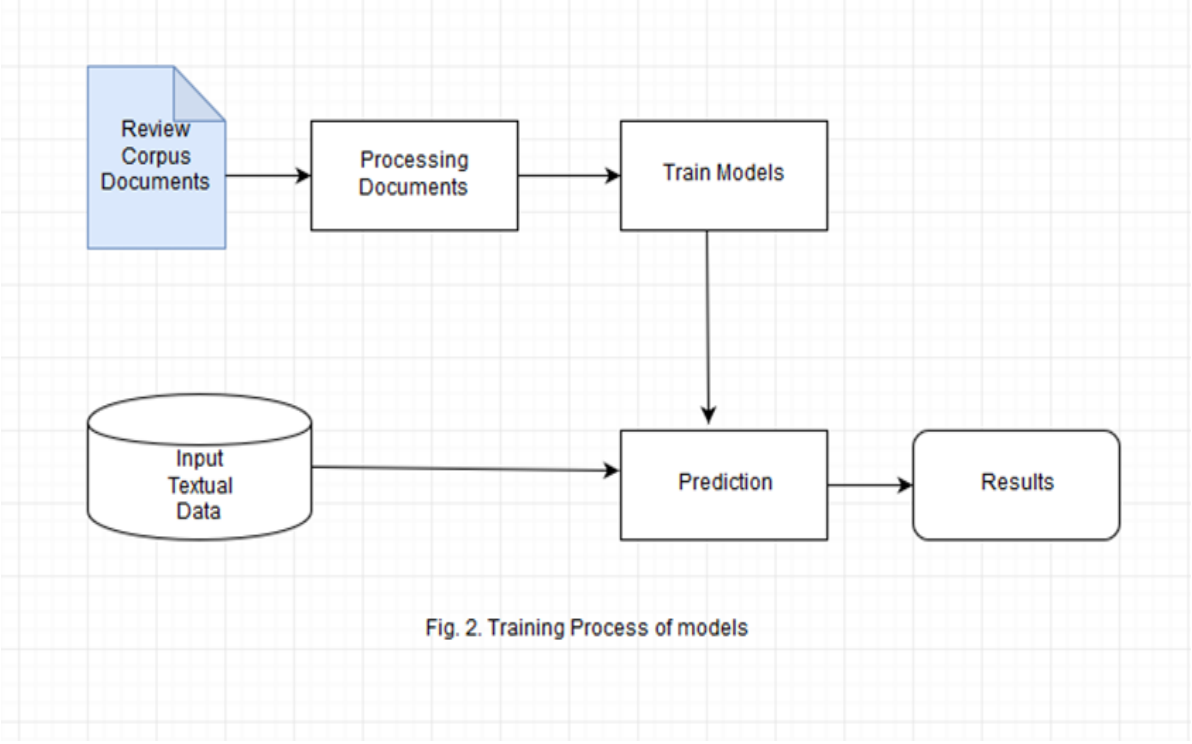


Fig. 2. Training Process of models

3.5 Cross Validation:

In our corpus datasets that we utilized to train the models, it contains 2185 lines of review comments(Documentation 82, Visual Representation 71, Organization 278, Solution Approach 128, Resource 3, Validation 73 , Logical 145, Synchronization 30, Supporting Library 2, Defect 35, API Calls 11, False Positive 5, Others 1322 lines of reviews). We assess our model utilizing k-fold cross validation where K-Fold esteem is 5 and train our models considering 80% as

training dataset and test the models with remaining 20% test dataset. Test dataset is used to provide an unbiased evaluation of a null model t with the training dataset. Section 4.1 shows the details of the validation results and Figure 3,4 and 5 shows the overall cross validation process and learning curve of SVC.

Chapter 4: Result & Discussion

In this segment, we clarify the prediction of ML Algorithms result, the assessment of code review comments.

4.1. Classification Results:

To categorized our preprocessing datasets, we utilized 6 diverse classification algorithms. Table 3 shows the precision of the various classifiers that we utilized. It is colossal to see a portion of the broadly utilized algorithms neglect to accomplish palatable execution for this case. Most outstandingly, the Logistic Regression, KNN, Bernoulli NB and Multinomial Naive Bayes classifiers accomplish a normal precision of about 61%. In any case, different variations of Support Vector Machine and Stochastic Gradient Descent (SGD) gave an exactness of over 63% now and again. The Stochastic Gradient Descent (SGD) classifier gives the best precision, it's over 63.89%.

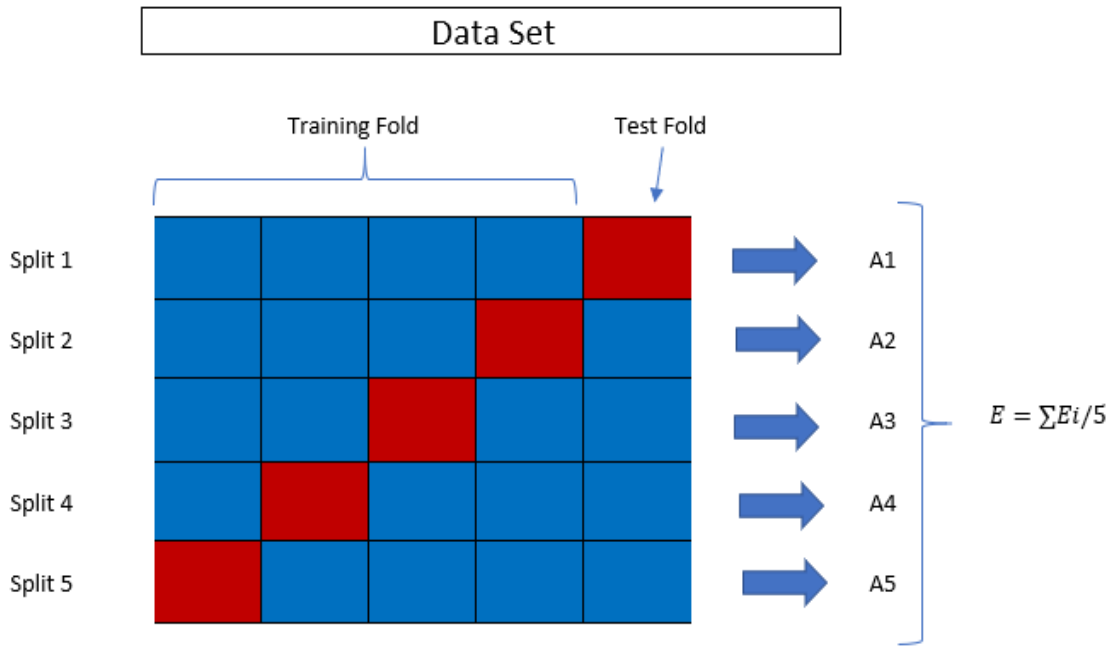


Fig. 3. Cross Validation Process

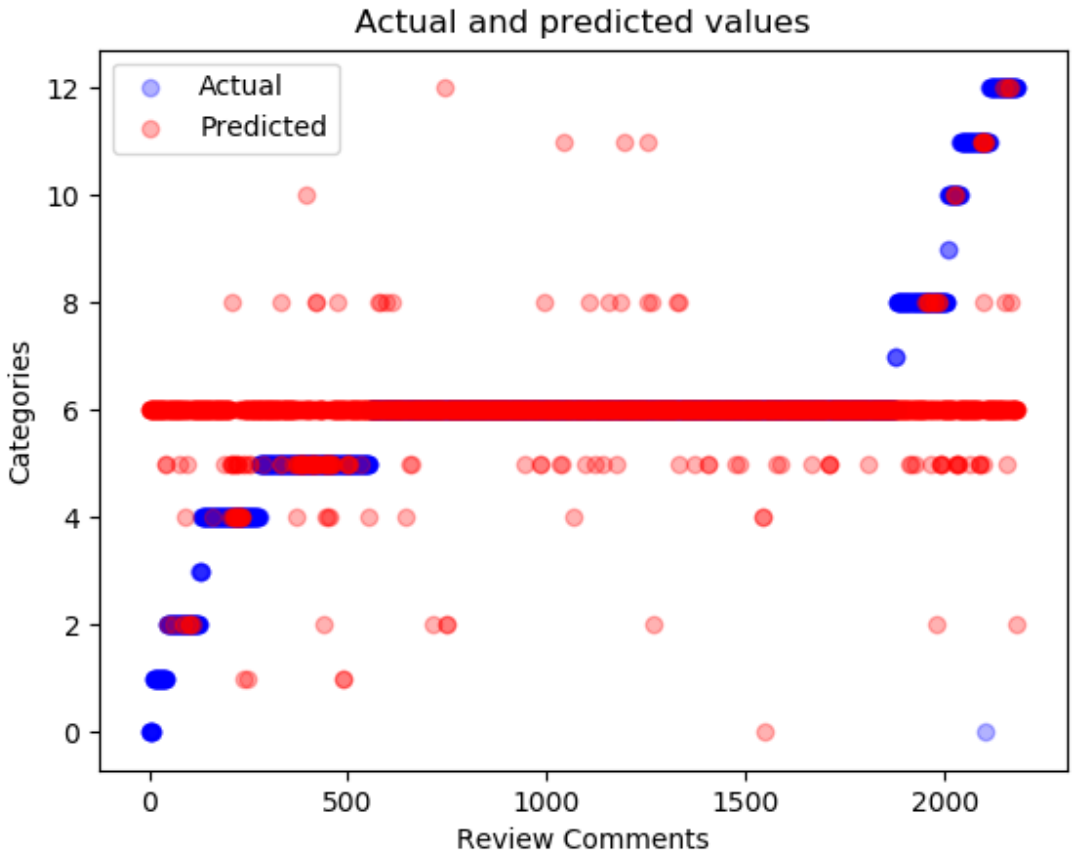


Fig 4: The graph between actual and predicted value.

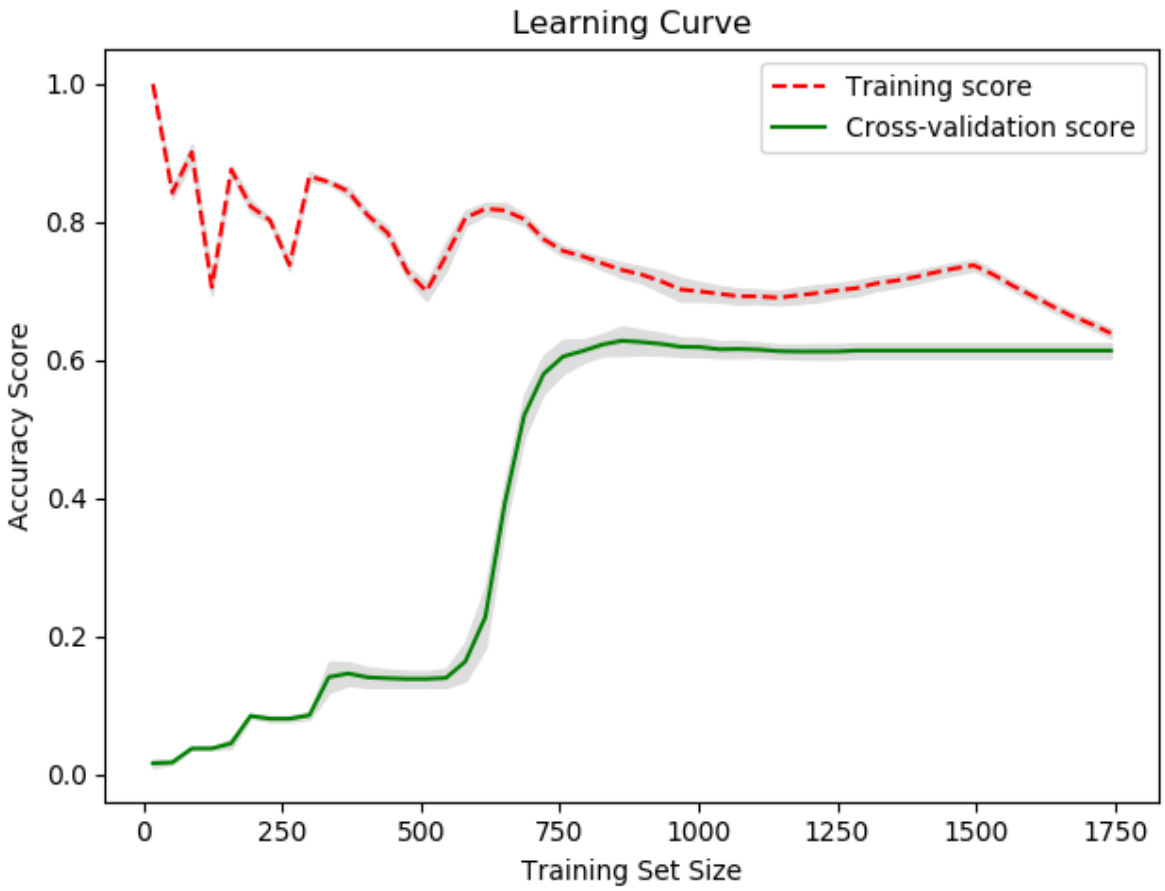


Fig. 5: Learning Curve of SVC

Algorithm	Accuracy (%)
Logistic Regression	61.60
k-Nearest Neighbors (KNN)	61.60
Bernoulli Naive Bayes	61.51
Multinomial Naive Bayes (MNB)	61.37
Support Vector Machine	63.38
Stochastic Gradient Descent (SGD)	63.89

Table 4. Accuracy of different classifiers Algorithms

4.2 Precision, Recall and F-measure for Various Categories of Code Reviews Datasets:

Though the Stochastic Gradient Descent (SGD) gives the most noteworthy exactness, that is the reason we prepared our model with this. By gathering the reference esteem for every classification, we figure the precision, recall, and F1-score of the Stochastic Gradient Descent (SGD) classifier. Table 5 shows the confusion matrix of SGD.

Category	Precision	Recall	F1-score
Documentation	0.53	0.24	0.35
Visual Representation	0.43	0.17	0.24
Organization	0.50	0.28	0.36
Solution Approach	0.27	0.11	0.16
Resource	0.00	0.00	0.00
Validation	0.38	0.14	0.20
Logical	0.56	0.19	0.29
Synchronization	0.75	0.20	0.32
Supporting Library	0.00	0.00	0.00
Defect	0.75	0.17	0.28
API Calls	0.00	0.00	0.00
False Positive	0.00	0.00	0.00
Others	0.67	0.93	0.78

Table 5: Confusion Matrix of Review Comments Datasets

We can state that every ‘Others’ review comment is correctly identified with 93% recall. By identifying the others comment correctly, this result can help the reviewers commenting others comments.

Chapter 5: Conclusions and Recommendations

Code review is a significant static check procedure for improving software quality just as advances information sharing inside a software project. Because of a lot of information examined in our examination, we couldn't recognize specific events of code audit that could assist us with making different investigations. Notwithstanding whether this was possible, given that we used data from the past to have absolute studies, originators would perhaps not review unequivocal cases.

This paper speaks to the automated classification of code reviews dependent on review comments utilizing 6 ML algorithms. Essentially that helps our development lifecycle progressively simpler. We quantitatively explored the effect of code looking into rehearses and supplemented our discoveries with a subjective examination.

Reference:

1. Madera, Michał, and Rafał Tomoń. "A case study on machine learning model for code review expert system in software engineering." 2017 Federated Conference on Computer Science and Information Systems (FedCSIS). IEEE, 2017.

2. Kalyan, Akshay, et al. "A collaborative code review platform for GitHub." *2016 21st International Conference on Engineering of Complex Computer Systems (ICECCS)*. IEEE, 2016.

3. P. C. Rigby and C. Bird, “Convergent contemporary software peer review practices,” in Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. ACM, 2013, pp. 202–212.

4. Sculley, David, et al. "Hidden technical debt in machine learning systems." Advances in neural information processing systems. 2015.

5. Breck, Eric, et al. "What’s your ML Test Score? A rubric for ML production systems." (2016).

6. McMahan, H. Brendan, et al. "Ad click prediction: a view from the trenches." Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013.

7. Chandra, Tushar, et al. "Sibyl: a system for large scale machine learning." Keynote I PowerPoint presentation, Jul 28 (2010).

8. Louis et al.(2018)Louis, Dash, Barr, and Sutton. Annie Louis, Santanu Kumar Dash,Earl T Barr, and Charles Sutton. Deep learning to detect redundant method comments. arXiv preprint arXiv:1806.04616, 2018.

9. White et al.(2016)White, Tufano, Vendome, and Poshyvanyk. Martin White, Michele Tufano, Christopher Vendome, and Denys Poshyvanyk. Deep learning code frag-ments for code clone detection. In Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, pages 87–98. ACM, 2016.

10. Rahman et al.(2019)Rahman, Hossain, and Islam. Shadikur Rahman, Syeda Sumbul Hossain, and Saiful Islam. Context-based news headlines analysis using machine learning approach. In Proceedings of the 11th International Conference Computational Collective Intelligence. Springer, 2019.

11. Bosu, Amiangshu, Michaela Greiler, and Christian Bird. "Characteristics of useful code reviews: An empirical study at microsoft." *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 2015.

12. Louis et al.(2018)Louis, Dash, Barr, and Sutton. Annie Louis, Santanu Kumar Dash, Earl T Barr, and Charles Sutton. Deep learning to detect redundant method comments. arXiv preprint arXiv:1806.04616, 2018.

13. Movshovitz-Attias and Cohen(2013). Dana Movshovitz-Attias and William W Cohen. Natural language models for predicting programming comments. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), volume 2, pages 35–40, 2013.

14. White et al.(2016)White, Tufano, Vendome, and Poshyvanyk. Martin White, Michele Tufano, Christopher Vendome, and Denys Poshyvanyk. Deep learning code fragments for code clone detection. In Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, pages 87–98. ACM, 2016.

15. S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan. An empirical study of the impact of modern code review practices on software quality. Empirical Software Eng., page To appear, 2015.

16. F. E. Harrell Jr. Regression modeling strategies. Springer, 2001.
17. Rennie et al.(2003)Rennie, Shih, Teevan, and Karger. Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. Tackling the poor assumptions of naive bayes text classifiers. In Proceedings of the 20th international conference on machine learning (ICML-03), pages 616{623, 2003.

18. Ho et al.(1994)Ho, Hull, and Srihari. Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. Decision combination in multiple classifier systems. IEEE Transactions on Pattern Analysis & Machine Intelligence, (1):66{75, 1994.

19. Bottou(2010). Leon Bottou. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010, pages 177{186. Springer, 2010.

20. Gunn et al.(1998). Steve R Gunn et al. Support vector machines for classification and regression. ISIS technical report, 14(1):5{16, 1998.

21. Scholkopf et al.(2000)Scholkopf, Smola, Williamson, and Bartlett. Bernhard Scholkopf, Alex J Smola, Robert C Williamson, and Peter L Bartlett. New support vector algorithms. Neural computation, 12(5):1207{1245, 2000.

22. "DO-278/ED-109 Software Standard for Non-Airborne Systems," Radio Technical Commission for Aeronautics (RTCA) and the European Organization for Civil Aviation Equipment (EUROCAE), 2002.

23. Thongtanunam P, Tantithamthavorn C, Kula RG, Yoshida N, Iida H, Matsumoto Ki (2015b) Who should review my code? afile location-based code-reviewer recommendation approach for modern code review. In: SANER 2015. IEEE.pp 141–150.

24. M. Ciolkowski, O. Laitenberger, and S. Bi. "Software reviews, the state of the practice." Software, IEEE, 20(6):46–51, Nov.-Dec. 2003.

25. L. Harjumaa, I. Tervonen, and a Huttunen, "Peer Reviews in Real Life - Motivators and Demotivators," Fifth International Conference on Quality Software (QSIC 05), 2005, pp. 29-36.

26. M. Stein, J. Riedl, S. J. Harner, and V. Mashayekhi, "A case study of distributed, asynchronous software inspection." in ICSE '97: Proceedings of the 19th international conference on Software engineering, (New York, NY, USA), pp. 107–117, ACM, 1997.

27. B. Meyer. "Design and code reviews in the age of the internet." Commun. ACM, 51(9):66–71, 2008.

28. Sodhi, Balwinder, and Shipra Sharma. "Using stackoverflow content to assist in code review." *arXiv preprint arXiv:1803.05689* (2018).

29. I. Sommerville, Software engineering. Pearson, 2010.

30. S. McConnell, Code complete: a practical handbook of software construction. Microsoft Press. (Redmond, WA), 1993.

31. Ashley, Kevin D., and Vern R. Walker. "Toward constructing evidence-based legal arguments using legal decision documents and machine learning." *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law*. ACM, 2013.

32. Fan, J., Kalyanpur, A., Gondek, D. C., and Ferrucci, D. A. Automatic knowledge extraction from documents.IBM J. Res. & Dev.Vol. 56No. 3/4 Paper 5May/July (2012).

33. McCallum et al.(1998)McCallum, Nigam, et al.. Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In AAAI-98 workshop on learning for text categorization, volume 752, pages 41{48. Citeseer, 1998.

34. Lal, Harsh, and Gaurav Pahwa. "Code review analysis of software system using machine learning techniques." *2017 11th International Conference on Intelligent Systems and Control (ISCO)*. IEEE, 2017.

35. http://www.cs.cornell.edu/courses/cs578/2003fa/performancemeasures.pdf, Performance Measures for Machine Learning

36. Liljeson, Mattias, and Alexander Mohlin. "Software defect prediction using machine learning on test and source code metrics." (2014).

37. Mukadam, Murtuza, Christian Bird, and Peter C. Rigby. "Gerrit software code review data from android." *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013.

38. Sonnenburg, SÃ¦ren, et al. "The need for open source software in machine learning." *Journal of Machine Learning Research* 8.Oct (2007): 2443-2466.

39. Benson, S. J., et al. "TAO user manual (revision 1.7)(Technical Report ANL/MCS-TM-242). Mathematics and Computer Science Division, Argonne National Laboratory." (2004).

40. Suffian, Muhammad Dhiauddin Mohamed, and Suhaimi Ibrahim. "A prediction model for system testing defects using regression analysis." *arXiv preprint arXiv:1401.5830* (2014).

41. Merugu, Chaithanya Kadari1 Anusha. "EXHUMING SOFTWARE DEFECT USING VARIOUS STATIC DEFECT MODELS IN DATA MINING."

42. Challagulla, Venkata Udaya B., et al. "Empirical assessment of machine learning based software defect prediction techniques." *International Journal on Artificial Intelligence Tools* 17.02 (2008): 389-400.

43. Challagulla, Venkata Udaya B., et al. "Empirical assessment of machine learning based software defect prediction techniques." *International Journal on Artificial Intelligence Tools* 17.02 (2008): 389-400.

44. Hossain, Syeda Sumbul. "Challenges and Mitigation Strategies in Reusing Requirements in Large-Scale Distributed Agile Software Development: A Survey Result." *Intelligent Computing-Proceedings of the Computing Conference*. Springer, Cham, 2019.

45. Bacchelli, Alberto, and Christian Bird. "Expectations, outcomes, and challenges of modern code review." *Proceedings of the 2013 international conference on software engineering*. IEEE Press, 2013.

46. Holzmann, Gerard J. "SCRUB: a tool for code reviews." *Innovations in Systems and Software Engineering* 6.4 (2010): 311-318.

47. Ahmed, Toufique, et al. "SentiCR: a customized sentiment analysis tool for code review interactions." *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 2017.

48. Moskowitz, Milton, William Potter, and Wayne King. "Automatic computer code review tool." U.S. Patent Application No. 10/769,535.

49. Sadowski, Caitlin, et al. "Modern code review: a case study at google." *Proceedings of the International Conference on Software Engineering: Software Engineering in Practice*. Association for Computing Machinery, 2018.

