

AN EXPERIMENTAL STUDY FOR SOFTWARE QUALITY PREDICTION USING MACHINE LEARNING METHODS

¹J.Sravan Kumar Varma, ²S.Reddy Mubaraq

¹M.Tech Student, Dept. of CSE, Golden Valley Integrated Campus, Madanapalli, Andhra Pradesh, India

²Asst.Professor, Dept. of CSE, Golden Valley Integrated Campus, Madanapalli, Andhra Pradesh, India

ABSTRACT

Software quality estimation is an activity needed at various stages of software development. It may be used for planning the project's quality assurance practices and for benchmarking. In earlier previous studies, two methods (Multiple Criteria Linear Programming and Multiple Criteria Quadratic Programming) for estimating the quality of software had been used. Also, C5.0, SVM and Neutral network were experimented with for quality estimation. These studies have relatively low accuracies. In this study, we aimed to improve estimation accuracy by using relevant features of a large dataset. We used a feature selection method and correlation matrix for reaching higher accuracies. In addition, we have experimented with recent methods shown to be successful for other prediction tasks. Machine learning algorithms such as Xgboost, Random Forest and Decision Tree are applied to the data to predict the software quality and reveal the relation between the quality and development attributes. The experimental results show that the quality level of software can be well estimated by machine learning algorithms.

Keywords: — Estimation, Machine Learning, Software Quality..

1. Introduction

Software applications may contain defects, originating from requirements analysis, specification and other activities conducted in the software development. Therefore, software quality estimation is an activity needed at various stages [1]. It may be used for planning the project based quality assurance practices and for benchmarking. In addition, the number of defects per unit is considered one of the most important factors that indicate the quality of the software [2].

There are two directly comparable studies on software quality prediction using defect quantities in ISBGS dataset. In the first study, the two methods (MCLP and MCQP) were experimented with the dataset and the results were compared [3]. The quality level was classified according to: number of minor defect + 2*number of major defect + 4*number of extreme defect. The quality of level was to be either high or low. They used k-fold cross-validation technique to measure MCLP and MCQP's performance on the ISBSG database. Release 10 Dataset (released in January 2007) which contained 4,017 records and 106 attributes was used. After preprocessing, 374 records and 11 attributes remained in the dataset.

In another study, the same data set was used again [4]. The software belonged to high quality class if it fulfills the following requirements: the extreme defects exist or the number of major defects is more than 1 or the number of minor defects is more than 10. The rest are assumed to belong to low quality class. After preprocessing, 746 projects and 53 attributes remained in the dataset. They used C5.0, SVM and Neutral network for classification.

As an example to a more application oriented study Rashid et al. [5] used case based reasoning (CBR) for software quality estimation. CBR is a machine learning model which performs the learning process using the results of the previous experiments. Line of code, number of function, difficulty level, and development type and programmers experience are entered and these attributes are used for estimation. The deviation is calculated by using Euclidian distance (ED) or The Manhattan distance (MD). If the error in estimation is less than 10% then the record is saved to the database.

Number of inputs that can be obtained from the user is limited. Also, it is necessary to have close values in the database in order to estimating precise values.

In these studies, quality estimation was done by binary classification. We tried to improve these prediction models, taking into account the size in terms of function points and using 4-level classification. We have experimented with recent classification methods shown to be successful for other prediction tasks

2. Existing System

Software quality estimation is an activity needed at various stages of software development. It may be used for planning the project's quality assurance practices and for benchmarking. In earlier previous studies, two methods (Multiple Criteria Linear Programming and Multiple Criteria Quadratic Programming) for estimating the quality of software had been used. Also, C5.0, SVM and Neural network were experimented with for quality estimation. These studies have relatively low accuracies.

Demerits of Existing System

- To improve estimation accuracy by using relevant features of a large dataset.

3. Proposed System

In this paper We used a feature selection method and correlation matrix for reaching higher accuracies. In addition, we have experimented with recent methods shown to be successful for other prediction tasks. Machine learning algorithms such as Xgboost, Random Forest and Decision Tree are applied to the data to predict the software quality and reveal the relation between the quality and development attributes. The experimental results show that the quality level of software can be well estimated by machine learning algorithms.

4. Literature Survey

[1] Vijay, T. John, D. M. G. Chand, and D. H. Done. "Software quality metrics in quality assurance to study the impact of external factors related to time." *International Journal of Advanced Research in Computer Science and Software Engineering*, 2017.

The purpose of this research is to build a software quality assessment model to evaluate the quality of mobile-based elderly fall detection software. The assessment is based on the quality factors found in the software quality model. The quality factor is adjusted to the characteristics of the software. The model is needed because the software has its own characteristics. This research consists of several stages. The first thing to do is analysing the software domain to determine its characteristics. The second is defining the software assessment needs by mapping software characteristics with the quality standards used (ISO / IEC 25010: 2011) to obtain the appropriate quality factors. The software quality metrics is determined after the quality factors is obtained. The metric to be used is Goal Question Metrics (GQM). The third is software quality weighting process, including its criterias and sub-criterias. Determination of the equation for software quality assesment is the final stage of the research. Based on the reseach process, it can be concluded that the model developed successfully can be used to assess the software.

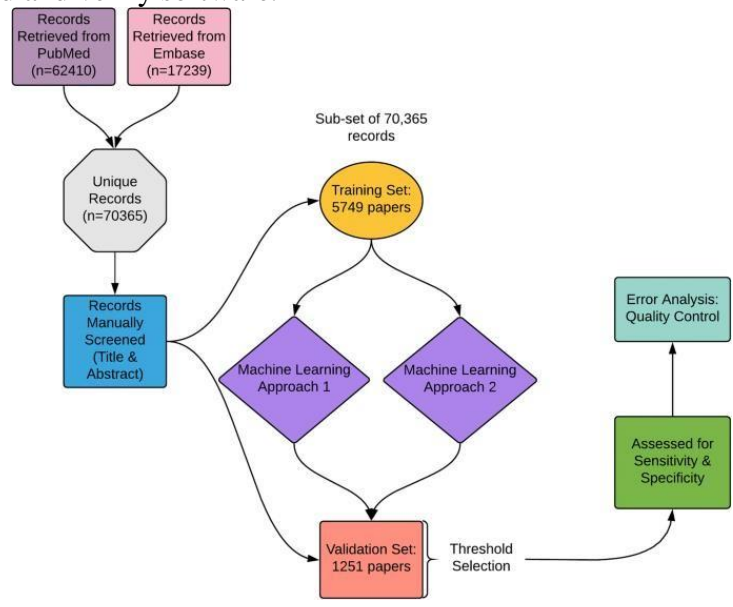
[2] D. Bowes, T. Hall, and J. Petrić, "Software defect prediction: do different classifiers find the same defects?." *Software Quality Journal*, 26(2), 2018, pp. 525-552.

During the last 10 years, hundreds of different defect prediction models have been published. The performance of the classifiers used in these models is reported to be similar with models rarely performing above the predictive performance ceiling of about 80% recall. We investigate the individual defects that four classifiers predict and analyse the level of prediction uncertainty produced by these classifiers. We perform a sensitivity analysis to compare the performance of Random Forest, Naïve Bayes, RPart and SVM classifiers when predicting defects in NASA, open source and commercial datasets. The defect predictions that each classifier makes is captured in a confusion matrix and the prediction uncertainty of each classifier is compared. Despite similar predictive performance values for these four classifiers, each detects different sets of defects. Some classifiers are more consistent in predicting defects than others. Our results confirm that a unique subset of

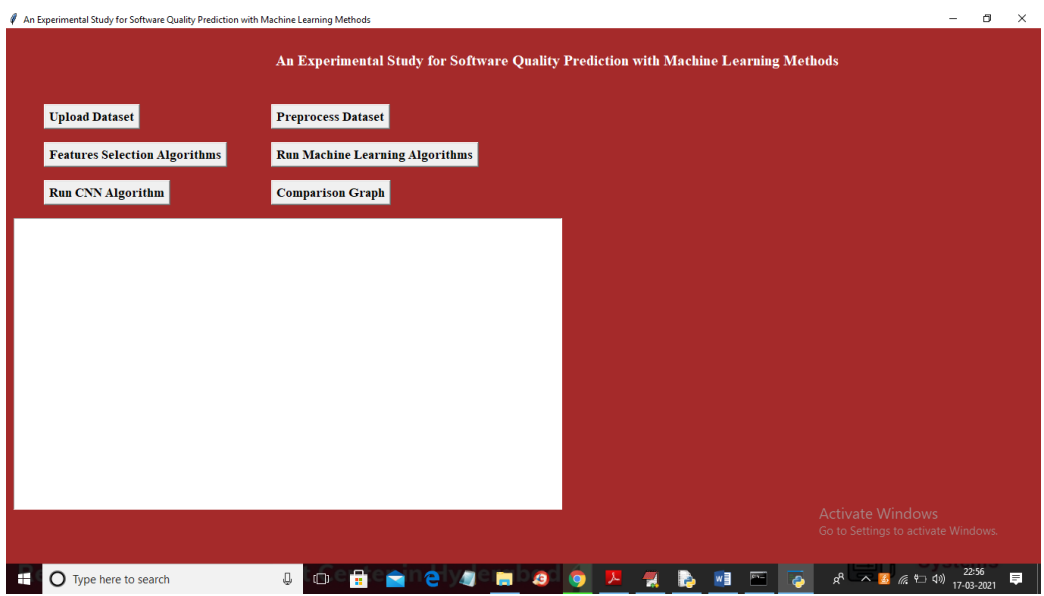
defects can be detected by specific classifiers. However, while some classifiers are consistent in the predictions they make, other classifiers vary in their predictions. Given our results, we conclude that classifier ensembles with decision-making strategies not based on majority voting are likely to perform best in defect prediction.

5. System Architecture

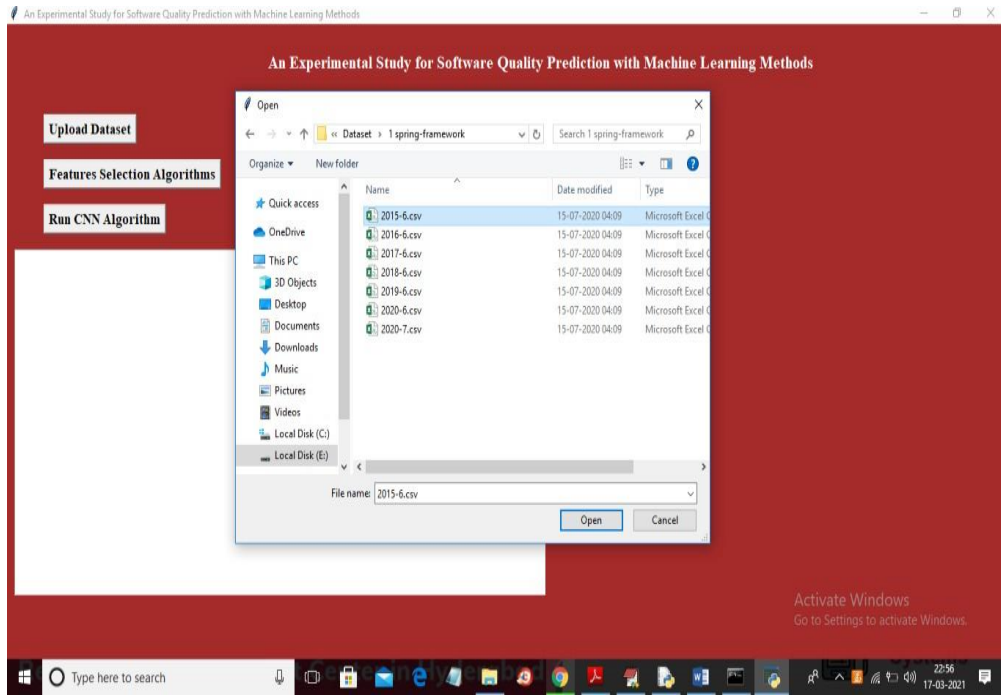
Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirements have been specified and analyzed, system design is the first of the three technical activities - design, code and test that is required to build and verify software.



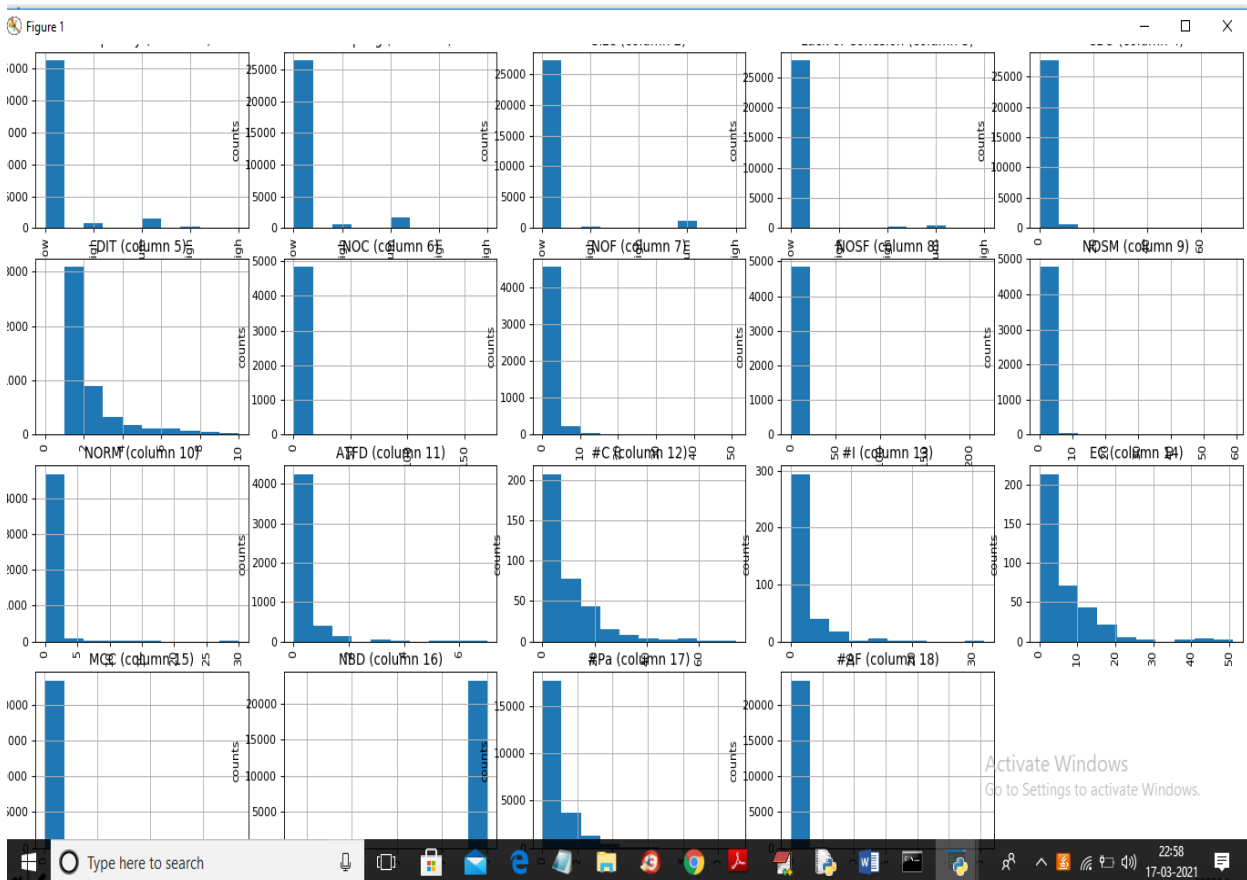
6. Experimental Outcomes



In above screen click on ‘Upload Dataset’ button to and upload dataset

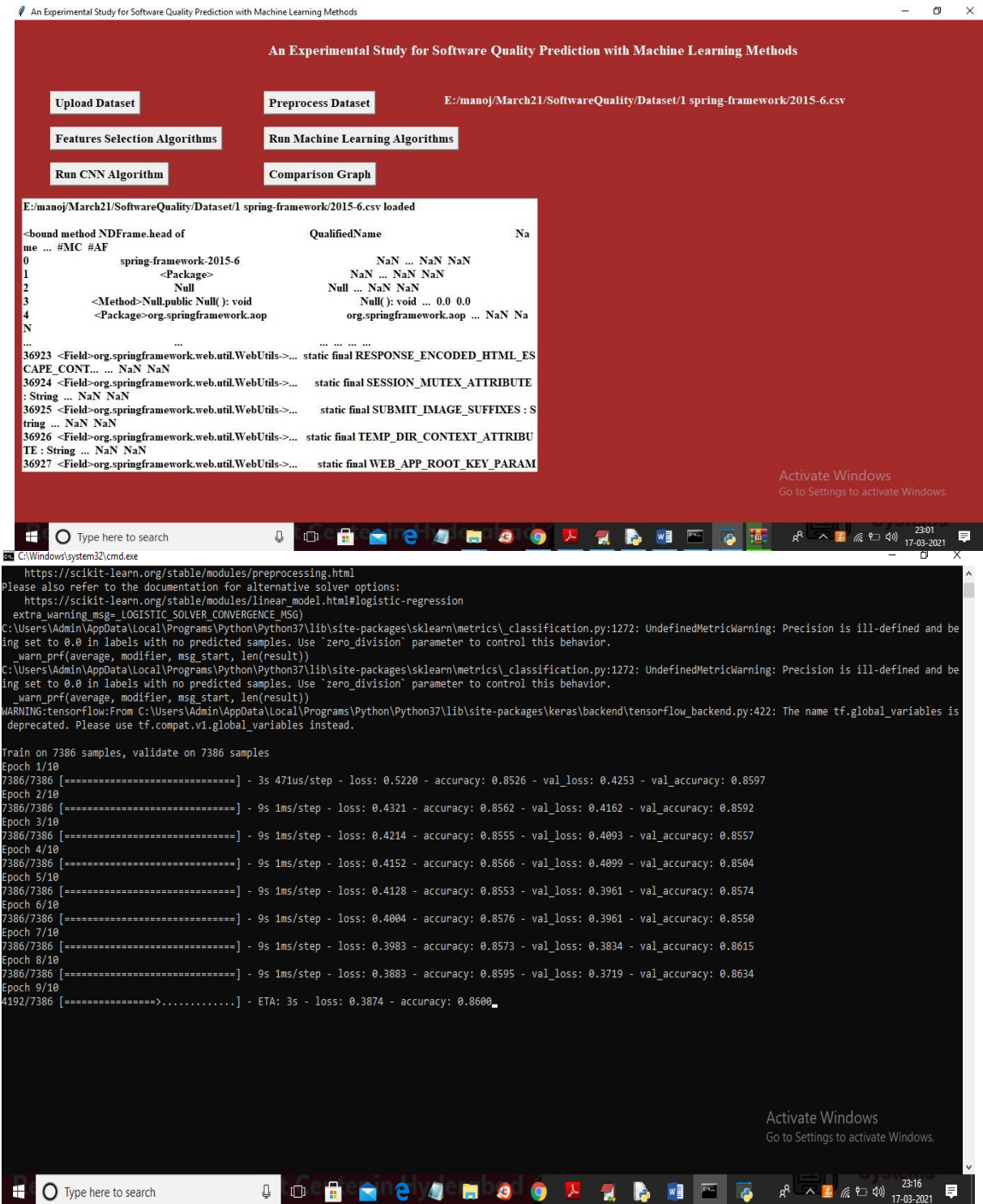


In above screen selecting and uploading ‘2015-6.csv’ dataset file and then click on ‘Open’ button to load dataset and to get below screen



In above graph we can see each graph represents one column from dataset and from that columns its counting each distinct value from and plot in that graph for example in second graph NOC columns 3 different values and its plotting 3 different bars with count and no close above graph to get below screen

In above screen displaying values from dataset and we can see dataset contains NAN (missing values) and string non numeric values and we need to replace all missing and non-numeric values with their count so click on 'Preprocess Dataset' button



An Experimental Study for Software Quality Prediction with Machine Learning Methods

Upload Dataset Preprocess Dataset E:/manoj/March21/SoftwareQuality/Dataset/1 spring-framework/2015-6.csv

Features Selection Algorithms Run Machine Learning Algorithms

Run CNN Algorithm Comparison Graph

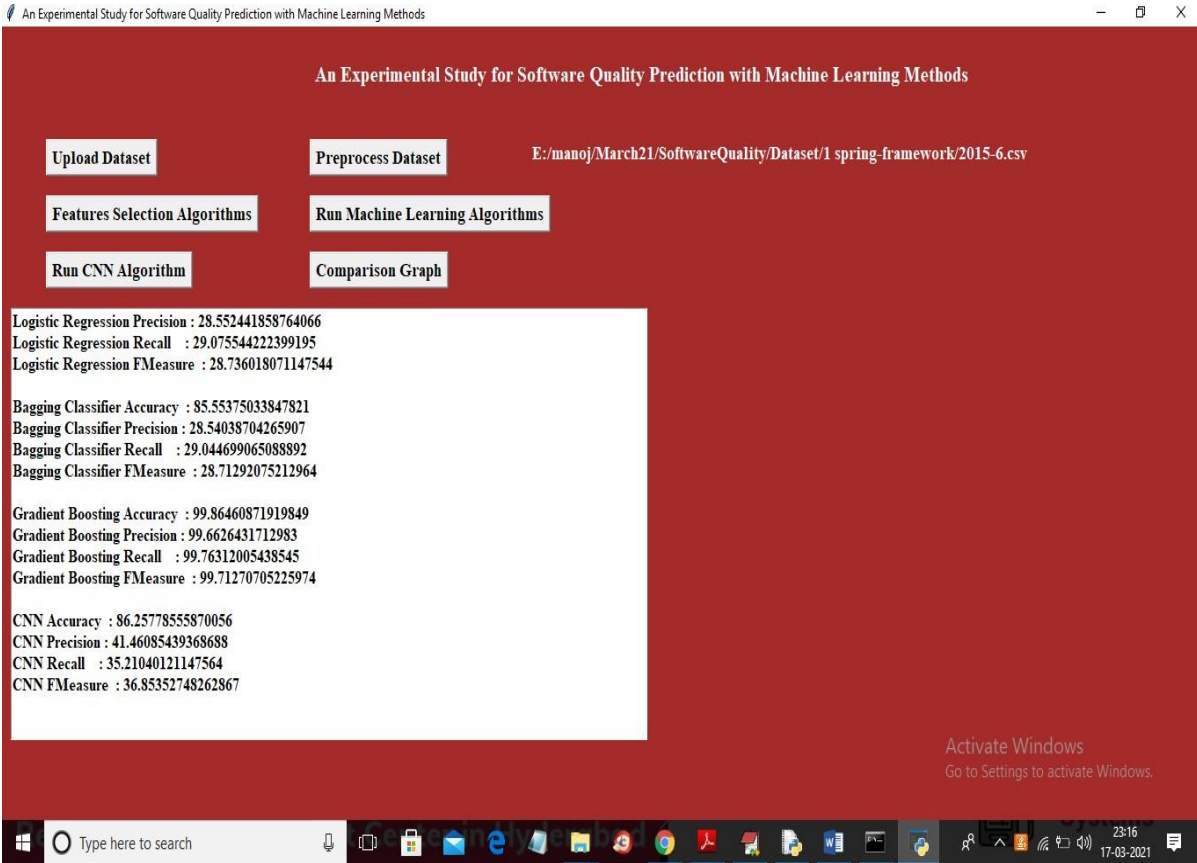
```
E:/manoj/March21/SoftwareQuality/Dataset/1 spring-framework/2015-6.csv loaded
```

me ... #MC #AF	QualifiedName	Na
0	spring-framework-2015-6	NaN ... NaN NaN
1	<Package>	NaN ... NaN NaN
2	Null	Null ... NaN NaN
3	<Method>Null.public Null(): void	Null(): void ... 0.0 0.0
4	<Package>org.springframework.aop	org.springframework.aop ... NaN Na
...
36923	<Field>org.springframework.web.util.WebUtils->... static final RESPONSE_ENCODED_HTML_ESCAPE_CONT... .. NaN NaN	
36924	<Field>org.springframework.web.util.WebUtils->... static final SESSION_MUTEX_ATTRIBUTE : String ... NaN NaN	
36925	<Field>org.springframework.web.util.WebUtils->... static final SUBMIT_IMAGE_SUFFIXES : S	
...
36926	<Field>org.springframework.web.util.WebUtils->... static final TEMP_DIR_CONTEXT_ATTRIBU	
...
36927	<Field>org.springframework.web.util.WebUtils->... static final WEB_APP_ROOT_KEY_PARAM	

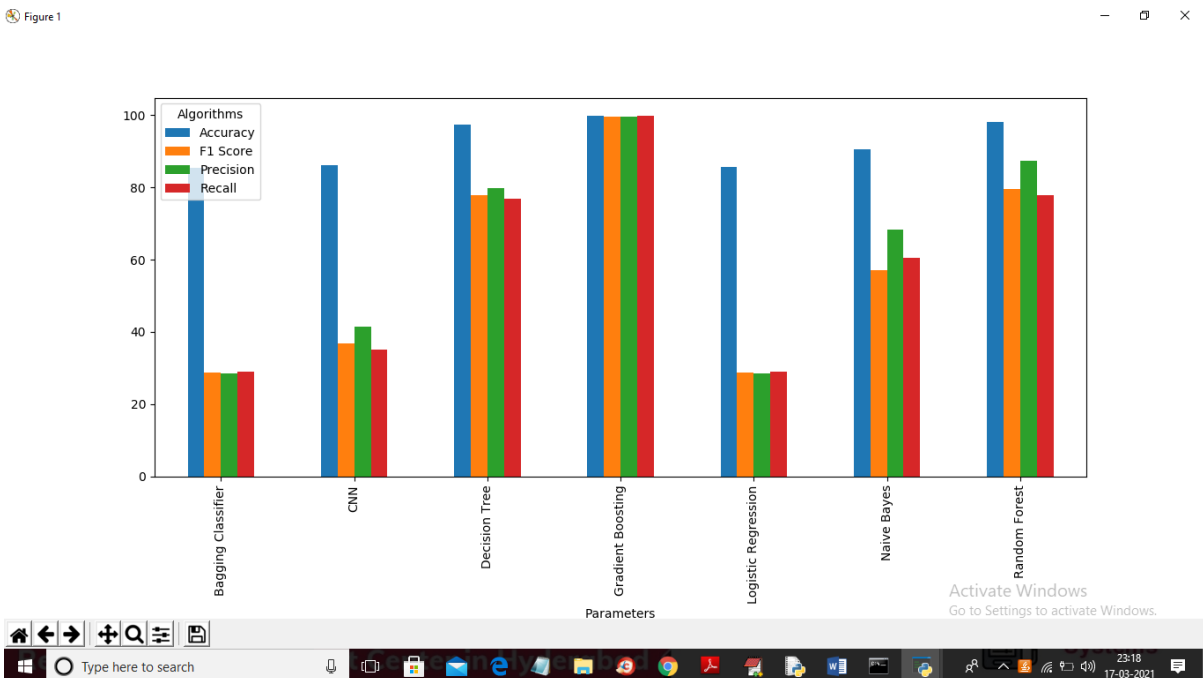
```
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg= LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
  warn_prf(average, modifier, msg_start, len(result))
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
  warn_prf(average, modifier, msg_start, len(result))
WARNING:tensorflow:From C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

Train on 7386 samples, validate on 7386 samples
Epoch 1/10
7386/7386 [=====] - 3s 471us/step - loss: 0.5220 - accuracy: 0.8526 - val_loss: 0.4253 - val_accuracy: 0.8597
Epoch 2/10
7386/7386 [=====] - 9s 1ms/step - loss: 0.4321 - accuracy: 0.8562 - val_loss: 0.4162 - val_accuracy: 0.8592
Epoch 3/10
7386/7386 [=====] - 9s 1ms/step - loss: 0.4214 - accuracy: 0.8555 - val_loss: 0.4093 - val_accuracy: 0.8557
Epoch 4/10
7386/7386 [=====] - 9s 1ms/step - loss: 0.4152 - accuracy: 0.8566 - val_loss: 0.4099 - val_accuracy: 0.8504
Epoch 5/10
7386/7386 [=====] - 9s 1ms/step - loss: 0.4128 - accuracy: 0.8553 - val_loss: 0.3961 - val_accuracy: 0.8574
Epoch 6/10
7386/7386 [=====] - 9s 1ms/step - loss: 0.4004 - accuracy: 0.8576 - val_loss: 0.3961 - val_accuracy: 0.8550
Epoch 7/10
7386/7386 [=====] - 9s 1ms/step - loss: 0.3983 - accuracy: 0.8573 - val_loss: 0.3834 - val_accuracy: 0.8615
Epoch 8/10
7386/7386 [=====] - 9s 1ms/step - loss: 0.3883 - accuracy: 0.8595 - val_loss: 0.3719 - val_accuracy: 0.8634
Epoch 9/10
4192/7386 [=====]..... - ETA: 3s - loss: 0.3874 - accuracy: 0.8600_
```

In above screen to train CNN we took 10 iterations or epoch and at each epoch accuracy get better and loss get reduce and after 10 iterations will get below screen



In above screen we got output values for CNN also and now click on ‘Comparison Graph’ button to get below screen



In above graph we are plotting accuracy, precision, recall and accuracy for each algorithm

7. CONCLUSION AND FUTURE ENHANCEMENT

In this paper we have experimented classification algorithms using Scikit-learn library on two dataset. We have experimented with recent algorithms that support multi-class classification. The accuracies achieved by using these algorithms are 92.28% on EBSPM Dataset and 92.22% on ISBSG Dataset. In comparison to previous directly comparable studies, acceptable level multiclass quality prediction could be achieved.

8. References

- [1] Vijay, T. John, D. M. G. Chand, and D. H. Done. "Software quality metrics in quality assurance to study the impact of external factors related to time." *International Journal of Advanced Research in Computer Science and Software Engineering*, 2017.
- [2] D. Bowes, T. Hall, and J. Petrić, "Software defect prediction: do different classifiers find the same defects?." *Software Quality Journal*, 26(2), 2018, pp. 525-552.
- [3] X. Wang, Y. Zhang, L. Zhang and Y. Shi, "A Knowledge Discovery Case Study of Software Quality Prediction: ISBSG Database," 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Toronto, ON, 2010, pp.219-222.
- [4] X. Wang, Y. Zhang, L. Zhang and Y. Shi, "A Knowledge Discovery Case Study of Software Quality Prediction Based on Classification Models: ISBSG Database," *The 11th International Symposium on Knowledge Systems Sciences (KSS 2010)*, 2010
- [5] E. Rashid, S. Patnaik, and V. Bhattacharjee, "Software quality estimation using machine learning: Case-Based reasoning technique, " *International Journal of Computer Applications*, 2012
- [6] www.isbsg.org
- [7] <https://goverdson.nl/>
- [8] H. Huijgens, "Evidence-based software portfolio management: a tool description and evaluation", 20th International Conference on Evaluation and Assessment in Software Engineering (EASE '16), 201