An Experimental Study for Software Quality Prediction with Machine Learning Methods

Ch. Nuthana[1], Ch. Sree Harika[1], Ch. Chandana[1], K. Kumara Swamy[2]

[1]UG Student, [2]Assistant Professor, [1,2]Department of Information Technology

[1,2]Malla Reddy Engineering College for Women (UGC-Autonomous), Maisammaguda, Secunderabad, Telangana, 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: software quality prediction, machine learning, supervised learning model.

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. 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. 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. 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. The software belonged to high quality class if it fulfils 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 functions, 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. System Analysis

2.1 EXISTING SYSTEM

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. 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.

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. 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.

2.2 PROPOSED SYSTEM

In this project, various machine learning algorithms such as Random Forest, Decision Tree, Gradient Boosting, Bagging Classifier, Logistic Regression, BernoulliNB and CNN are used to predict software quality. To implement this project author has used two datasets but those datasets are not available on internet so whatever dataset student sent I am using that dataset only. This dataset saved inside Dataset folder.

3. MODULES

- Upload Dataset
- Data Preprocessing
- Feature Selection
- Machine Learning Algorithms
- Deep Learning Algorithms

UPLOAD DATASET

First person needs to upload the dataset. The dataset should contain the data that is related to the software and with many parameters based on which the software is built. We need to select the dataset from our system and the selected dataset is used for the software quality analysis.

PREPROCESS DATASET

There we are missing values in datasets. As it is important for training accuracy, some of the rows with missing and undefined values were deleted. The summary of the final datasets obtained are given.

Secondly, both of datasets had attributed with many null values. So, the attributes which have the least number of empty rows were selected as predictor attributes. After this process, ISBSG Dataset was reduced to 11 attributes and 1 target class. EBSPM Dataset was reduced to 10 attributes and 1 target class. Furthermore, we divided the level of software quality variable into 4 classes according to the defect density values of the project . Finally, some of columns had string values. Final preprocessing operation on datasets is converting the string values to a set of categories represented as integers for enumeration only. This categorical data was needed for the training chosen classification algorithms during the learning phase.

FEATURES SELECTION ALGORITHM

While building a machine learning model for real-life dataset, we come across a lot of features in the dataset and not all these features are important every time. Adding unnecessary features while training the model leads us to reduce the overall accuracy of the model, increase the complexity of the model and decrease the generalization capability of the model and makes the model biased. Even the saying "Sometimes less is better" goes as well for the machine learning model. Hence, feature selection is one of the important steps while building a machine learning model. Its goal is to find the best possible set of features for building a machine learning model.

MACHINE LEARNING ALGORITHMS

In this project we have used various machine learning algorithms such as Random Forest, Decision Tree, Gradient Boosting, Bagging Classifier, Logistic Regression, BernoulliNB and Naive Bayes to predict software quality. To implement this project author has used two datasets but those datasets are not available on internet so whatever dataset student sent I am using that dataset only. This dataset saved inside Dataset folder.

DEEP LEARNING ALGORITHM

In our project we have used the CNN algorithm, A convolutional neural network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals. CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces. This characteristic that makes convolutional neural network so robust for computer vision.

- CNN can run directly on an underdone image and do not need any preprocessing.
- A convolutional neural network is a feed forward neural network, seldom with up to 20.
- The strength of a convolutional neural network comes from a particular kind of layer called the convolutional layer.

4. ABOUT DATASET

EBSPM is a research repository developed as an evidence-based, practical model to support software companies to actively improve their software delivery portfolio. The goal of the EBSPM is to measure, analyze, and benchmark the performance of interconnected sets of software projects in terms of size, cost, duration, and number of defects, in order to support innovation of a company's software delivery capability.

On the other hand, ISBSG is a not-for-profit organization which was founded in Australia in 1997 by a group of national software metrics associations. They aimed to promote the use of IT industry data to improve software processes and products.

Evidence-Based Software Portfolio Management (EBSPM) database has 492 finalized software projects from 4 different companies in The Netherlands and Belgium. In figure 1 some of the attributes of the EPSPM dataset is shown. This dataset was published relatively recently, on 24 Jul 2017. Although the dataset included Functional Size and Defect Process values, there was no Defect Density value. Therefore, the defect density value is calculated as Defect * 1000 / Functional size. We have added software quality attribute to this dataset based in defect types and density.

5. Literature Survey

5.1 Software quality metrics in quality assurance to study the impact of external factors related to time:

Software quality assurance is a formal process for evaluating and documenting the quality of the work products during each stage of the software development lifecycle. The practice of applying software metrics to operational factors and to maintain factors is a complex task. Successful software quality assurance is highly dependent on software metrics. It needs linkage the software quality model and software metrics through quality factors in order to offer measure method for software quality assurance. The contributions of this paper build an appropriate method of Software quality metrics application in quality life cycle with software quality assurance. Design: The purpose approach defines some software metrics in the factors and discussed several software quality assurance model and some quality factors measure method. Methodology: This paper solves customer value evaluation problem are: Build a framework of combination of software quality criteria. Describes software metrics. Build Software quality metrics application in quality life cycle with software quality assurance. Results: From the appropriate method of Software quality metrics application in quality life cycle with software quality assurance, each activity in the software life cycle, there is one or more QA quality measure metrics focus on ensuring the quality of the process and the resulting product. Future research is need to extend and improve the methodology to extend metrics that have been validated on one project, using our criteria, valid measures of quality on future software project.

5.2 Software defect prediction: do different classifiers find the same defects:

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.

5.3 A Knowledge Discovery Case Study of Software Quality Prediction:

Software becomes more and more important in modern society. However, the quality of software is influenced by many un-trustworthy factors. This paper applies MCLP model on ISBSG database to predict the quality of software and reveal the relation between the quality and development attributes. The experimental result shows that the quality level of software can be well predicted by MCLP Model. Besides, several useful conclusions have been drawn from the experimental result.

5.4 Evidence-based software portfolio management:

In this paper we describe and evaluate a tool for Evidence-Based Software Portfolio Management (EBSPM) that we developed over time in close cooperation with software practitioners from The Netherlands and Belgium. Objectives: The goal of the EBSPM-tool is to measure, analyze, and benchmark the performance of interconnected sets of software projects in terms of size, cost, duration, and number of defects, in order to support innovation of a company's software delivery capability. The tool supports building and maintaining a research repository of finalized software projects from different companies, business domains, and delivery approaches. Method: The tool consists of two parts. First, a Research Repository, at this moment holding data of for now 490 finalized software projects, from four different companies. Second, a Performance Dashboard, built from a so-called Cost Duration Matrix. Results: We evaluated the tool by describing its use in two practical applications in case studies in industry. Conclusions: We show that the EBSPM-tool can be used successfully in an industrial context, especially regarding its benchmarking and visualization purposes.

6. Proposed Methodology

6.1 MACHINE LEARNING

Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data. Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is like the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively.

6.1.1 CATEGORIES OF MACHINE LEARNING

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

- Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

- Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data.

Machine learning, which is a subset of artificial intelligence. Since neural networks imitate the human brain and so deep learning will do. In deep learning, nothing is programmed explicitly. Basically, it is a machine learning class that makes use of numerous nonlinear processing units so as to perform feature extraction as well as transformation. The output from each preceding layer is taken as input by each one of the successive layers.

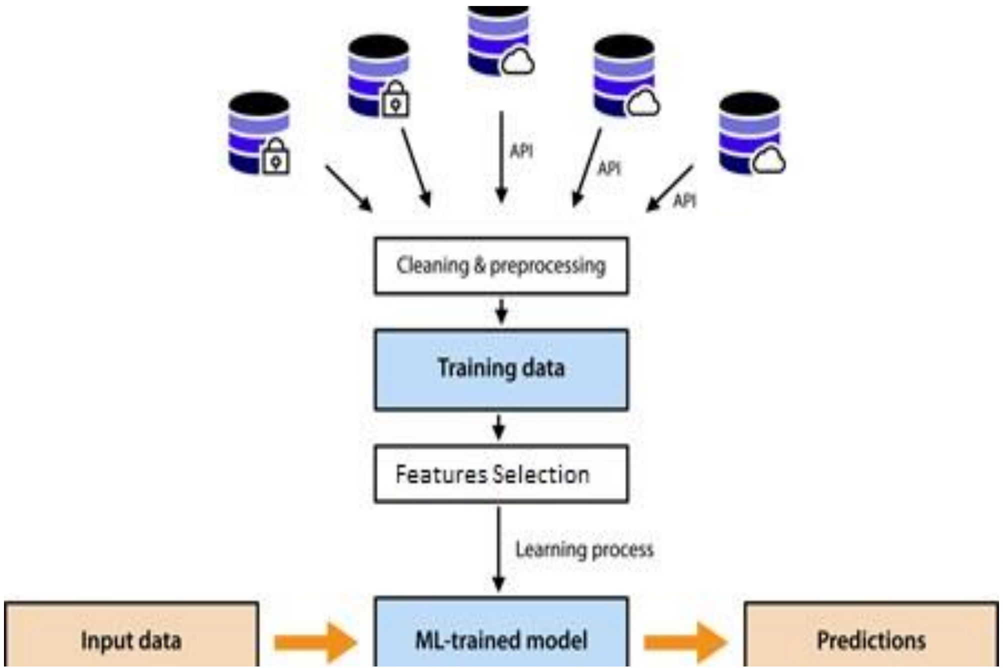


Fig. 2 Proposed System architecture.

6.1.2 Terminologies of Machine Learning

- Model – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- Feature – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- Target (Label) – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- Training – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- Prediction – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

6.2 NAIVE BAYES ALGORITHM

Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.

- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Naive Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Naive: It is called Naive because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem

6.3 DECISION TREE ALGORITHM

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome.

- The decisions or the test are performed based on features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, like a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- A decision tree simply asks a question and based on the answer (Yes/No), it further split the tree into subtrees.

6.4 RANDOM FOREST ALGORITHM

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting. Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

6.5 LOGISTIC REGRESSION ALGORITHM

Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable (or output), y, can take only discrete values for a given set of features(or inputs), X. Contrary to popular belief, logistic regression IS a regression model. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as "1". Just like Linear regression assumes that the data follows a linear function, Logistic regression models the data using the sigmoid function. Logistic regression becomes a classification technique only when a decision threshold is brought into the picture.

The setting of the threshold value is a very important aspect of Logistic regression and is dependent on the classification problem itself.

6.6 BAGGING CLASSIFIER ALGORITHM

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out OF IT. Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples(or data) from the original training dataset – whereN is the size of the original training set. Training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out. Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.

6.7 GRADIENT BOOSTING ALGORITHM

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor's error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.

There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees).

There is an important parameter used in this technique known as Shrinkage.

Shrinkage refers to the fact that the prediction of each tree in the ensemble is shrunk after it is multiplied by the learning rate (eta) which ranges between 0 to 1. There is a trade-off between eta and number of estimators, decreasing learning rate needs to be compensated with increasing estimators in order to reach certain model performance. Since all trees are trained now, predictions can be made.

6.8 DEEP LEARNING

Deep learning models are capable enough to focus on the accurate features themselves by requiring a little guidance from the programmer and are very helpful in solving out the problem of dimensionality. Deep learning algorithms are used, especially when we have a huge no of inputs and outputs. Since deep learning has been evolved by the machine learning, which itself is a subset of artificial intelligence and as the idea behind the artificial intelligence is to mimic the human behavior, so same is “the idea of deep learning to build such algorithm that can mimic the brain”.

- Deep learning is implemented with the help of Neural Networks, and the idea behind the motivation of Neural Network is the biological neurons, which is nothing but a brain cell.
- Deep learning is a collection of statistical techniques of machine learning for learning feature hierarchies that are actually based on artificial neural networks.

So basically, deep learning is implemented by the help of deep networks, which are nothing but neural networks with multiple hidden layers. We provide the raw data of images to the first layer of the input layer. After then, these input layer will determine the patterns of local contrast that means it will differentiate on the basis of colors, luminosity, etc. Then the 1st hidden layer will determine the face feature, i.e., it will fixate on eyes, nose, and lips, etc. And then, it will fixate those face features on the correct face template. So, in the 2nd hidden layer, it will actually determine the correct face here as it can be seen in the above image, after which it will be sent to the output layer.

6.9 CONVOLUTIONAL NEURAL NETWORK(CNN)

A convolutional neural network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals. CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces. This characteristic that makes convolutional neural network so robust for computer vision.

- CNN can run directly on a underdone image and do not need any preprocessing.
- A convolutional neural network is a feed forward neural network, seldom with up to 20.
- The strength of a convolutional neural network comes from a particular kind of layer called the convolutional layer.
- CNN contains many convolutional layers assembled on top of each other, each one competent of recognizing more sophisticated shapes.
- With three or four convolutional layers it is viable to recognize handwritten digits and with 25 layers it is possible to differentiate human faces.
- The agenda for this sphere is to activate machines to view the world as humans do, perceive it in a alike fashion and even use the knowledge for a multitude of duty such as image and video recognition, image inspection and classification, media recreation, recommendation systems, natural language processing, etc.

4. Results and Discussion

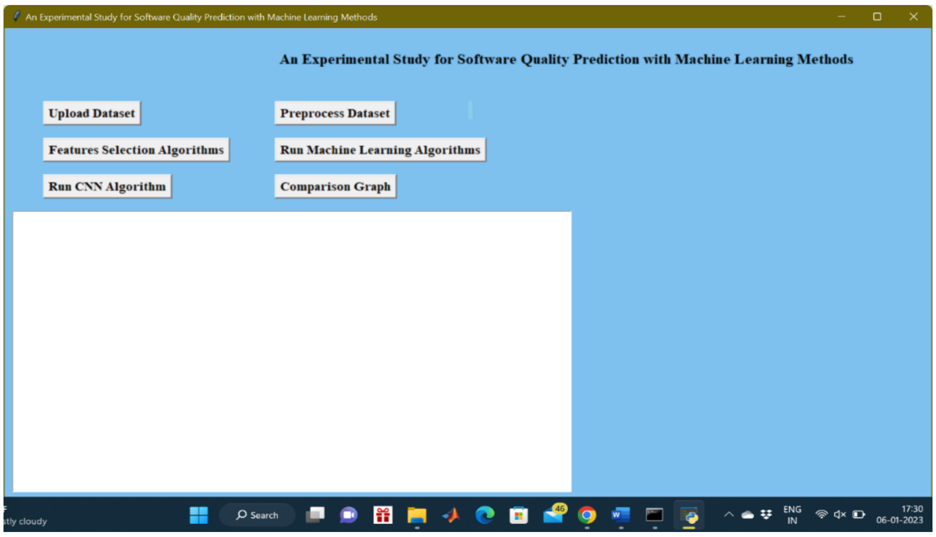
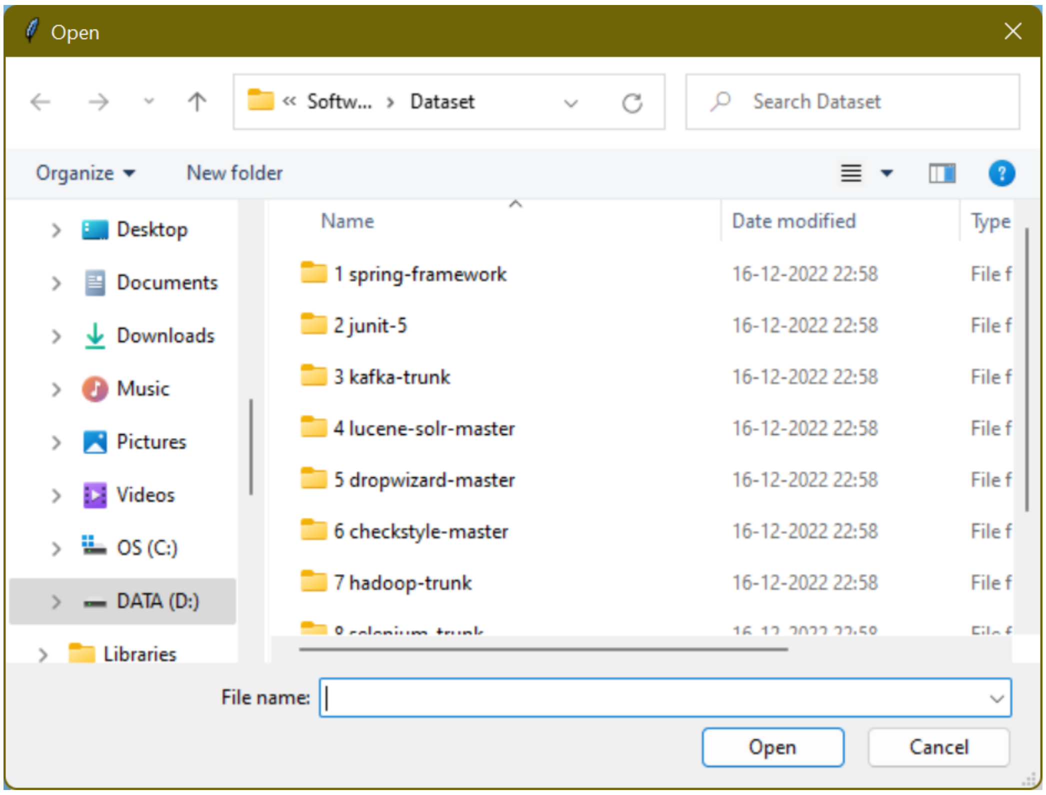
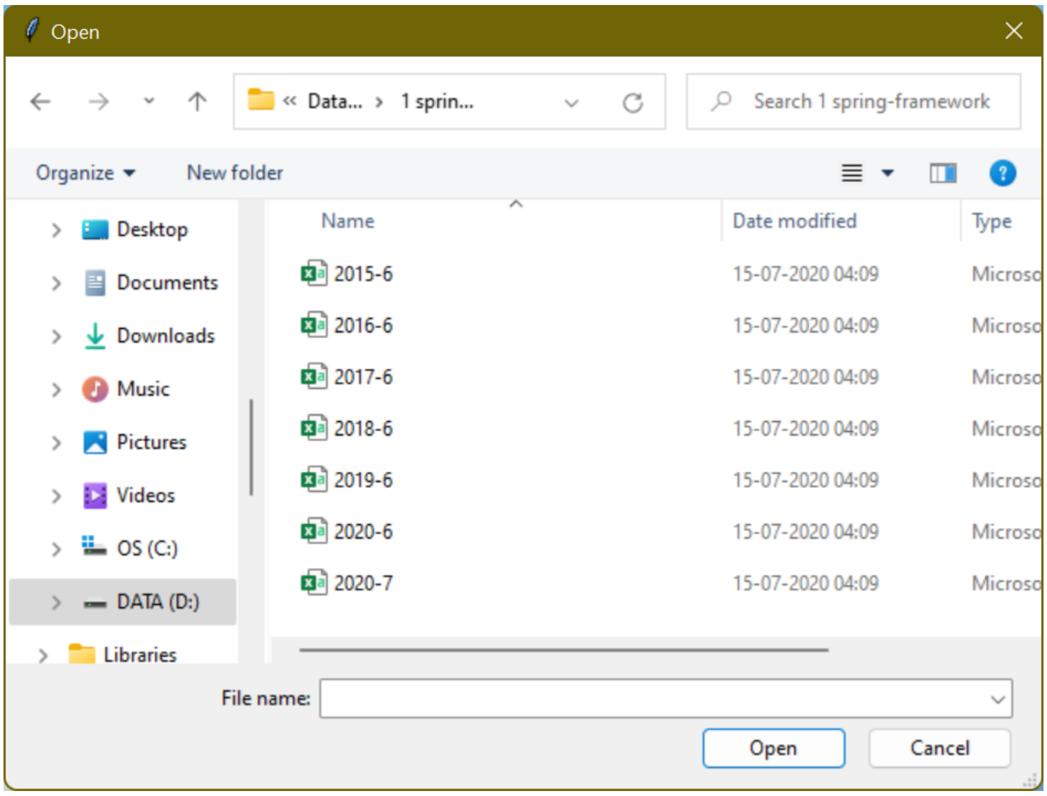

An Experimental Study for Software Quality Prediction with Machine Learning Methods
An Experimental Study for Software Quality Prediction with Machine Learning Methods
Upload Dataset
Preprocess Dataset
Features Selection Algorithms
Run Machine Learning Algorithms
Run CNN Algorithm
Comparison Graph

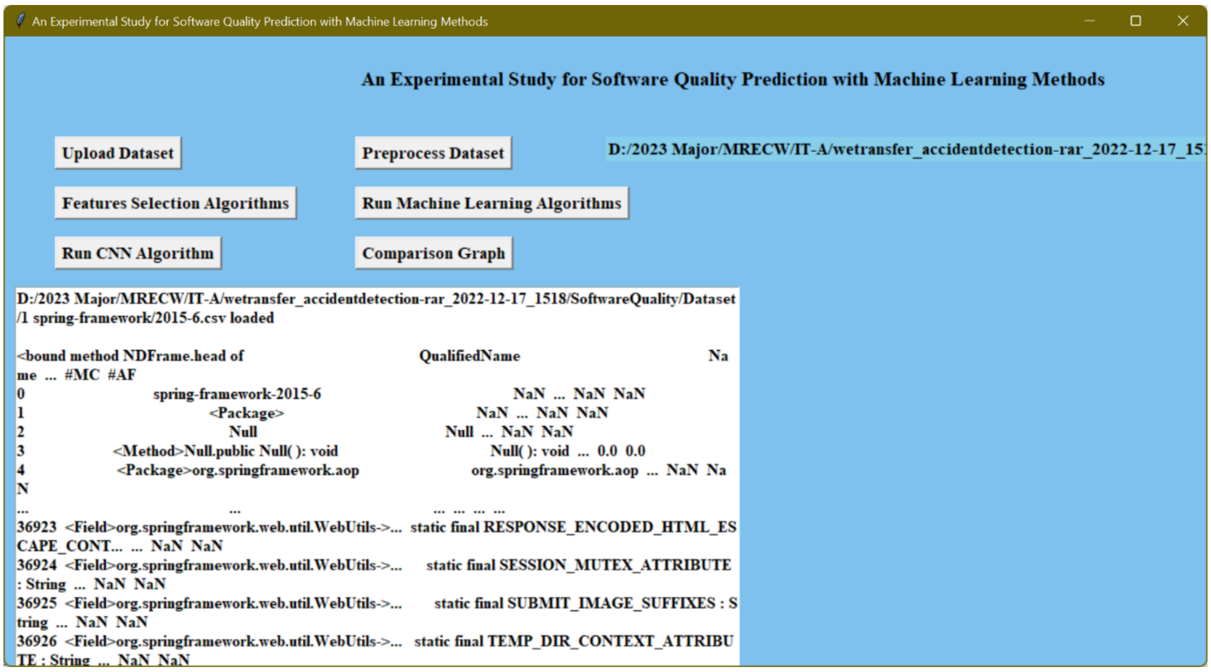
Fig. 3 An Experimental Study for Software Quality Prediction with Machine Learning Methods

In above screen click on 'Upload Dataset' button to and upload dataset.



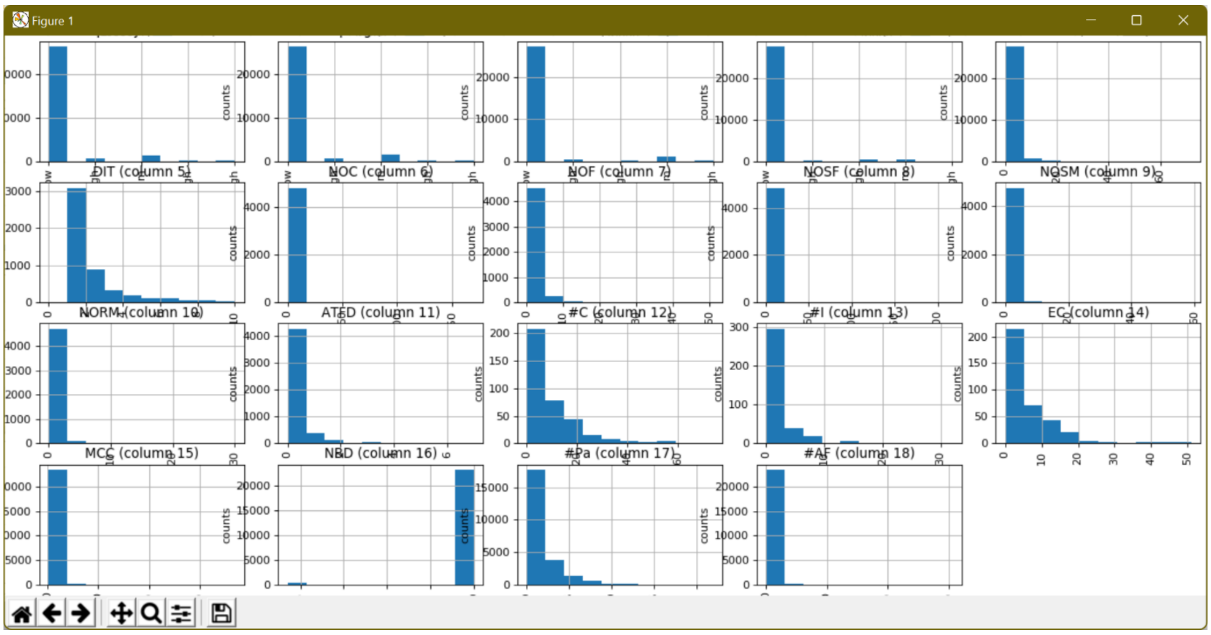
Name	Date modified	Type
1 spring-framework	16-12-2022 22:58	File f
2 junit-5	16-12-2022 22:58	File f
3 kafka-trunk	16-12-2022 22:58	File f
4 lucene-solr-master	16-12-2022 22:58	File f
5 dropwizard-master	16-12-2022 22:58	File f
6 checkstyle-master	16-12-2022 22:58	File f
7 hadoop-trunk	16-12-2022 22:58	File f
8 selenium-trunk	16-12-2022 22:58	File f


Open
« Data... › 1 sprin...
Search 1 spring-framework
Organize ▾
New folder
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
DATA (D:)
Libraries
Name
Date modified
Type
2015-6 15-07-2020 04:09 Microso
2016-6 15-07-2020 04:09 Microso
2017-6 15-07-2020 04:09 Microso
2018-6 15-07-2020 04:09 Microso
2019-6 15-07-2020 04:09 Microso
2020-6 15-07-2020 04:09 Microso
2020-7 15-07-2020 04:09 Microso
File name:
Open
Cancel


An Experimental Study for Software Quality Prediction with Machine Learning Methods
An Experimental Study for Software Quality Prediction with Machine Learning Methods
Upload Dataset
Preprocess Dataset
D:/2023 Major/MRECW/IT-A/wetransfer_accidentdetection-rar_2022-12-17_15
Features Selection Algorithms
Run Machine Learning Algorithms
Run CNN Algorithm
Comparison Graph

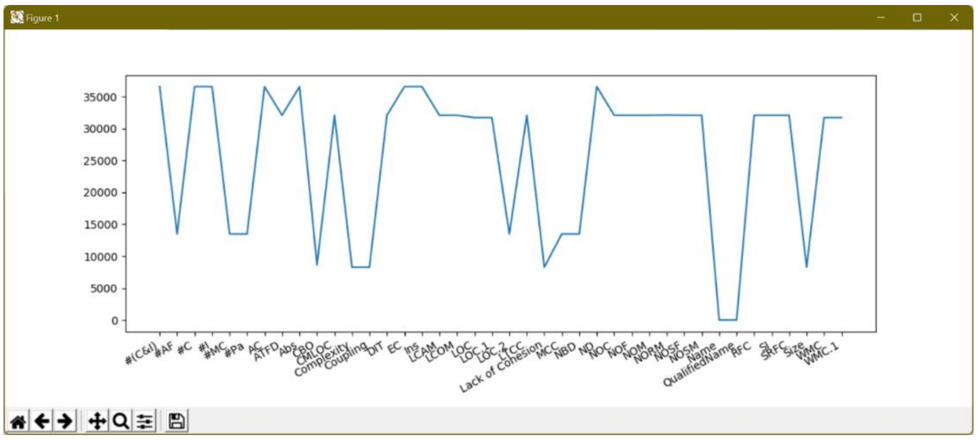
```
D:/2023 Major/MRECW/IT-A/wetransfer_accidentdetection-rar_2022-12-17_1518/SoftwareQuality/Dataset
/1 spring-framework/2015-6.csv loaded

<bound method NDFrame.head of                              QualifiedName                        Na
me ... #MC  #AF
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
N
...                               ...                    ... ... ... ...
36923  <Field>org.springframework.web.util.WebUtils->...  static final RESPONSE_ENCODED_HTML_ES
CAPE_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
tring ... NaN  NaN
36926  <Field>org.springframework.web.util.WebUtils->...  static final TEMP_DIR_CONTEXT_ATTRIBU
TE : String ... NaN  NaN
```

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

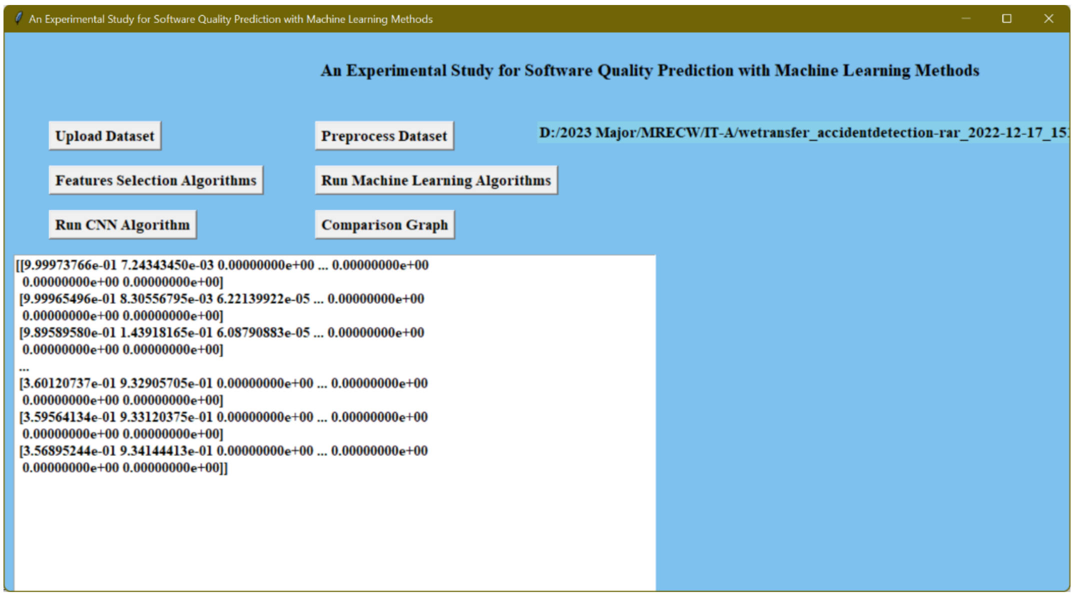


In above graph we can see each graph represents one column from dataset and from that columns its counting each distinct value from and plot in that graph for example in second graph NOC columns 3 different values and its plotting 3 different bars with count.

Thereafter it will display values from dataset and we can see dataset contains NAN (missing values) and string nonnumeric values and we need to replace all missing and non-numeric values with their count so click on ‘Preprocess Dataset’ button

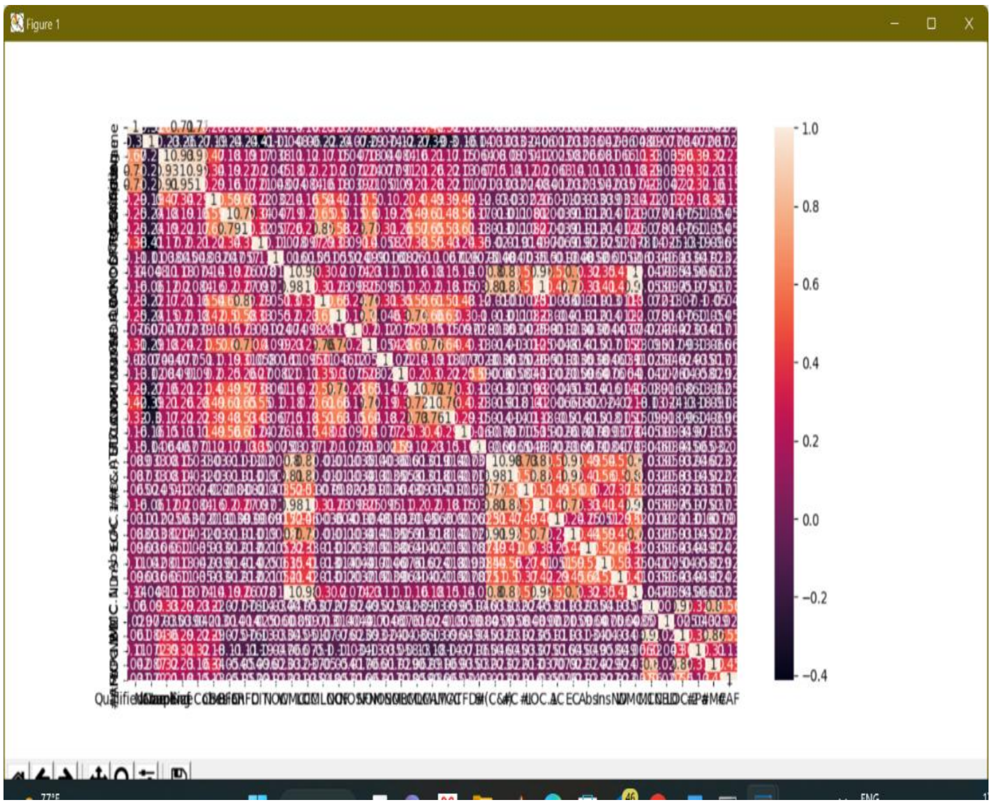


In above graph x-axis represents column names and y-axis represents total missing values counts in that column and now close above graph to get below screen

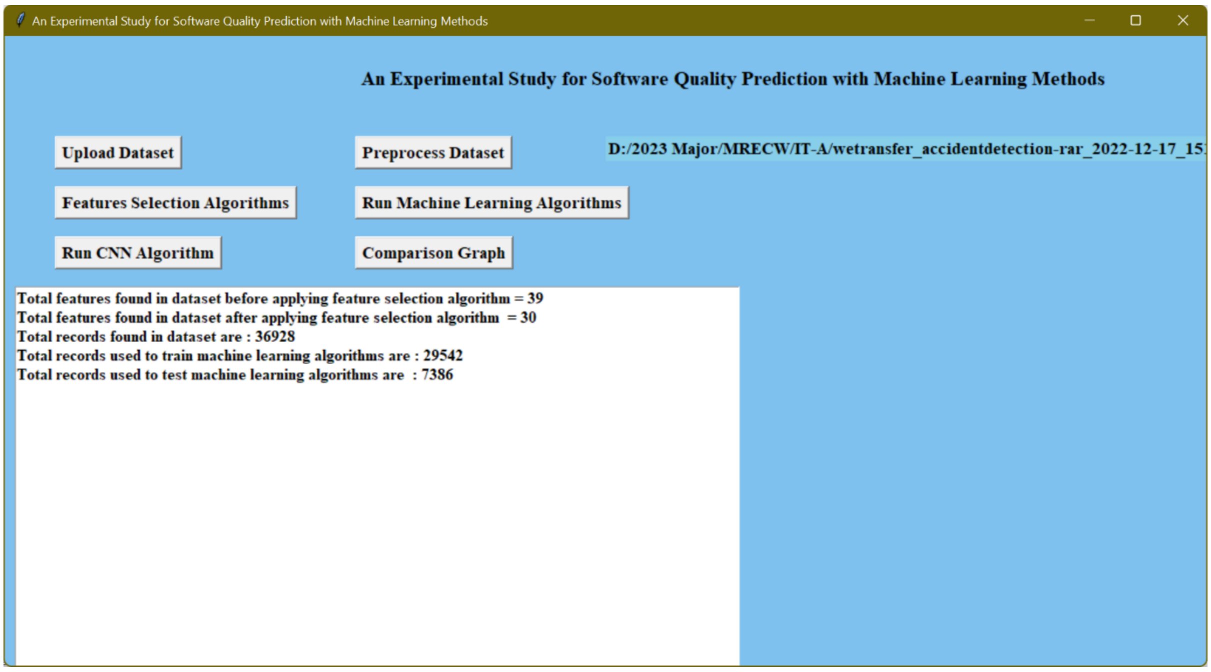

An Experimental Study for Software Quality Prediction with Machine Learning Methods
An Experimental Study for Software Quality Prediction with Machine Learning Methods
Upload Dataset
Preprocess Dataset
D:/2023 Major/MRECW/IT-A/wetransfer_accidentdetection-rar_2022-12-17_15
Features Selection Algorithms
Run Machine Learning Algorithms
Run CNN Algorithm
Comparison Graph

```
[[9.99973766e-01 7.24343450e-03 0.00000000e+00 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [9.99965496e-01 8.30556795e-03 6.22139922e-05 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [9.89589580e-01 1.43918165e-01 6.08790883e-05 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 ...
 [3.60120737e-01 9.32905705e-01 0.00000000e+00 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [3.59564134e-01 9.33120375e-01 0.00000000e+00 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [3.56895244e-01 9.34144413e-01 0.00000000e+00 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]]
```

In above screen all missing a string values are replace with numeric values and now click on ‘Features Selection Algorithms’ button to select important features from dataset and then split dataset into train and test part

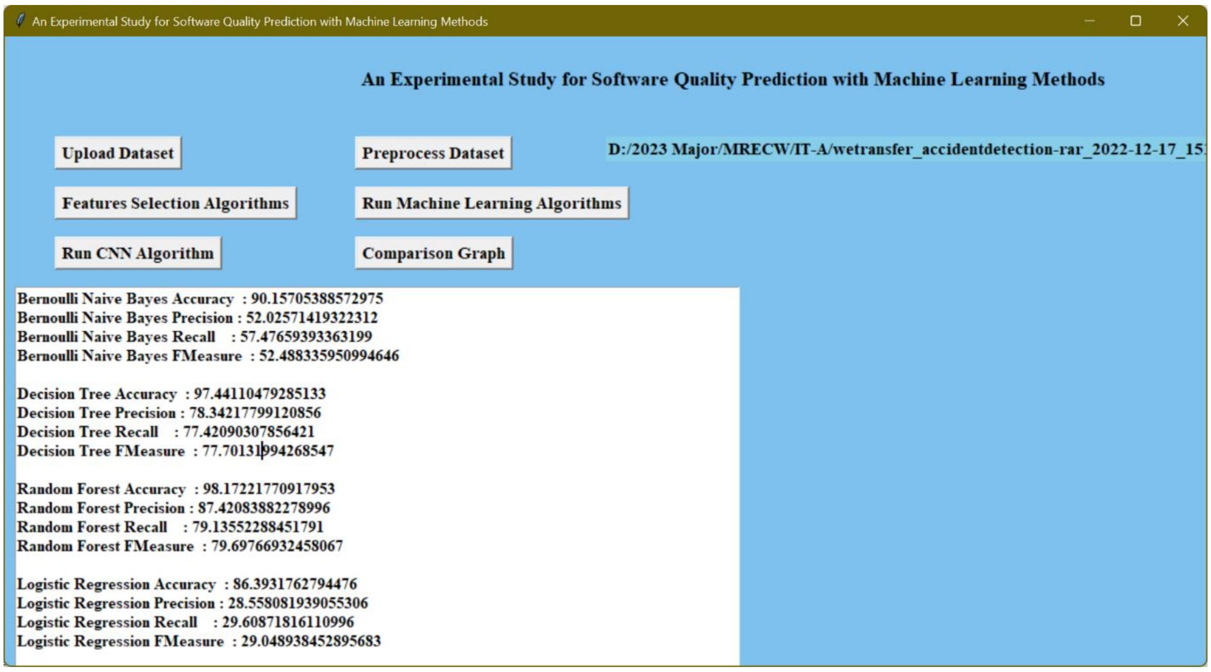


In above graph the box which contains value >0.5 will be consider as important attributes and now close above graph to get below screen

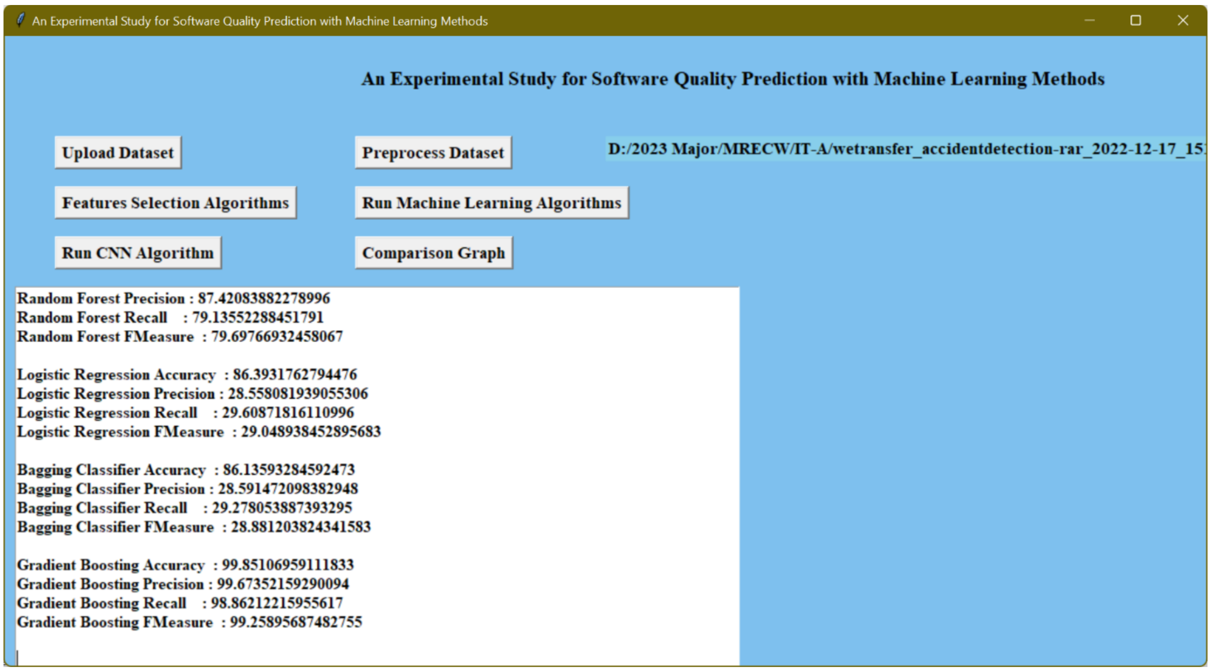

An Experimental Study for Software Quality Prediction with Machine Learning Methods
An Experimental Study for Software Quality Prediction with Machine Learning Methods
Upload Dataset
Preprocess Dataset
D:/2023 Major/MRECW/IT-A/wetransfer_accidentdetection-rar_2022-12-17_15
Features Selection Algorithms
Run Machine Learning Algorithms
Run CNN Algorithm
Comparison Graph

```
Total features found in dataset before applying feature selection algorithm = 39
Total features found in dataset after applying feature selection algorithm  = 30
Total records found in dataset are : 36928
Total records used to train machine learning algorithms are : 29542
Total records used to test machine learning algorithms are  : 7386
```

In above screen before applying feature selection algorithm dataset contains 39 features/columns and after applying PCA feature selection we got 30 important features and dataset contains 36928 records and application using 7386 records for testing and 29542 records for training and now both train and test dataset is ready and now click on 'Run Machine Learning Algorithms' button to run all machine learning algorithms


An Experimental Study for Software Quality Prediction with Machine Learning Methods
An Experimental Study for Software Quality Prediction with Machine Learning Methods
Upload Dataset
Preprocess Dataset
D:/2023 Major/MRECW/IT-A/wetransfer_accidentdetection-rar_2022-12-17_15
Features Selection Algorithms
Run Machine Learning Algorithms
Run CNN Algorithm
Comparison Graph

```
Bernoulli Naive Bayes Accuracy  : 90.15705388572975
Bernoulli Naive Bayes Precision : 52.02571419322312
Bernoulli Naive Bayes Recall    : 57.47659393363199
Bernoulli Naive Bayes FMeasure  : 52.488335950994646

Decision Tree Accuracy  : 97.44110479285133
Decision Tree Precision : 78.34217799120856
Decision Tree Recall    : 77.42090307856421
Decision Tree FMeasure  : 77.70131994268547

Random Forest Accuracy  : 98.17221770917953
Random Forest Precision : 87.42083882278996
Random Forest Recall    : 79.13552288451791
Random Forest FMeasure  : 79.69766932458067

Logistic Regression Accuracy  : 86.3931762794476
Logistic Regression Precision : 28.558081939055306
Logistic Regression Recall    : 29.60871816110996
Logistic Regression FMeasure  : 29.048938452895683
```

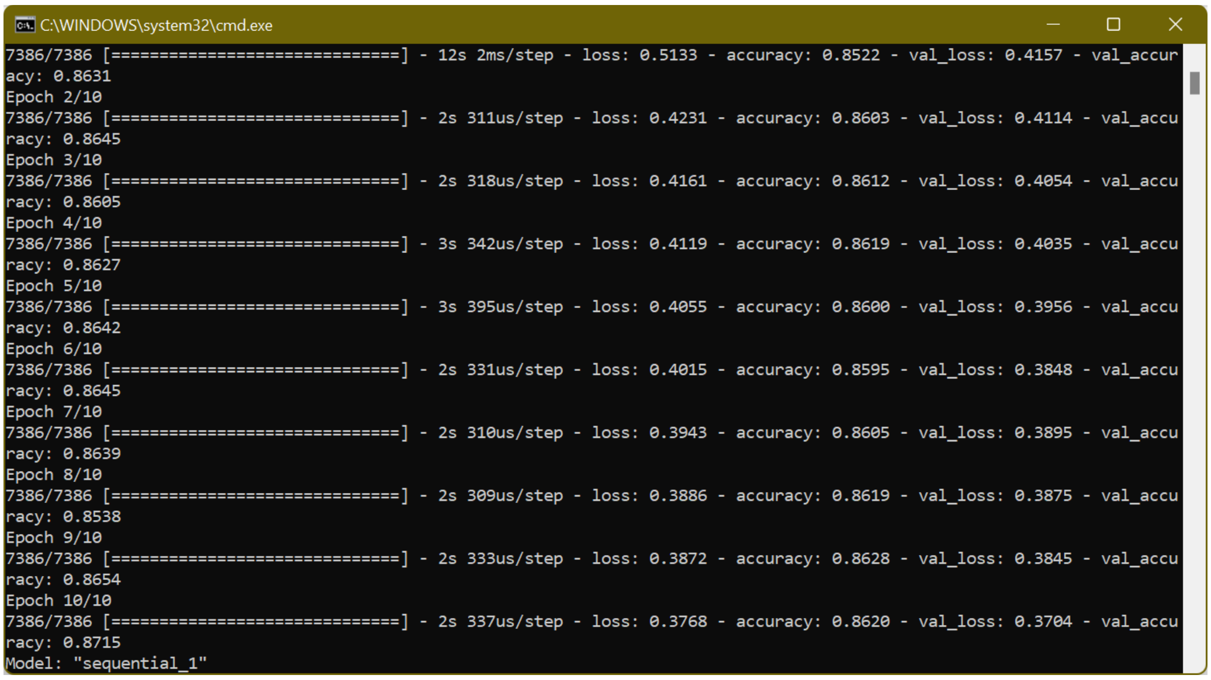

An Experimental Study for Software Quality Prediction with Machine Learning Methods
An Experimental Study for Software Quality Prediction with Machine Learning Methods
Upload Dataset
Preprocess Dataset
Features Selection Algorithms
Run Machine Learning Algorithms
Run CNN Algorithm
Comparison Graph

D:/2023 Major/MRECW/IT-A/wetransfer_accidentdetection-rar_2022-12-17_15

```
Random Forest Precision : 87.42083882278996
Random Forest Recall    : 79.13552288451791
Random Forest FMeasure  : 79.69766932458067

Logistic Regression Accuracy  : 86.3931762794476
Logistic Regression Precision : 28.558081939055306
Logistic Regression Recall    : 29.60871816110996
Logistic Regression FMeasure  : 29.048938452895683

Bagging Classifier Accuracy  : 86.13593284592473
Bagging Classifier Precision : 28.591472098382948
Bagging Classifier Recall    : 29.278053887393295
Bagging Classifier FMeasure  : 28.881203824341583

Gradient Boosting Accuracy  : 99.85106959111833
Gradient Boosting Precision : 99.67352159290094
Gradient Boosting Recall    : 98.86212215955617
Gradient Boosting FMeasure  : 99.25895687482755
```

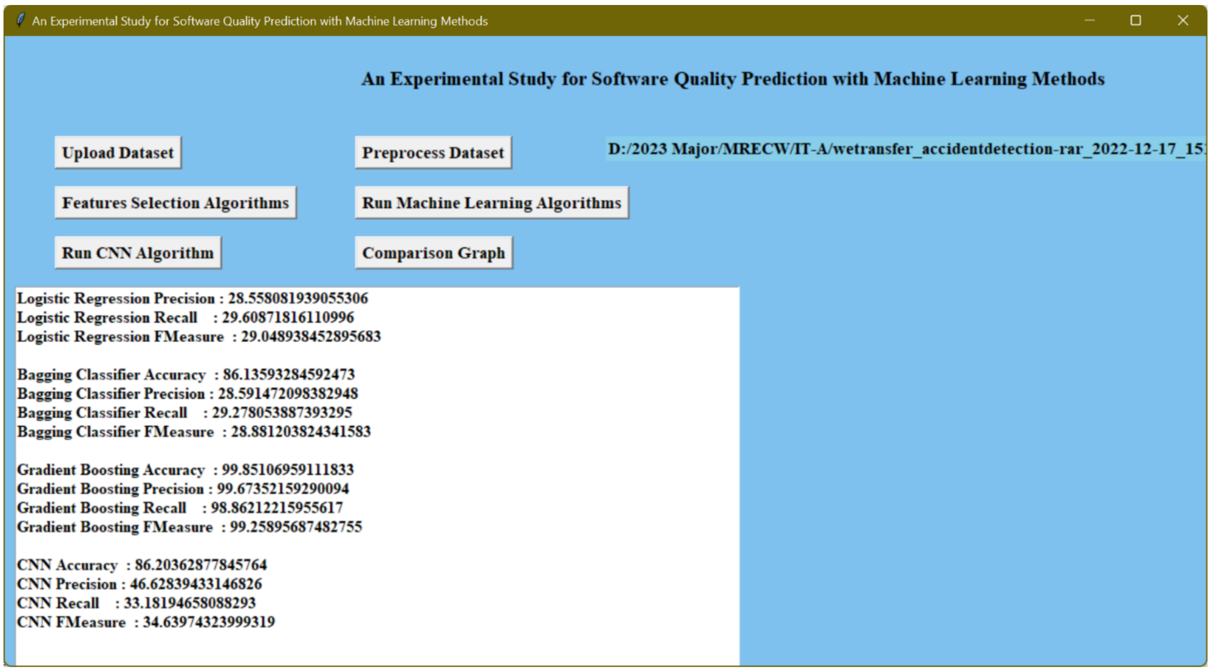
In above screen we can precision, recall, accuracy and f-measure for all machine learning algorithms

Now click on 'Run CNN Algorithm' button to run CNN algorithm and to get below screen



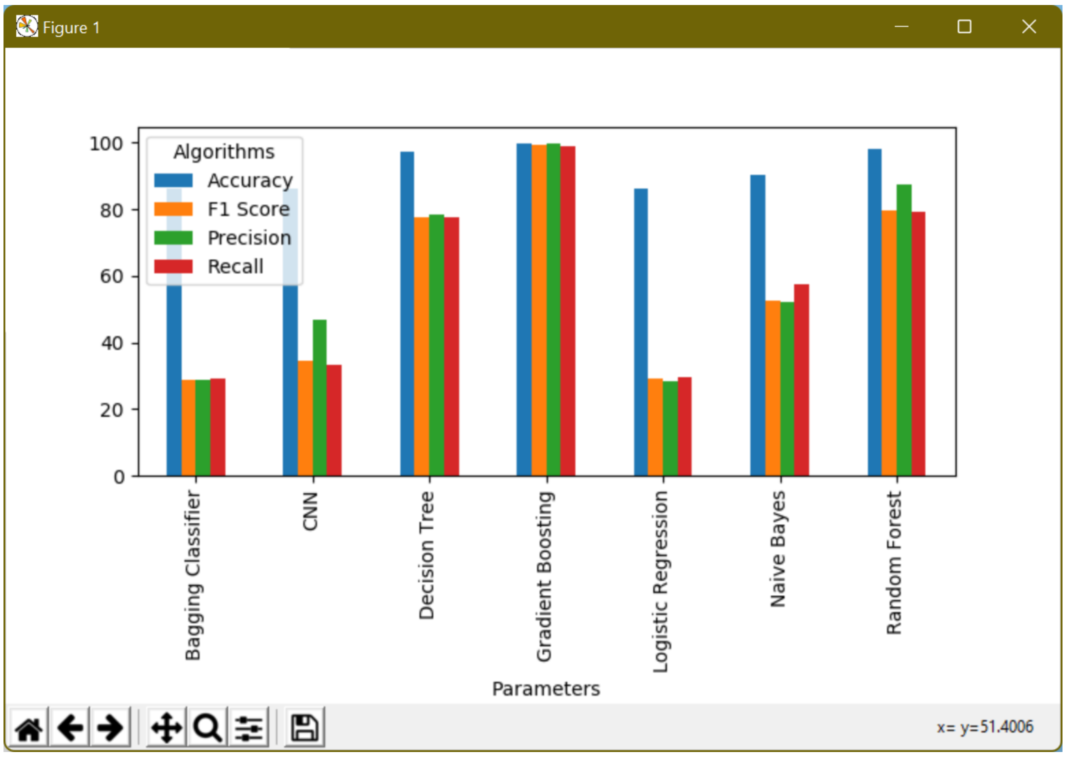
```
7386/7386 [==============================] - 12s 2ms/step - loss: 0.5133 - accuracy: 0.8522 - val_loss: 0.4157 - val_accur
acy: 0.8631
Epoch 2/10
7386/7386 [==============================] - 2s 311us/step - loss: 0.4231 - accuracy: 0.8603 - val_loss: 0.4114 - val_accu
racy: 0.8645
Epoch 3/10
7386/7386 [==============================] - 2s 318us/step - loss: 0.4161 - accuracy: 0.8612 - val_loss: 0.4054 - val_accu
racy: 0.8605
Epoch 4/10
7386/7386 [==============================] - 3s 342us/step - loss: 0.4119 - accuracy: 0.8619 - val_loss: 0.4035 - val_accu
racy: 0.8627
Epoch 5/10
7386/7386 [==============================] - 3s 395us/step - loss: 0.4055 - accuracy: 0.8600 - val_loss: 0.3956 - val_accu
racy: 0.8642
Epoch 6/10
7386/7386 [==============================] - 2s 331us/step - loss: 0.4015 - accuracy: 0.8595 - val_loss: 0.3848 - val_accu
racy: 0.8645
Epoch 7/10
7386/7386 [==============================] - 2s 310us/step - loss: 0.3943 - accuracy: 0.8605 - val_loss: 0.3895 - val_accu
racy: 0.8639
Epoch 8/10
7386/7386 [==============================] - 2s 309us/step - loss: 0.3886 - accuracy: 0.8619 - val_loss: 0.3875 - val_accu
racy: 0.8538
Epoch 9/10
7386/7386 [==============================] - 2s 333us/step - loss: 0.3872 - accuracy: 0.8628 - val_loss: 0.3845 - val_accu
racy: 0.8654
Epoch 10/10
7386/7386 [==============================] - 2s 337us/step - loss: 0.3768 - accuracy: 0.8620 - val_loss: 0.3704 - val_accu
racy: 0.8715
Model: "sequential_1"
```

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


An Experimental Study for Software Quality Prediction with Machine Learning Methods
An Experimental Study for Software Quality Prediction with Machine Learning Methods
Upload Dataset
Preprocess Dataset
D:/2023 Major/MRECW/IT-A/wetransfer_accidentdetection-rar_2022-12-17_15
Features Selection Algorithms
Run Machine Learning Algorithms
Run CNN Algorithm
Comparison Graph

```
Logistic Regression Precision : 28.558081939055306
Logistic Regression Recall    : 29.60871816110996
Logistic Regression FMeasure  : 29.048938452895683

Bagging Classifier Accuracy  : 86.13593284592473
Bagging Classifier Precision : 28.591472098382948
Bagging Classifier Recall    : 29.278053887393295
Bagging Classifier FMeasure  : 28.881203824341583

Gradient Boosting Accuracy  : 99.85106959111833
Gradient Boosting Precision : 99.67352159290094
Gradient Boosting Recall    : 98.8621221595561 7
Gradient Boosting FMeasure  : 99.25895687482755

CNN Accuracy  : 86.20362877845764
CNN Precision : 46.62839433146826
CNN Recall    : 33.18194658088293
CNN FMeasure  : 34.63974323999319
```

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.

5. Conclusion

Successful implementation of a software product entirely depends on the quality of the software developed. However, prediction of the quality of a software product prior to its implementation in real-world applications presents significant challenges to the software developer during the process of development. A limited spectrum of research in this area has been reported in the literature as of today. We have experimented with recent algorithms that support multi-class classification. The accuracies achieved by using these algorithms are impressive as compared to existing models. In comparison to previous directly comparable studies, acceptable level multiclass quality prediction could be achieved. In future there is a possibility of testing more datasets with feature selection-based advanced machine learning algorithms

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. Bhattacherjee, "Software quality estimation sing 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), 2016.