MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad) Gandipet, Hyderabad – 500075, Telangana.

CERTIFICATE

This is to certify that the project entitled "**Software Fault Prediction Using Machine Learning**" is being submitted by **MADDURI SAI MALLESH** bearing Roll No. **20261A05F2** in partial fulfilment of the requirements for the Mini Project (CS704PC) in **COMPUTER SCIENCE AND ENGINEERING** is a record of bonafide work carried out by him. The results of the investigations enclosed in this report have been verified and found satisfactory.

Project Guide	Head of the Department
Mr. R Mohan Krishna	**Dr. C.R.K .Reddy**
Assistant Professor	Professor, Dept. of CSE

External Examiner

DECLARATION

This is to certify that the work reported in the project titled “Software Defect Detection” is a record of work done by me in the Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad.
No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the work done entirely by me and not copied from any other source.

MADDURI SAI MALLESH
20261A05F2

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success. They have been a guiding light and source of inspiration towards the completion of the project.

I would like to express my sincere thanks to **Dr. G. Chandra Mohan Reddy, Principal MGIT,** providing the working facilities in college.

I wish to express my sincere thanks and gratitude to **Dr. C.R.K Reddy, Professor and HoD,** Department of CSE, MGIT, for all the timely support and valuable suggestions during the period of project.

I am extremely thankful to **Dr. C.R.K. Reddy, Professor and HoD, and Dr. Ratnaraju, Assistant Professor,** Department of CSE, MGIT, mini project coordinators for their encouragement and support throughout the project.

I am extremely thankful and indebted to my internal guide **Mr. R Mohan Krishna, Assistant Professor,** Department of CSE, for her constant guidance, encouragement and moral support throughout the project.

Finally, I would also like to thank all the faculty and staff of CSE Department who helped me directly or indirectly, for completing this project.

Madduri Sai Mallesh

20261A05F2

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Fault-prediction techniques are aiming towards prediction of the software modules that are faulty so that it could be beneficial in the upcoming phases of software development. Difference performance criteria are being employed in order to boost the performance of the already existing ways. However, the main issue is the perspective of compiling their performances which is ignored constantly. Classification is the most used technique that is being used for the exclusion of faulty from non-faulty modules. Machine-learning techniques are used to find the defect, fault, ambiguity, and bad smell to accomplish quality, maintainability, and reusability in software. Software fault prediction techniques are used to predict software faults by using statistical techniques. However, Machine-learning techniques are also valuable in detecting software fault. Here we Introduce an overview of software fault prediction using machine-learning techniques like Random Forest and SVM models to predict the fault.

1. Introduction

In the past decade, humans have progressively focused on software-based systems in which software quality is regarded as the most critical element in user functionality. Because of the vast production of application software, software quality remains an unresolved problem that gives inadequate output for industrial and private applications. Designs of defect prediction are commonly utilized by industries and Such models help in predicting faults, estimating effort testing software reliability, hazard analysis, etc. during the growth stage. A supervised machine learning predictive algorithm is consumed with the predefined collection of training data. The algorithm then gains expertise from the training dataset and produces rules for predicting the class label for a new data set.

Learning phases consists to use mathematical algorithms to generate and strengthen the predictor function. Training data used in this process has an attribute input value and its defined output value. The expected ML algorithm quality is compared with the often known output. This is repeated in many iterations of training data until the optimal prediction accuracy is reached or the upper limit number of loops is finished. In the field of unsupervised learning algorithms, the class label output value is not known in data. Alternatively, a cluster of data loads the software, and the algorithm identifies a pattern and relationships within it.

Software quality can be enhanced by predicting defect modules. Defect prediction is the method of designing models that are utilized in the initial stages of the process to detect defective systems such as units or classes. This can be achieved by classifying the modules as defect prone or not. Different methods are used to identify the classification module, the most common of which is support vector classifier (SVC), random forest, naive bayes, decision trees (DT), neural networks.

1.1 Problem Definition

The overarching problem is the persistent challenge of ensuring software quality in the past decade, with a focus on defect prediction models using supervised machine learning algorithms. Despite the increasing importance of software quality, there is an ongoing issue of inadequate output in industrial and private applications. The proposed solution involves designing and implementing predictive models to detect defective software employing algorithms such as support vector classifier, random forest, naive bayes, decision trees, and neural networks. The process includes iterative learning phases with predefined training data to optimize prediction accuracy and enhance overall software quality.

1.2 Existing System

In the context of software defect prediction, assume that the existing model is based on a Convolutional Neural Network (CNN). CNNs are a class of deep learning algorithms commonly associated with image recognition but have been adapted for various sequential data tasks, including software quality improvement. In this scenario, the CNN is likely applied to analyze patterns and features within the input data, which may represent characteristics of software modules. The CNN model would undergo a training phase using labeled datasets, where the network learns to identify distinctive patterns associated with defect-prone or non-defective modules. The convolutional layers in the CNN are adept at capturing hierarchical representations of input data, making them suitable for tasks where spatial relationships are crucial. The model's effectiveness would be evaluated based on its ability to accurately predict defect-prone modules in unseen data, contributing to the broader goal of enhancing software quality through advanced machine learning techniques.

1.3 Proposed System

Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. It is generally utilized in characterization issues. In the SVM calculation, we plot every datum thing as a point in n-dimensional space (where n is number of highlights you have) with the estimation of each element being the estimation of a specific arrange. At that point, we perform order by finding the hyper-plane that separates the two classes quite well. Bolster Vectors are essentially the co-ordinates of individual perception. The SVM classifier is a wilderness which best isolates the two classes (hyper- plane/line)

Working with Support vector machine in Python:

```
#Import Library
```

```
# Import other necessary

libraries like pandas, numpy…

from sklearn import tree
```

# Assumed you have, X (predictor) and Y (target) for training data set and x_test (predictor) of test_dataset

```
# Create tree object

model = SVC (criterion='gini') # for classification # model =

SVR() forregression
```

# Train the model using the training sets and check score

```
model.fit(X,y) model.score(X,y) #Predict Output
predicted = model.predict(x_test)
```

Random Forest

Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision *trees*, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

How Random Forest works

The following are the basic steps involved in performing the random forest algorithm

1. Pick N random records from the dataset.
2. Build a decision tree based on these N records.
3. Choose the number of trees you want in your algorithm and repeat steps 1 and 2.
4. For classification problem, each tree in the forest predicts the category to which the new

record belongs. Finally, the new record is assigned to the category that wins the majority vote.

1.4 Requirements Specification

1.4.1Software Requirements

- Operating system : The web application is platform-independent and compatible with various operating systems, including but not limited to Windows, macOS, and Linux.

• Frontend Development	:	ReactJs.
• Backend Development	:	Python
• AI Integration	:	GPT-3.5 Turbo API.

1.4.2Hardware Requirements:

• Processor	:	Intel Core i5
• RAM	:	4 GB (min)
• Hard Disk	:	256 GB SSD
• Internet connectivity	:	15Mbps (min)
• Web Browser	:	Google Chrome, Mozilla Firefox and Safari

Modules

1. Data Collection
2. Data Pre-Processing
3. Feature Extraction
4. Evaluation Model

Data Collection

Data used in this paper is a software data of JM1. This step is concerned with selecting the subset of all available data that you will be working with. ML problems start with data preferably, lots of data (examples or observations) for which you already know the target answer. Data for which you already know the target answer is called *labelled data*.

Data Pre-Processing

Organize your selected data by formatting, cleaning and sampling from it.
Three common data pre-processing steps are:

- Formatting: The data you have selected may not be in a format that is suitable for you to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and you would like it in a relational database or a text file.
- Cleaning: Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be removed. Additionally, there may be sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely.
- Sampling: There may be far more selected data available than you need to work with. More data can result in much longer running times for algorithms and larger computational and memory requirements. You can take a smaller representative sample of the selected data that may be much faster for exploring and prototyping solutions before considering the whole datasets

Feature Extraction

Next thing is to do Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes. Finally, our models are trained using Classifier algorithm. We use classify module on Natural Language Toolkit library on Python. We use the labelled dataset gathered. The rest of our labelled data will be used to evaluate the models. Some machine learning algorithms were used to classify pre-processed data. The chosen classifiers were Random forest. These algorithms are very popular in text classification tasks.

Evaluation Model

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and over fitted models.
Performance of each classification model is estimated base on its averaged. The result will be in the visualized form. Representation of classified data in the form of graphs.
Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

Proposed Approach Steps

1. First, we take software dataset.
2. Filter dataset according to requirements and create a new dataset which has attribute according to analysis to be done
3. Perform Pre-Processing on the dataset
4. Split the data into training and testing
5. Train the model with training data then analyze testing dataset over classification algorithm
6. Finally you will get results as accuracy metrics

2.Literature Survey

1. Software defect prediction techniques using metrics based on neural network classifiers. Cluster Computing, by Jayanthi, R. and Florence, L.

Software industries strive for software quality improvement by consistent bug prediction, bug removal and prediction of fault-prone module. This area has attracted researchers due to its significant involvement in software industries. Various techniques have been presented for software defect prediction. Recent researches have recommended data-mining using machine learning as an important paradigm for software bug prediction. state-of-art software defect prediction task suffer from various issues such as classification accuracy. However, software defect datasets are imbalanced in nature and known fault prone due to its huge dimension. To address this issue, here we present a combined approach for software defect prediction and prediction of software bugs. Proposed approach delivers a concept of feature reduction and artificial intelligence where feature reduction is carried out by well-known principle component analysis (PCA) scheme which is further improved by incorporating maximum-likelihood estimation for error reduction in PCA data reconstruction. Finally, neural network based classification technique is applied which shows prediction results. A framework is formulated and implemented on NASA software dataset where four datasets i.e., KC1, PC3, PC4 and JM1 are considered for performance analysis using MATLAB simulation tool. An extensive experimental study is performed where confusion, precision, recall, classification accuracy etc. parameters are computed and compared with existing software defect prediction techniques. Experimental study shows that proposed approach can provide better performance for software defect prediction.

2. Integrated approach to software defect prediction. IEEE Access, 5, pp.21524-21547 by Felix, E.A. and Lee, S.P.

Software defect prediction provides actionable outputs to software teams while contributing to industrial success. Empirical studies have been conducted on software defect prediction for both crossproject and within-project defect prediction. However, existing studies have yet to demonstrate a method of predicting the number of defects in an upcoming product release. This paper presents such a method using predictor variables derived from the defectcceleration, namely, the defect density, defect velocity, and defect introduction time, and determines the correlation of each predictor variable with the number of defects.

We report the application of an integrated machine learning approach based on regression models constructed from these predictor variables. An experiment was conducted onten different data sets collected from the PROMISE repository, containing 22 838 instances. The regression model constructed as a function of the average defect velocity achieved an adjusted R-square of 98.6%, with a p-value of < 0.001. The average defect velocity is strongly positively correlated with the number of defects, with a correlation coefficient of 0.98. Thus, it is demonstrated that this technique can provide a blueprint for program testing to enhance the effectiveness of software development activities.

3.Multiple kernel ensemble learning for software defect prediction by Wang, T., Zhang, Z., Jing, X., Zhang, L.

Software defect prediction aims to predict the defect proneness of new software modules with the historical defect data so as to improve the quality of a software system. Software historical defect data has a complicated structure and a marked characteristic of class-imbalance; how to fully analyze and utilize the existing historical defect data and build more precise and effective classifiers has attracted considerable researchers' interest from both academia and industry. Multiple kernel learning and ensemble learning are effective techniques in the field of machine learning. Multiple kernel learning can map the historical defect data to a higher- dimensional feature space and make them express better, and ensemble learning can use a series of weak classifiers to reduce the bias generated by the majority class and obtain better predictive performance. In this paper,Xu, Z., Xuan, J., Liu, J., Cui, X.: MICHAC: defect prediction via feature selection based on maximal information coefficient with hierarchical agglomerative clustering. In: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Suita, pp. 370–381 (2016). e advantages of both multiple kernel learning and ensemble learning. We thus propose a multiple kernel ensemble learning (MKEL) approach for software defect classification and prediction. Considering the cost of risk in software defect prediction, we design a new sample weight vector updating strategy to reduce the cost of risk caused by misclassifying defective modules as non-defective ones. We employ the widely used NASA MDP datasets as test data to evaluate the performance of all compared methods; experimental results show that MKEL outperforms several representative state-of-the-art defect prediction methods.

4.Defect prediction via feature selection based on maximal information coefficient with hierarchical agglomerative clustering by Xu, Z., Xuan, J., Liu, J., Cui, X.: MICHAC.

Defect prediction aims to estimate software reliabil- ity via learning from historical defect data. A defect prediction method identifies whether a software module is defect-prone or not according to metrics that are mined from software pro- jects. These metric values, also known as features, may involve irrelevance and redundancy, which will hurt the performance of defect prediction methods. Existing work employs feature selection to preprocess defect data to filter out useless features. In this paper, we propose a novel feature selection framework, MICHAC, short for defect prediction via Maximal Infor- mation Coefficient with Hierarchical Agglomerative Cluster- ing. MICHAC consists of two major stages.

First, MICHAC employs maximal information coefficient to rank candidate features to filter out irrelevant ones; second, MICHAC groups features with hierarchical agglomerative clustering and selects one feature from each resulted group to remove redundant features. We evaluate our proposed method on 11 widely- studied NASA projects and four open-source AEEEM projects using three different classifiers with four performance metrics (precision, recall, F-measure, and AUC). Comparison with five existing methods demonstrates that MICHAC is effective in selecting features in defect prediction.

5.Effective multi-objective naïve Bayes learning for cross-project defect prediction.by Ryu, D., Baik, J

A piece of maximum information, datacorrelation-based technique is proposed to tackle this problem. Recently, Duksan et al.[5] discussed the unbalanced nature of software defect results, and very few occurrences display attributes that belong to the defective class during the prediction process.

6.Software defect prediction model based on LLE and SVM.by Shan C., Chen B., Hu C., Xue J., Li N.

This phase creates a reduction of efficiency in software industries and therefore involves a specific classification scheme. To resolve this problem, the whole issue is transformed into an issue of multi-objective optimization, where a Multi goal system of learning is implemented by analyzing a varied cross-project environment. Shan et al.[6] “utilized a well-known methodology in machine learning, i.e. SVM (support vector machine). Besides, predictability in attributes is discussed through the diligence of a locally linear embedding strategy with a support vector classifier. SVM constraints are indeed configured with a tenfold cross-validation process and grid search scheme according to this approach”

7. A novel radial basis function neural network for discriminant analysis by Yang, Z.R.

Experimental analysis reveals that the LLE-SVM works well for detecting defects. Yang et al.[7] “proposed the Predicting Software Deficiencies using a neural network method in which the neural network concept is incorporated along with the Bayesian approach as a radial basis.

8.A software reliability prediction method based on software development process by K. Han, J.-H. Cao, S.-H. Chen, and W.-W. Liu,

The efficiency of the radial neural network can be enhanced by optimizing the weight update framework, using a single Gaussian and two Gaussian structures, while the motivation- minimization scheme is often employed for weight realization”. Han et al., [8] “as stated in the proposed software development based stable program quality estimation model.

S. No.	Year of Publication	Title	Technology and Methodology	Description	Merits and Demerits
1.	2021	Software defect prediction techniques using metrics based on neural network classifiers. Cluster Computing	Neural Network, Principal Component Analysis (PCA)	Feature reduction through PCA - Incorporates maximum-likelihood estimation for error reduction in PCA data reconstruction - Utilizes neural network-based classification	Limited details on the neural network architecture used
2.	2020	Integrated Approach to Software Defect Prediction	Machine Learning, Regression Models	Predicts the number of defects in an upcoming product release - Uses predictor variables derived from defect acceleration	Limited discussion on the datasets used for experimentation
3.	2019	Multiple Kernel Ensemble Learning for Software Defect Prediction	Multiple Kernel Learning, Ensemble Learning	Combines advantages of multiple kernel learning and ensemble learning - Introduces a sample weight vector updating strategy to reduce the cost of risk	Lacks detailed comparison with other state-of-the-art methods
4.	2018	Defect Prediction via Feature Selection Based on Maximal Information Coefficient with Hierarchical Agglomerative Clustering	Feature Selection, Maximal Information Coefficient, Hierarchical Agglomerative Clustering	Effectively selects features in defect prediction - Employs a two-stage process for feature selection	Limited discussion on the specific classifiers used for evaluation.

5.	2018	Effective Multi-Objective Naïve Bayes Learning for Cross-Project Defect Prediction	Maximum Information, Data Correlation, Naïve Baye	Addresses the unbalanced nature of software defect results - Proposes a multi-objective learning approach	Limited details on the specific multi-objective optimization scheme employed
6.	2015	Software Defect Prediction Model Based on LLE and SVM	Support Vector Machine (SVM), Locally Linear Embedding (LLE)	Utilizes SVM and LLE for defect prediction - Discusses predictability in attributes	Limited explanation of the SVM configuration and parameter tuning
7.	2015	Software Reliability Prediction Method Based on Software Development Process	Radial Neural Network, Gaussian Structures	Enhances radial neural network efficiency through weight optimization - Proposes a stable program quality estimation model	Limited details on the weight optimization framework

Table 1: Literature Survey

3. System Design

3.1System Architecture:-

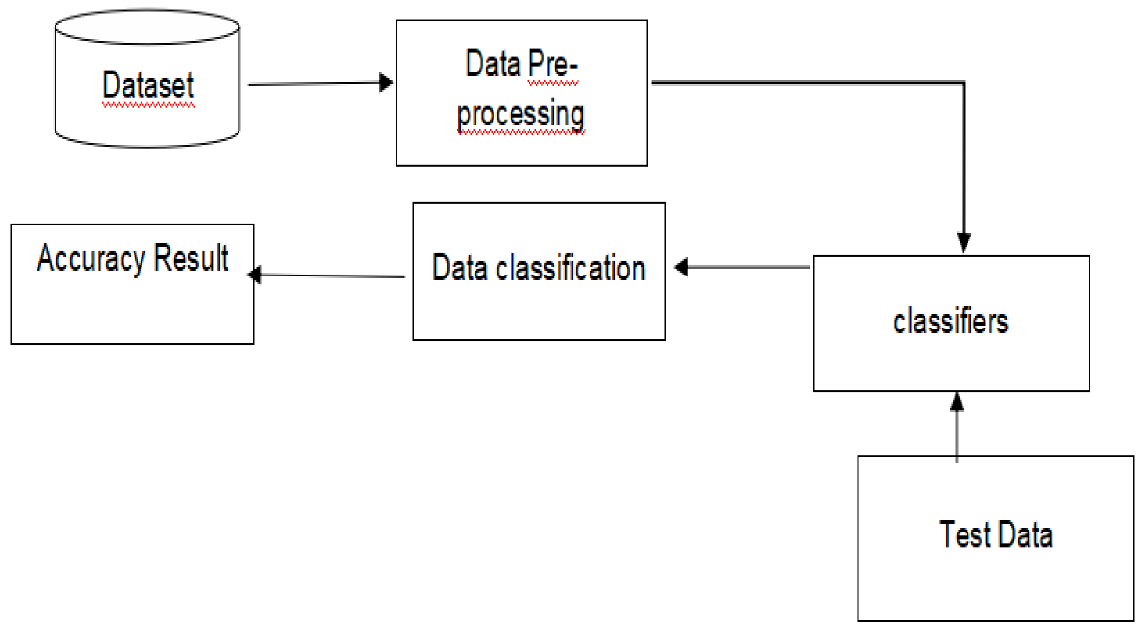


Figure 1: System Architechture

In this first we train the system with the dataset which was prepocessed using different classifier moldels and then we test the new input to fing the class lable. And then we find the accuracy result.

3.2UML Diagrams

3.2.1Dataflow diagram:-

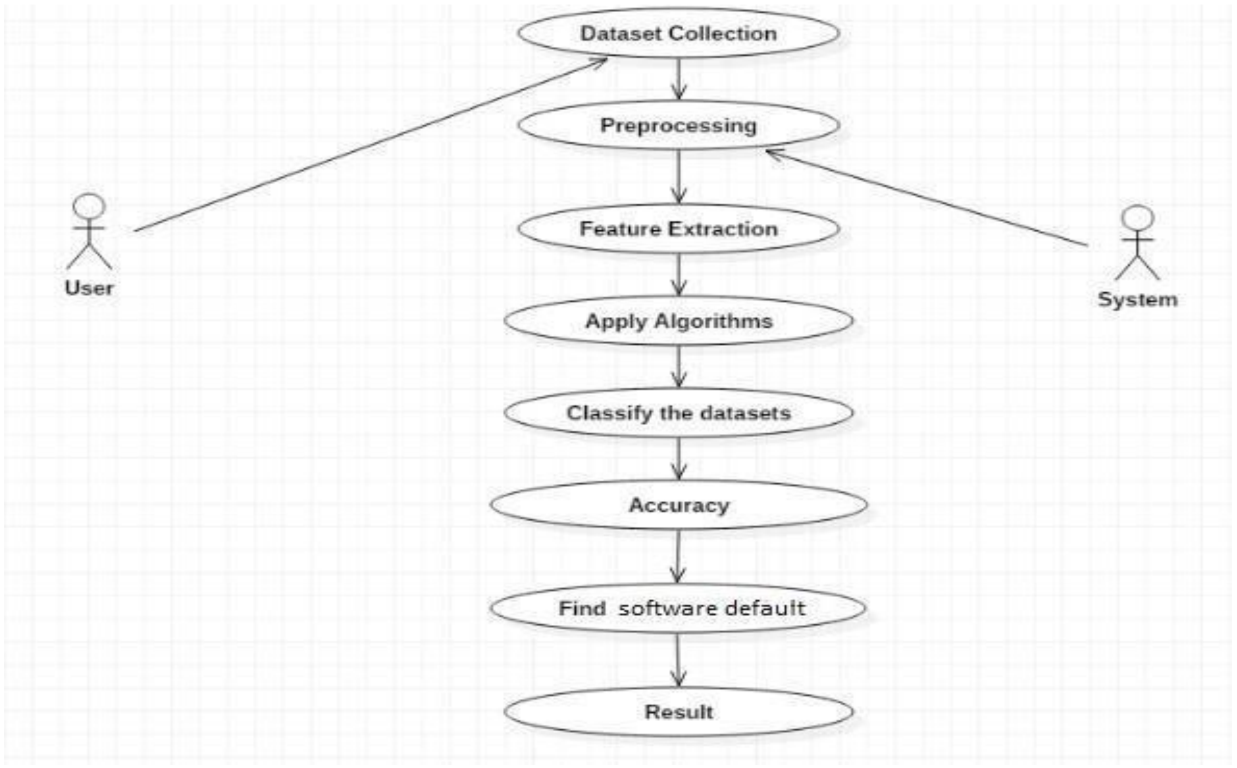


Figure 2: Data Flow Diagram

This data flow diagram describes about the flow of my project where its starts with data preprocesseing and gets trained and tested.

3.2.2Class Diagram

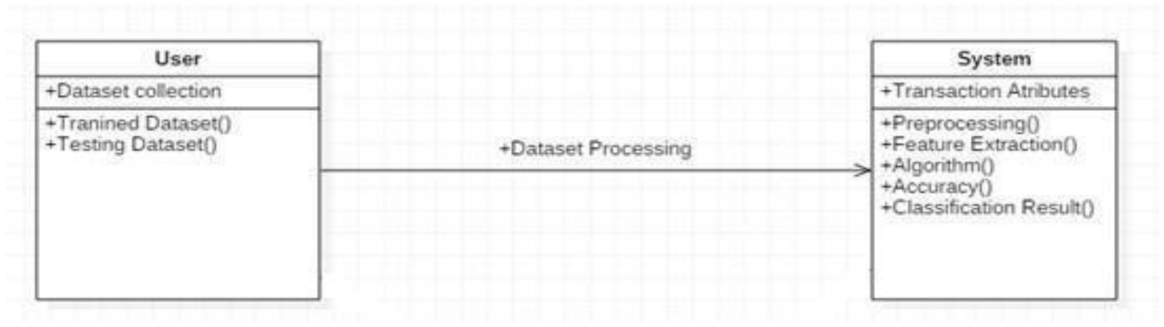


Figure 3: Class Diagram

This class diagram describes about the attributes and functions present in my project. And describes the cluster of functions which are used in this machine learning process.

3.2.3Sequence Diagram

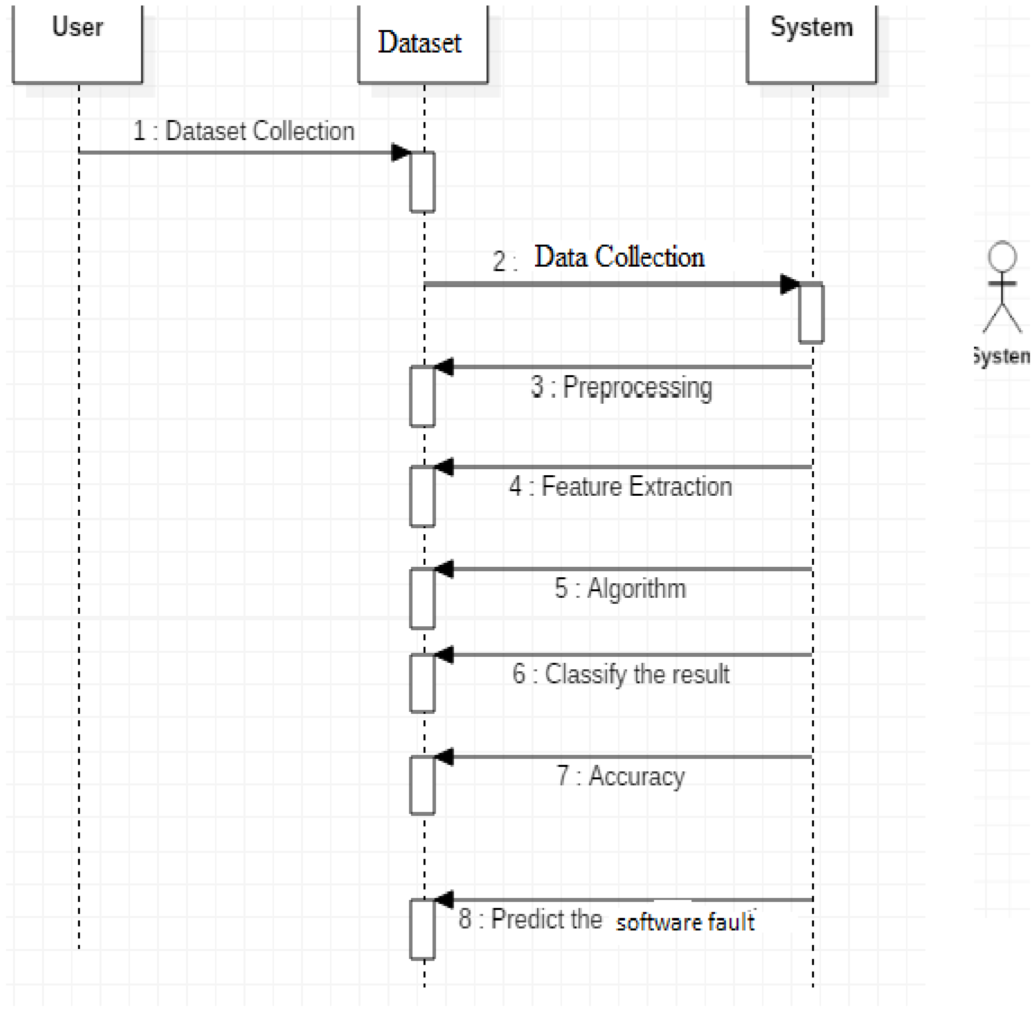


Figure 4: Sequence Diagram

This sequence diagram is mainly helpfull in describing the sequence of steps which are taken during the process with respect to the time. And gives us a rough idea about how it actually works.

3.2.4Activity Diagram

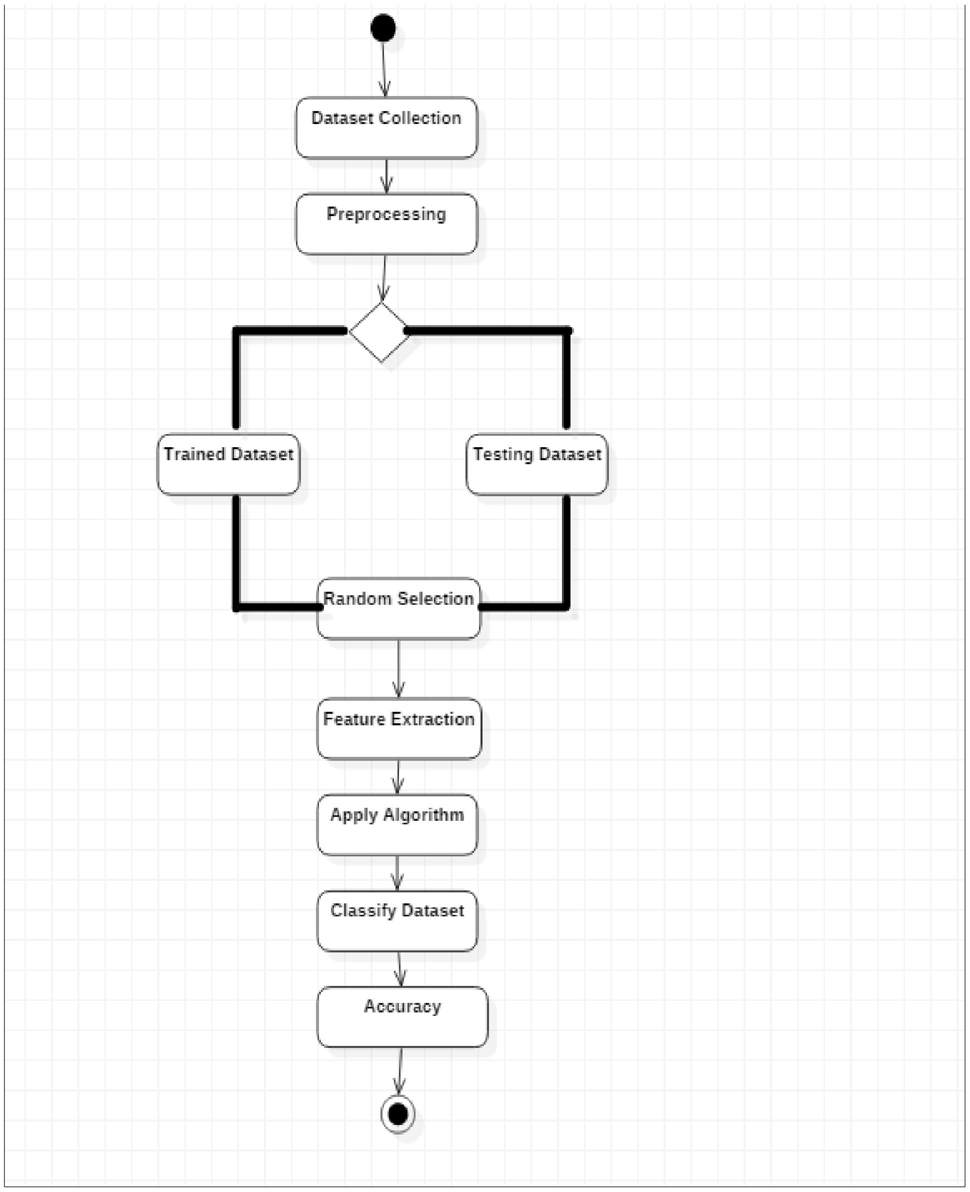


Figure 5: Activity Diagram

This activity diagram describes about the different steps involved while executing the process with the help of the flow chart.

4.Domain Specification

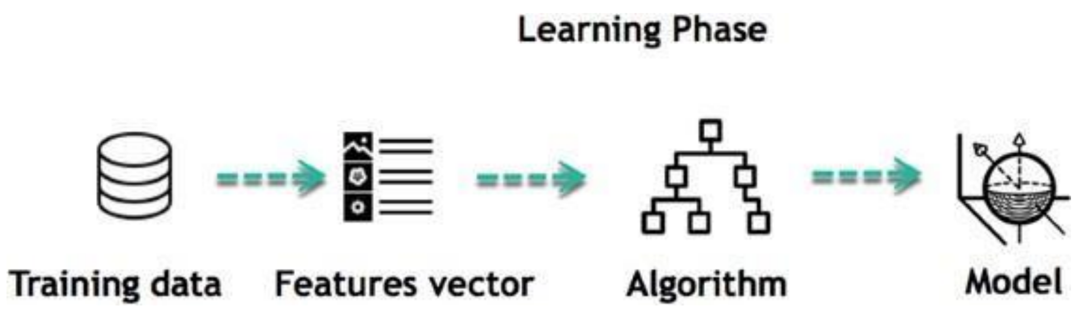
Machine Learning

Machine Learning is a system that can learn from example through self-improvement and without being explicitly coded by programmer. The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results.
Machine learning combines data with statistical tools to predict an output. This output is then used by corporate to makes actionable insights. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input, use an algorithm to formulate answers.
A typical machine learning tasks are to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendation.
Machine learning is also used for a variety of task like fraud detection, predictive maintenance, portfolio optimization, automatize task and so on.

How does Machine learning work?

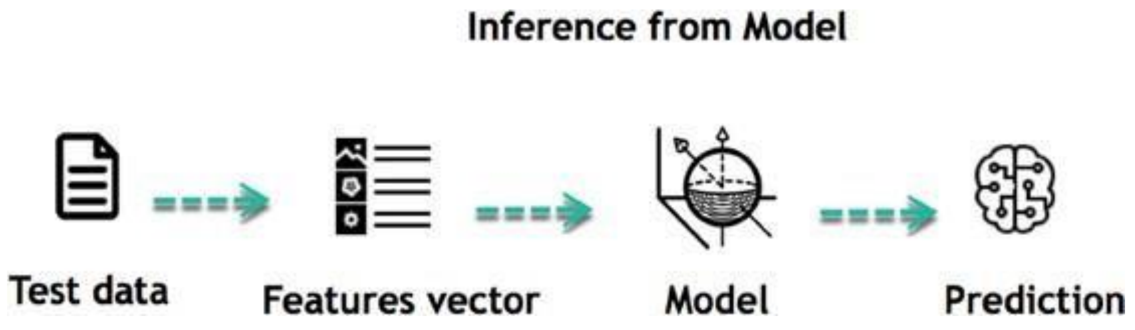
Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same.
To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.
The core objective of machine learning is the learning and inference. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the data. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a feature vector. You can think of a feature vector as a subset of data that is used to tackle a problem.

The machine uses some fancy algorithms to simplify the reality and transform this discovery into a model. Therefore, the learning stage is used to describe the data and summarize it into a model.



For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model Inferring.

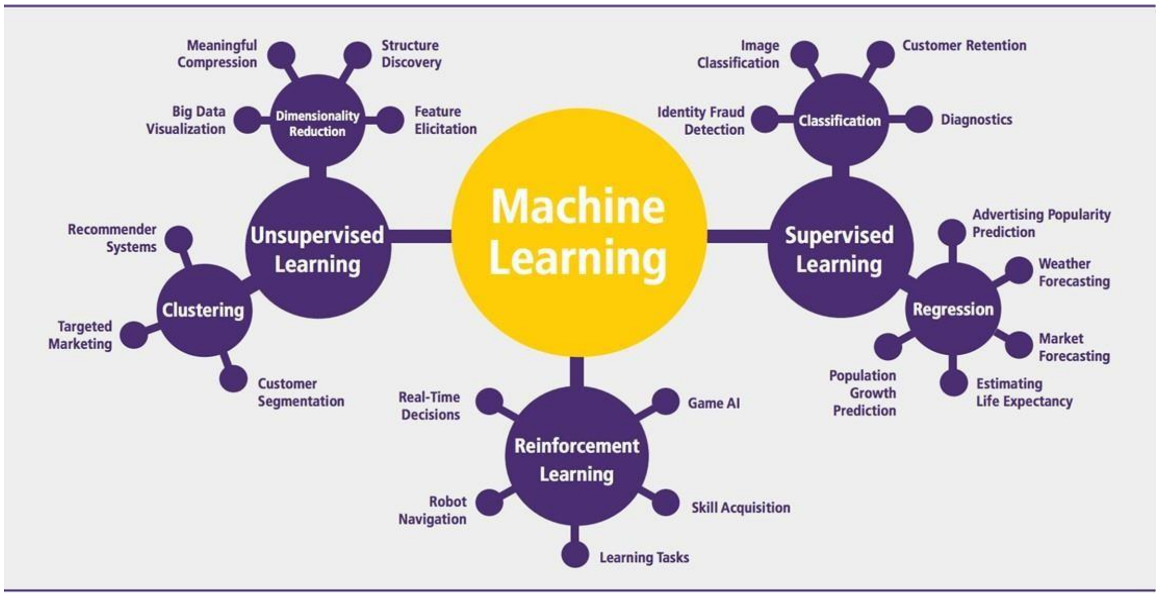
When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.



The life of Machine Learning programs is straightforward and can be summarized in the following points:

1. Define a question

2. Collect data
3. Visualize data
4. Train algorithm
5. Test the Algorithm
6. Collect feedback
7. Refine the algorithm
8. Loop 4-7 until the results are satisfying



Application of Machine learning Augmentation:

Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.

Automation:

Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.

Finance Industry

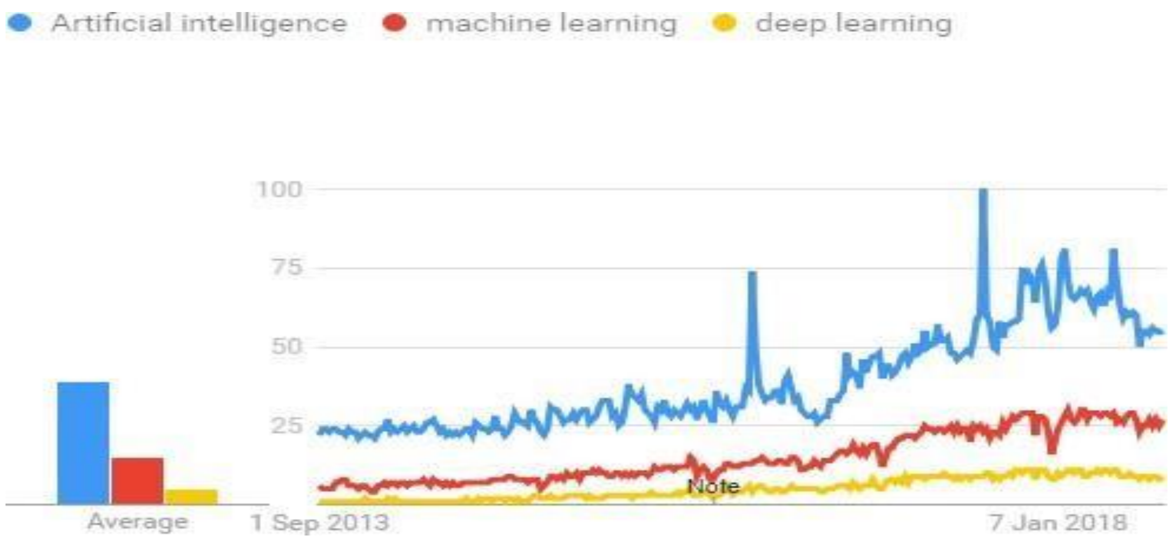
Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.

Government organization

The government makes use of ML to manage public safety and utilities. Take the example of China with the massive face recognition. The government uses Artificial intelligence to prevent jaywalker.

Healthcare industry

Healthcare was one of the first industry to use machine learning with image detection.



TensorFlow

The most famous deep learning library in the world is Google's TensorFlow. Google product uses machine learning in all of its products to improve the search engine, translation, image captioning or recommendations.

To give a concrete example, Google users can experience a faster and more refined the search with AI. If the user types a keyword a the search bar, Google provides a recommendation about what could be the next word.

Google wants to use machine learning to take advantage of their massive datasets to give users the best experience. Three different groups use machine learning:

- Researchers
- Data scientists
- Programmers.

They can all use the same toolset to collaborate with each other and improve their efficiency. Google does not just have any data; they have the world's most massive computer, so TensorFlow was built to scale. TensorFlow is a library developed by the Google Brain Team to accelerate machine learning and deep neural network research. It was built to run on multiple CPUs or GPUs and even mobile operating systems, and it has several wrappers in several languages like Python, C++ or Java.

TensorFlow Architecture

Tensor flow architecture works in three parts:

- Pre processing the data
- Build the model
- Train and estimate the model

It is called Tensor flow because it takes input as a multi-dimensional array, also known as tensors. You can construct a sort of flowchart of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.

This is why it is called TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

Where can Tensor flow run?

TensorFlow can hardware, and software requirements can be classified into:
Development Phase: This is when you train the mode. Training is usually done on your Desktop or laptop. Run Phase or Inference Phase: Once training is done Tensorflow can be run on many different platforms. You can run it on

- Desktop running Windows, macOS or Linux
- Cloud as a web service
- Mobile devices like iOS and Android

You can train it on multiple machines then you can run it on a different machine, once you have the trained model. The model can be trained and used on GPUs as well as CPUs. GPUs were initially designed for video games.

In late 2010, Stanford researchers found that GPU was also very good at matrix operations and algebra so that it makes them very fast for doing these kinds of calculations. Deep learning relies on a lot of matrix multiplication. TensorFlow is very fast at computing the matrix multiplication because it is written in C++. Although it is implemented in C++, TensorFlow can be accessed and controlled by other languages mainly, Python.

Finally, a significant feature of Tensor Flow is the Tensor Board. The Tensor Board enables to monitor graphically and visually what TensorFlow is doing.

List of Prominent Algorithms supported by TensorFlow

- Linear regression: tf. estimator .Linear Regressor
- Classification :tf. Estimator .Linear Classifier
- Deep learning classification: tf. estimator. DNN Classifier
- Booster tree regression: tf.estimator.BoostedTreesRegressor
- Boosted tree classification: tf.estimator.BoostedTreesClassifier

Python Overview

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, mac OS and Linux.

Why use Navigator?

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages and update them, all inside Navigator.

WHAT APPLICATIONS CAN I ACCESS USING NAVIGATOR?

The following applications are available by default in Navigator:

- Jupyter Lab
- Jupyter Notebook
- QT Console
- Spyder
- VS Code
- Glue viz
- Orange 3 App
- Rodeo
- RStudio

Advanced anaconda users can also build your own Navigator applications.

How can I run code with Navigator?

The simplest way is with Spyder. From the Navigator Home tab, click Spyder, and write and execute your code.You can also use Jupyter Notebooks the same way. Jupyter Notebooks are an increasingly popular system that combine your code, descriptive text, output, images and interactive interfaces into a single notebook file that is edited, viewed and used in a web browser.

5.Implementation

Data mining is a process to extract knowledge from existing data. It is used as a tool in banking and finance, in general, to discover useful information from the operational and historical data to enable better decision-making. It is an interdisciplinary field, the confluence of Statistics, Database technology, Information science, Machine learning, and Visualization. It involves steps that include data selection, data integration, data transformation, data mining, pattern evaluation, knowledge presentation. Banks use data mining in various application areas like marketing, fraud detection, risk management, money laundering detection and investment banking.

Data Exploration by grouping them on basis of class :

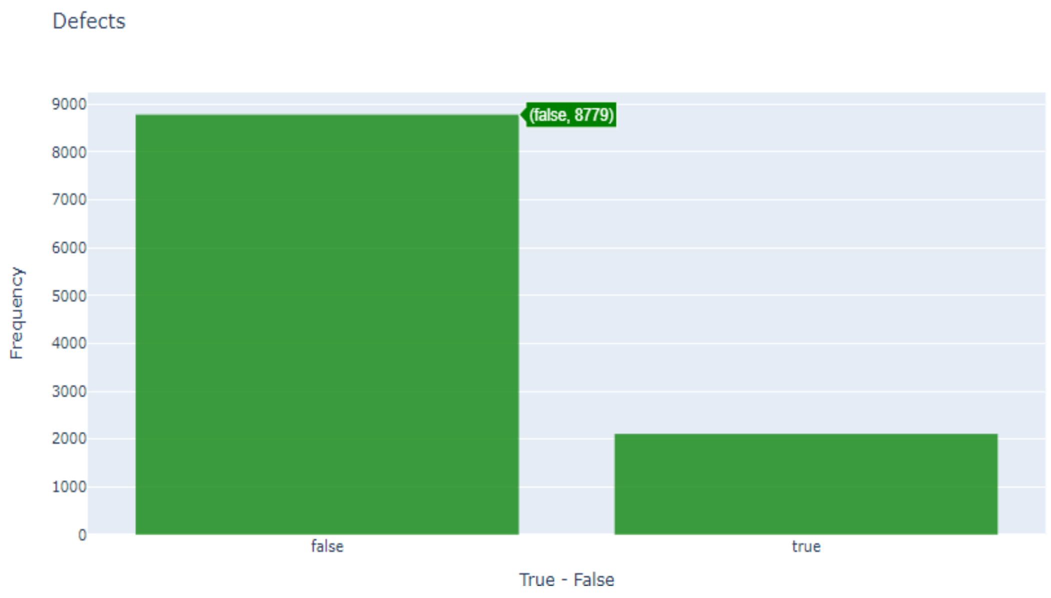


Figure 6: Class Label ratio

Feature Extraction using Correlation

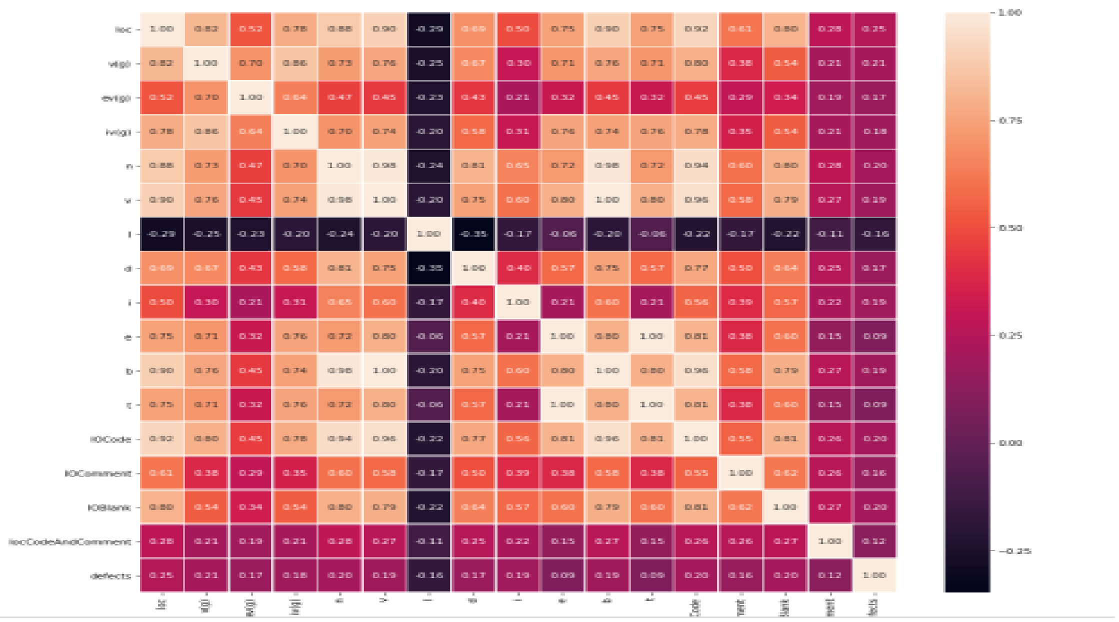
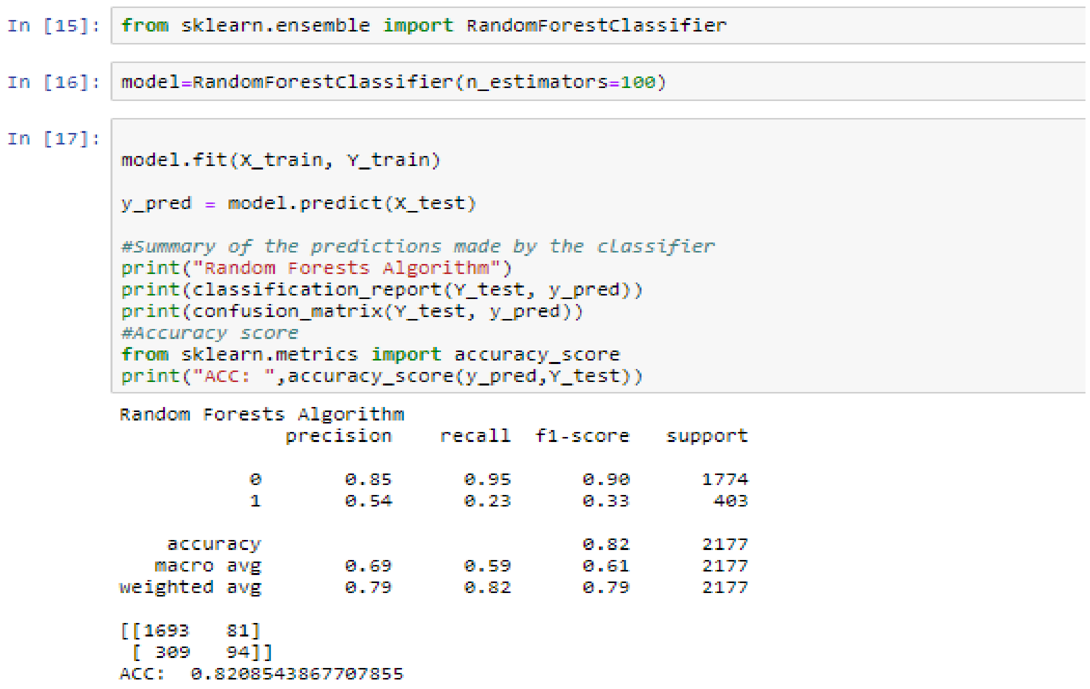
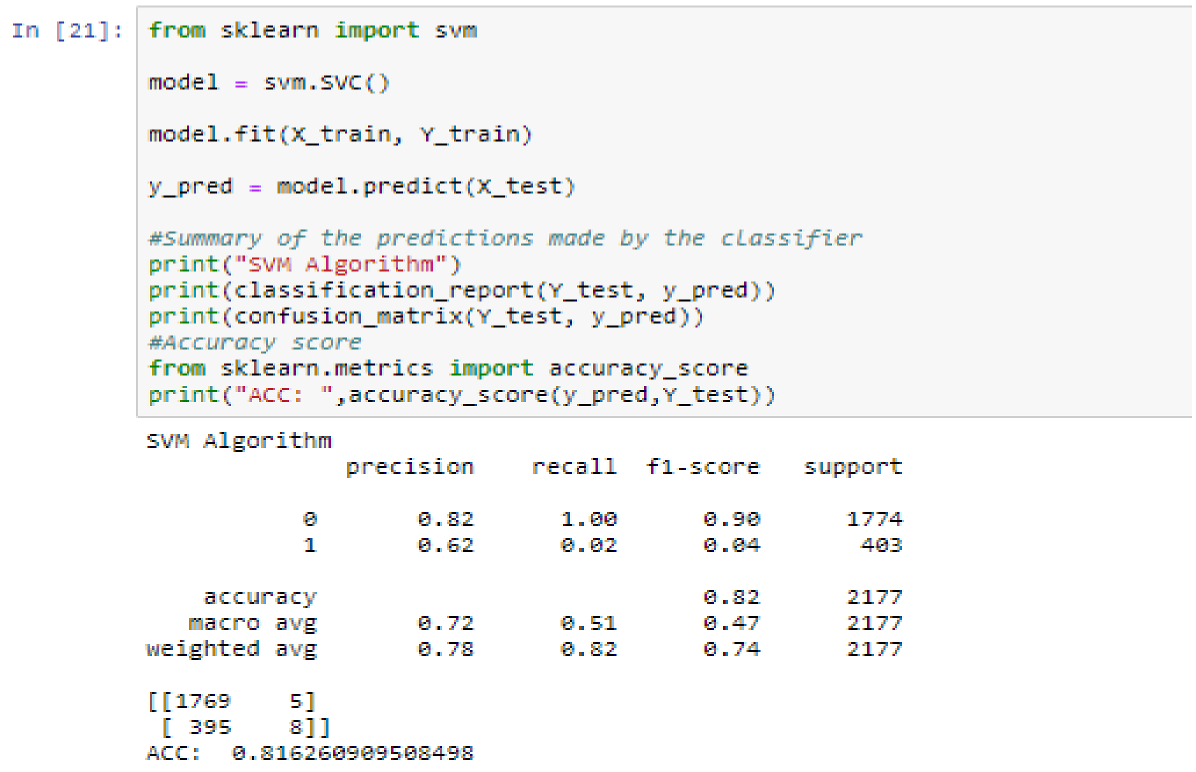


Figure 7: Correlation Graph

Accuracy based on Random Forest


```
In [15]: from sklearn.ensemble import RandomForestClassifier

In [16]: model=RandomForestClassifier(n_estimators=100)

In [17]: 
         model.fit(X_train, Y_train)

         y_pred = model.predict(X_test)

         #Summary of the predictions made by the classifier
         print("Random Forests Algorithm")
         print(classification_report(Y_test, y_pred))
         print(confusion_matrix(Y_test, y_pred))
         #Accuracy score
         from sklearn.metrics import accuracy_score
         print("ACC: ",accuracy_score(y_pred,Y_test))

         Random Forests Algorithm
                       precision    recall  f1-score   support

                    0       0.85      0.95      0.90      1774
                    1       0.54      0.23      0.33       403

             accuracy                           0.82      2177
            macro avg       0.69      0.59      0.61      2177
         weighted avg       0.79      0.82      0.79      2177

         [[1693   81]
          [ 309   94]]
         ACC:  0.8208543867707855
```

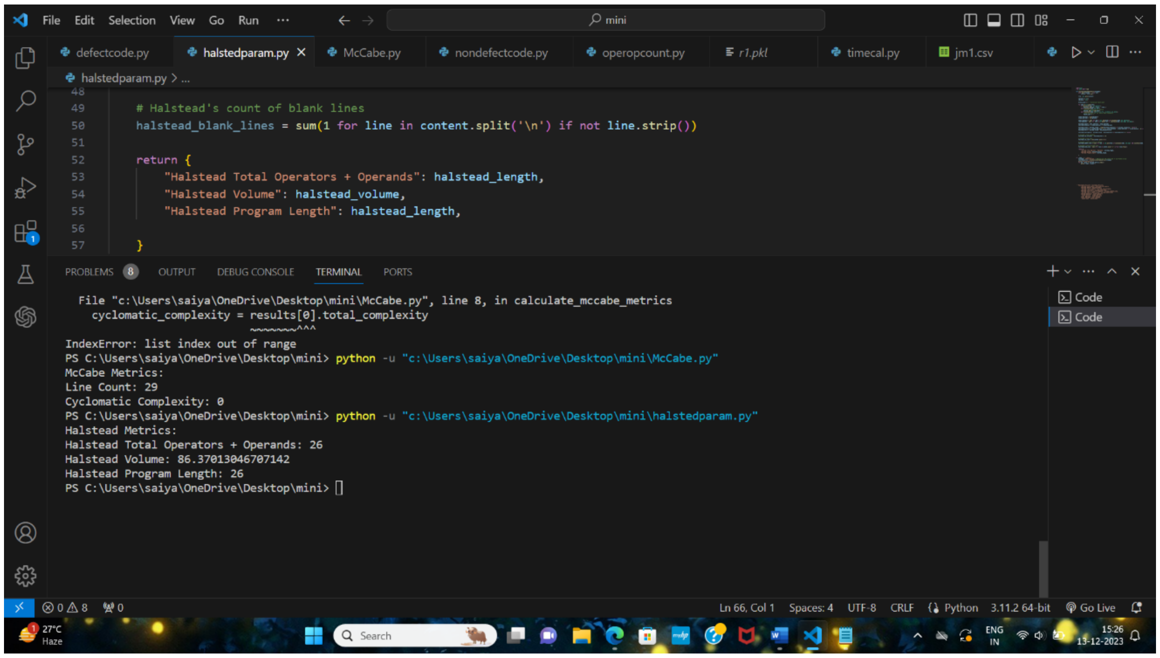
Accuracy based on Support Vector Machine



```
In [21]: from sklearn import svm

         model = svm.SVC()

         model.fit(X_train, Y_train)

         y_pred = model.predict(X_test)

         #Summary of the predictions made by the classifier
         print("SVM Algorithm")
         print(classification_report(Y_test, y_pred))
         print(confusion_matrix(Y_test, y_pred))
         #Accuracy score
         from sklearn.metrics import accuracy_score
         print("ACC: ",accuracy_score(y_pred,Y_test))
```

```
SVM Algorithm
              precision    recall  f1-score   support

           0       0.82      1.00      0.90      1774
           1       0.62      0.02      0.04       403

    accuracy                           0.82      2177
   macro avg       0.72      0.51      0.47      2177
weighted avg       0.78      0.82      0.74      2177

[[1769    5]
 [ 395    8]]
ACC:  0.816260909508498
```

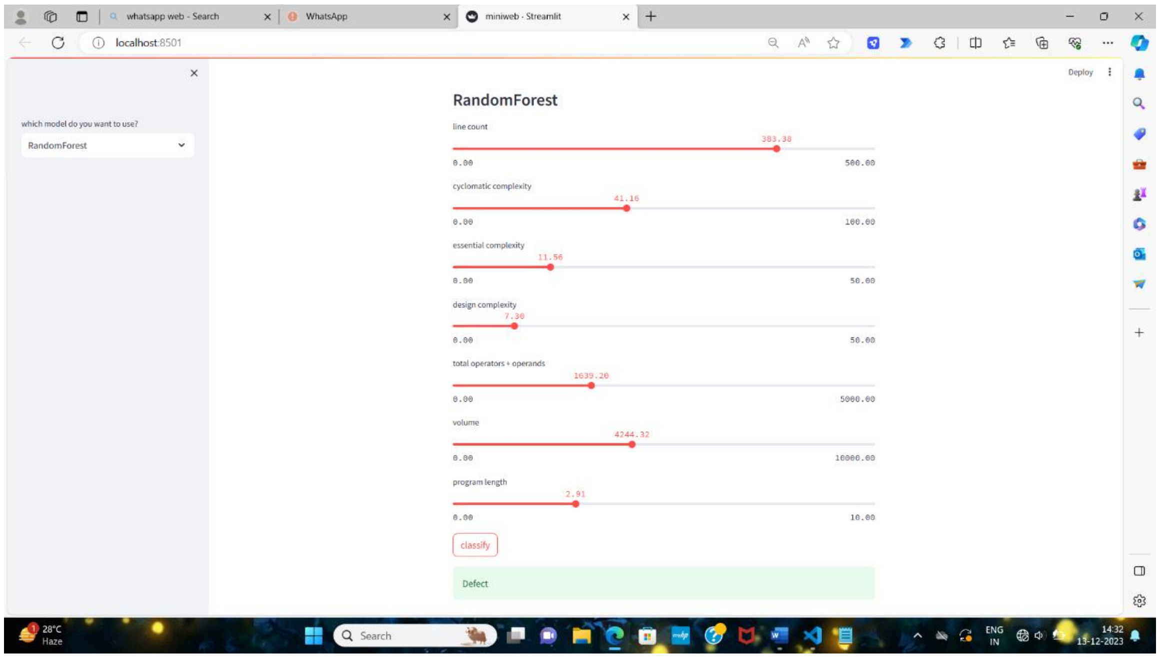
Calculating Parameters



```
    # Halstead's count of blank lines
    halstead_blank_lines = sum(1 for line in content.split('\n') if not line.strip())

    return {
        "Halstead Total Operators + Operands": halstead_length,
        "Halstead Volume": halstead_volume,
        "Halstead Program Length": halstead_length,

    }
```

```
  File "c:\Users\saiya\OneDrive\Desktop\mini\McCabe.py", line 8, in calculate_mccabe_metrics
    cyclomatic_complexity = results[0].total_complexity
                            ~~~~~~~~^^^
IndexError: list index out of range
PS C:\Users\saiya\OneDrive\Desktop\mini> python -u "c:\Users\saiya\OneDrive\Desktop\mini\McCabe.py"
McCabe Metrics:
Line Count: 29
Cyclomatic Complexity: 0
PS C:\Users\saiya\OneDrive\Desktop\mini> python -u "c:\Users\saiya\OneDrive\Desktop\mini\halstedparam.py"
Halstead Metrics:
Halstead Total Operators + Operands: 26
Halstead Volume: 86.37013046707142
Halstead Program Length: 26
PS C:\Users\saiya\OneDrive\Desktop\mini> 
```

Output



which model do you want to use?

RandomForest

RandomForest

line count: 383.38 (0.00 – 500.00)

cyclomatic complexity: 41.16 (0.00 – 100.00)

essential complexity: 11.56 (0.00 – 50.00)

design complexity: 7.30 (0.00 – 50.00)

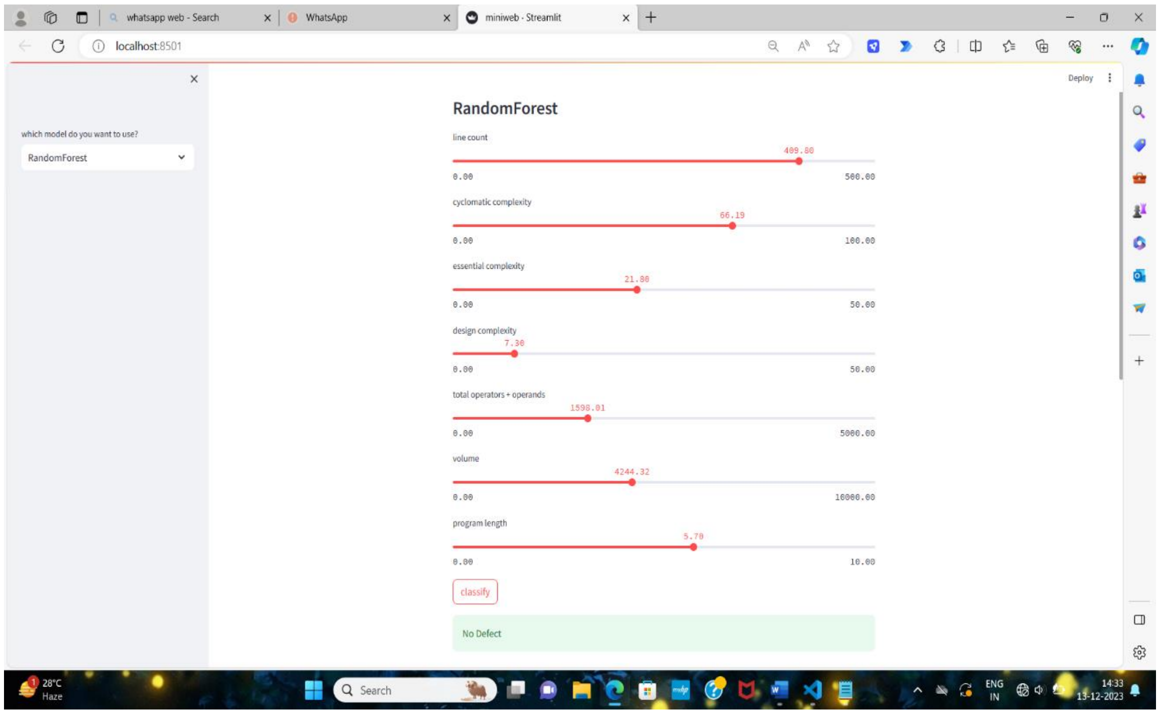
total operators + operands: 1639.20 (0.00 – 5000.00)

volume: 4244.32 (0.00 – 10000.00)

program length: 2.91 (0.00 – 10.00)

classify

Defect



which model do you want to use?

RandomForest

RandomForest

line count: 409.80 (0.00 – 500.00)

cyclomatic complexity: 66.19 (0.00 – 100.00)

essential complexity: 21.80 (0.00 – 50.00)

design complexity: 7.30 (0.00 – 50.00)

total operators + operands: 1598.01 (0.00 – 5000.00)

volume: 4244.32 (0.00 – 10000.00)

program length: 5.70 (0.00 – 10.00)

classify

No Defect

6.Testing and Results

TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks and implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

Software Testing can also be stated as the process of validating and verifying that a software program/application/product:

- Meets the business and technical requirements that guided its design and Development.
- Works as expected and can be implemented with the same characteristics.

TESTING METHODS

Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional testing:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

Unit Testing:

Features to be Tested:

- Format Validation:

  Ensure entries adhere to the correct format.
- Duplicate Entry Restriction:

  Verify that no duplicate entries are allowed.
- Link Functionality:

  Confirm all links direct users to the correct page.

Test Results:

All unit test cases passed successfully with no defects encountered.

Integration Testing:

Features to be Tested:

- Component Interaction:

  Check that integrated components interact without errors.

Test Results:

All integration test cases passed successfully with no defects encountered.

Functional Testing:

Features to be Tested:

- Field Entry Functionality:

  Confirm proper functioning of all field entries.

- Page Activation:

  Ensure pages are activated from identified links.

- Response Timeliness:

  Verify that entry screens, messages, and responses are not delayed.

Test Results:

All functional test cases passed successfully with no defects encountered

7.Conclusion and Future Scope

This research primarily seeks to use information-mining techniques to predict software-defects. Moreover, this domain is now a significant research field whereby numerous strategies have been explored to somehow enhance the efficiency of detecting software defects or predicting bugs. Throughout this study, by designing a new hybrid model using classification, dealt with the issue of classification accuracy for massive datasets.

These findings can be more improved with the use of several datasets. An increase in the number of datasets can enhance the findings. It is also possible to compare further techniques.

Software fault prediction and proneness has long been considered as a critical issue for the tech industry and software professionals. In the traditional techniques, it requires previous experience of faults or a faulty module while detecting the softwarefaults inside an pplication. An automated software fault recovery models enable the software to significantly predict and recover software faults using machine learning techniques. Such ability of the feature makes the software to run more effectively and reduce the faults, time and cost. In this paper, we proposed a software defect predictive development models using machine learning techniques that can enablethe software to continue its projected task. Moreover, we used different prominent evaluation benchmarks to evaluate the model's performance such as tenfold cross-validation techniques, precision, recall, specificity, f 1 measure, and accuracy.This study reports a significant classification performance of 98-100% using SVM onthree defect datasets in terms of f1 measure. However, software practitioners and researchers can attain independent understanding from this study while selecting automated tasks for their intended application.

8.Bibiliography

1. Jayanthi, R. and Florence, L., 2019. Software defect prediction techniques using metrics based on neural network classifiers. Cluster Computing, 22(1), pp.77-88.
2. Felix, E.A. and Lee, S.P., 2017. Integrated approach to software defect prediction. IEEE Access, 5, pp.21524-21547.
3. Wang, T., Zhang, Z., Jing, X., Zhang, L.: Multiple kernel ensemble learning for software defect prediction. Autom. Softw. Eng. 23, 569–590 (2015).
4. Xu, Z., Xuan, J., Liu, J., Cui, X.: MICHAC: defect prediction via feature selection based on maximal information coefficient with hierarchical agglomerative clustering. In: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Suita,
5. pp. 370–381 (2016).
6. Ryu, D., Baik, J.: Effective multi-objective naïve Bayes learning for cross-project defect prediction. Appl. Soft Comput. 49, 1062 (2016).
7. Shan C., Chen B., Hu C., Xue J., Li N.: Software defect prediction model based on LLE and SVM. In: Proceedings of the Communications Security Conference (CSC '14), pp. 1–5 (2014).
8. Yang, Z.R.: A novel radial basis function neural network for discriminant analysis. IEEE Trans. Neural Netw. 17(3), 604–612(2006).
9. K. Han, J.-H. Cao, S.-H. Chen, and W.-W. Liu, "A software reliability prediction method based on software development process," in Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE), 2013 International Conference on. IEEE, 2013, pp. 280–283.
10. S. Parthipan, S. Senthil Velan, and C. Babu, "Design level metrics to measure the complexity across versions of ao software," in Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on. IEEE, 2014, pp. 1708–1714.
11. A. Panichella, R. Oliveto, and A. De Lucia, "Cross-project defect prediction models: L'union fait la force," in Software Maintenance,
12. Bautista, A.M., Feliu, T.S.: Defect prediction in software repositories with artificial

neural networks. In: Mejia, J., Munoz,M., Rocha,Á., Calvo-Manzano, J. (eds.) Trends and Applications in Software Engineering.Advances in Intelligent Systems and Computing, vol.405. Springer, Cham (2016).

13. H. Lu, B. Cukic, and M. Culp, “Software defect prediction using semisupervised learning with dimension reduction,” in Automated Software Engineering (ASE), 2012 Proceedings of the 27th IEEE/ACM International Conference on. IEEE, 2012, pp. 314–317.

9.Appendix

```
//testing accuracies for SVM and RandomForest models
import numpy as np
import pandas as pd
import warnings
from sklearn import model_selection
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
warnings.filterwarnings("ignore")
# Load data
data1 = pd.read_csv('jm1.csv')
data = data1.sample(frac=0.2, random_state=42)
# Extract features and target variable
x = data.iloc[:, :-1].values
y = data['defects']  # Select the 'defects' column
# Check the number of features
# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(x, y, test_size=0.2,
random_state=0)
# Create and train the classifier
classifier = SVC(kernel="linear")
classifier.fit(X_train, Y_train)
# Make predictions
y_pred = classifier.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(Y_test, y_pred)
print("Accuracy:", accuracy)
from sklearn import model_selection
data = pd.read_csv('jm1.csv')
#print(data.shape[1])
```

```
x=data.iloc[:, :-1].values
y=data.iloc[:,data.columns =='defects']
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(x, y, test_size = 0.2,
random_state =0)
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_estimators=100)
model.fit(X_train, Y_train)
y_pred = (model.predict(X_test) > 0.5).astype(int).flatten()
accuracy = accuracy_score(Y_test, y_pred)
print("Accuracy:", accuracy)

//connecting the code with web using streamlit
import numpy as np
import pandas as pd
import warnings
import pickle
from sklearn.svm import SVC
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, MaxPooling2D
warnings.filterwarnings("ignore")
from sklearn import model_selection
data = pd.read_csv('jm1.csv')
#print(data.shape[1])
x=data.iloc[:, :-1].values
no=x.shape[1]
print(no)
y=data.iloc[:,data.columns =='defects']
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(x, y, test_size = 0.2,
random_state =0)
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_estimators=100)
```

```
model.fit(X_train, Y_train)
print("done1")
data1 = data.sample(frac=0.2, random_state=42)
x1=data1.iloc[:, :-1].values
y1=data1.iloc[:,data.columns =='defects']
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(x1, y1, test_size = 0.2,
random_state =0)
classifier = SVC(kernel="linear")
classifier.fit(X_train, Y_train)
print("done2")
model5 = Sequential()
model5.add(Dense(32, activation='relu', input_shape=(x1.shape[1],)))
model5.add(Flatten())
model5.add(Dense(128, activation='relu'))
model5.add(Dense(1, activation='sigmoid'))
model5.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
X_train, X_test, y_train, y_test = model_selection.train_test_split(x1, y1, test_size=0.2,
random_state=0)

# Train the model
model5.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2, verbose=1)
print("done3")
pickle.dump(model,open('r1.pkl','wb'))
pickle.dump(classifier,open('svm1.pkl','wb'))
pickle.dump(model5,open('cnn1.pkl','wb'))

//creating a web page
import streamlit as st
import pickle
import numpy as np
rm=pickle.load(open('r1.pkl','rb'))
```

```
svmm=pickle.load(open('svm1.pkl','rb'))
cnnm=pickle.load(open('cnn1.pkl','rb'))
def classify(num):
    if num<0.5:
        return "Defect"
    else: return "No Defect"
 def main():
 st.title("MINI PROJECT")
 html_temp= """
              <div style="background-color:#025246; padding:10px">
              <h2 style="color:white;text-align:center;">Software Defect Detection</h2>
              </div>
               """
st.markdown(html_temp, unsafe_allow_html=True)
activities=["RandomForest", "SVM", "CNN"]
option=st.sidebar.selectbox("which model do you want to use?",activities)
st.subheader(option)
st.spinner("hello")
loc=st.slider("line count",0.0,500.0)
cc=st.slider("cyclomatic complexity",0.0,100.0)
ec=st.slider("essential complexity",0.0,50.0)
dc=st.slider("design complexity",0.0,50.0)
n=st.slider("total operators + operands",0.0,5000.0)
v=st.slider("volume",0.0,10000.0)
pl=st.slider("program length",0.0,10.0)
inputs=[[loc,cc,ec,dc,n,v,pl]]
if st.button('classify'):
    if option=="RandomForest":
        st.success(classify((rm.predict(inputs))))
    elif option=="SVM":
        st.success(classify((svmm.predict(inputs))))
    else:
        st.success(classify((cnnm.predict(inputs))))
if __name__=='__main__':
    main()

//calculating the attribute values
//McCabe values
from radon.complexity import cc_visit

def calculate_mccabe_metrics(filename):
    # Calculate the cyclomatic complexity using radon
    results = cc_visit(filename)

    # Extracting McCabe's default metrics
    if results:
        cyclomatic_complexity = results[0].total_complexity
    else:
        cyclomatic_complexity = 0

    # Calculate line count
    with open(filename, 'r', encoding='utf-8') as file:
        line_count = len(file.readlines())

    return {
        "Line Count": line_count,
        "Cyclomatic Complexity": cyclomatic_complexity,
    }
if __name__ == "__main__":
    filename = "defectcode.py"  # Replace with the actual name of your Python script
    mccabe_metrics = calculate_mccabe_metrics(filename)
    print("McCabe Metrics:")
    for key, value in mccabe_metrics.items():
        print(f"{key}: {value}")
//hasledparameters
import ast
from math import log2
def calculate_halstead_metrics(filename):
    with open(filename, 'r') as file:
        content = file.read()

    tree = ast.parse(content)

    operators = set()
    operands = set()

    branch_count = 0  # Initialize branch count

    for node in ast.walk(tree):
        if isinstance(node, ast.operator):
            operators.add(node.__class__.__name__)
        elif isinstance(node, ast.Expr):
            # Consider expressions as operands
            operands.add(ast.dump(node))
        elif isinstance(node, (ast.If, ast.For, ast.While, ast.With)):
            # Count branches (If, For, While, With statements)
            branch_count += 1

    unique_operators = len(operators)
    unique_operands = len(operands)

    total_operators = sum(1 for node in ast.walk(tree) if isinstance(node, ast.operator))
    total_operands = sum(1 for node in ast.walk(tree) if isinstance(node, (ast.Expr,
ast.Name)))

halstead_length = total_operators + total_operands
halstead_vocabulary = unique_operators + unique_operands
halstead_volume = halstead_length * (log2(halstead_vocabulary)
if halstead_vocabulary > 0 else 0)
    halstead_difficulty = (unique_operators / 2) * (total_operands / unique_operands) if
unique_operands > 0 else 0
    halstead_effort = halstead_volume * halstead_difficulty

    halstead_intelligence = halstead_volume / halstead_effort if halstead_effort > 0 else 0
 # Halstead's time estimator
    halstead_time_estimator = halstead_effort / 18

    # Halstead's line count
    halstead_line_count = len(content.split('\n'))

    # Halstead's count of lines of comments
    halstead_comment_lines = sum(1 for node in ast.walk(tree) if isinstance(node,
ast.Expr) and isinstance(node.value, ast.Str))

    # Halstead's count of blank lines
    halstead_blank_lines = sum(1 for line in content.split('\n') if not line.strip())

    return {
        "Halstead Total Operators + Operands": halstead_length,
        "Halstead Volume": halstead_volume,
        "Halstead Program Length": halstead_length,

    }
```

```
if __name__ == "__main__":
    filename = "defectcode.py"  # Replace with the actual name of your Python script
    halstead_metrics = calculate_halstead_metrics(filename)
    print("Halstead Metrics:")
    for key, value in halstead_metrics.items():
        print(f"{key}: {value}")
```