**Software Fault Prediction Using Machine Learning**

**Approaches by Reduced Feature Set**

A PROJECT REPORT

By

**Tanvi Singh Chouhan RA1611020010009**

**Shubham Pratap RA1611020010045**

Under the guidance of

**Mr. C. Arun**

*“In partial fulfilment for the award of the degree”*

of

“BACHELOR OF TECHNOLOGY”

in

SOFTWARE ENGINEERING



**“FACULTY OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**Katankulathur, Kancheepuram, Tamil Nadu”**

MAY 2020

**BONAFIDE CERTIFICATE**

It is Certified that this project report **“****Software Fault Prediction Using Machine Learning Approaches by Reduced Feature Set”** is the bonafide work of “**TANVI SINGH CHOUHAN AND SHUBHAM PRATAP”** who carried out the project work under my supervision.

Dr. C. Lakshmi Mr. C. Arun

**Professor & Head Asst. Professor (O. G.)**

**Software Engineering Software Engineering**

INTERNAL EXAMINER EXTERNAL EXAMINER

**TABLE OF CONTENTS**

[**LIST OF FIGURES** 5](#_Toc40478506)

[**LIST OF TABLES** 5](#_Toc40478507)

[**ACKNOWLEDGEMENT** 6](#_Toc40478508)

[**CHAPTER 1** 7](#_Toc40478509)

[**INTRODUCTION** 7](#_Toc40478510)

[**CHAPTER 2** 8](#_Toc40478511)

[**PROJECT OVERVIEW** 8](#_Toc40478512)

[**2.1 LITERATURE SURVEY** 8](#_Toc40478513)

[**2.2 PROBLEM DESCRIPTION** 11](#_Toc40478514)

[**2.3 REQUIREMENTS GATHERING** 11](#_Toc40478515)

[**2.3.1 Brainstorming** 11](#_Toc40478516)

[**2.3.2 Questionnaire** 12](#_Toc40478517)

[**2.3.3 Interview** 13](#_Toc40478518)

[**2.3.4 Inference** 14](#_Toc40478519)

[**2.4** **REQUIREMENTS ANALYSIS** 14](#_Toc40478520)

[**2.4.1 FUNCTIONAL REQUIREMENTS** 14](#_Toc40478521)

[**2.4.2 NON-FUNCTIONAL REQUIREMENTS** 15](#_Toc40478522)

[**2.5 DATA SOURCE** 15](#_Toc40478523)

[**2.6 COST ESTIMATION** 15](#_Toc40478524)

[**2.7 PROJECT SCHEDULE** 18](#_Toc40478525)

[**2.8 RISK ANALYSIS** 19](#_Toc40478526)

[**2.9 Software Requirement Specification (SRS)** 19](#_Toc40478527)

[**2.9.1 INTRODUCTION** 19](#_Toc40478528)

[**2.4.1 FUNCTIONAL REQUIREMENTS** 20](#_Toc40478529)

[**2.4.2 NON-FUNCTIONAL REQUIREMENTS** 20](#_Toc40478530)

[**2.9.4 INTERFACE REQUIREMENT** 21](#_Toc40478531)

[**CHAPTER 3** 21](#_Toc40478532)

[**ARCHITECTURE & DESIGN** 21](#_Toc40478533)

[**3.1 SYSTEM ARCHITECTURE** 21](#_Toc40478534)

[**3.3 DATA FLOW DIAGRAM** 28](#_Toc40478535)

[**3.4 USE CASE DIAGRAM** 31](#_Toc40478536)

[**3.5 SEQUENCE DIAGRAM** 32](#_Toc40478537)

[**3.6 CLASS DIAGRAM** 33](#_Toc40478538)

[**CHAPTER 4** 34](#_Toc40478539)

[**IMPLEMENTATION** 34](#_Toc40478540)

[**4.1 DATABASE DESIGN** 34](#_Toc40478541)

[**4.1.1 ER DIAGRAM** 34](#_Toc40478542)

[**4.2 User Interface** 35](#_Toc40478543)

[**4.3 Middleware** 35](#_Toc40478544)

[**CHAPTER 5** 44](#_Toc40478545)

[**VERIFICATION AND VALIDATION** 44](#_Toc40478546)

[**5.4 COST ANALYSIS** 44](#_Toc40478547)

[**5.5 DEFECT ANALYSIS** 46](#_Toc40478548)

[**5.6 MC CALL’S QUALITY FACTORS** 46](#_Toc40478549)

[**5.6.1 Product Operation factors** 46](#_Toc40478550)

[**5.6.2 Product Revision Factors** 48](#_Toc40478551)

[**5.6.3 Product Transition Factors** 48](#_Toc40478552)

[**CHAPTER 6** 49](#_Toc40478553)

[**EXPERIMENT RESULTS AND ANALYSIS** 49](#_Toc40478554)

[**6.1 RESULTS** 49](#_Toc40478555)

[**6.2 RESULT ANALYSIS** 53](#_Toc40478556)

[**6.3 CONCLUSION AND FUTURE WORK** 57](#_Toc40478557)

[**REFRENCES** 58](#_Toc40478558)

[**PLAGIARISM REPORT** 59](#_Toc40478558)

**LIST OF FIGURES**

[Figure 1 : System Architecture 21](Draft%20final%20report%20(1).docx)

[Figure 2 : Architecture Diagram 22](#_3.1_SYSTEM_ARCHITECTURE)

[Figure 3 : Pre-Processing 2](Draft%20final%20report%20(1).docx)4

[Figure 4 : Data Flow Diagram 30](Draft%20final%20report%20(1).docx)

[Figure 5 : Data Flow Diagram 0](file:///C:\Users\Tanvi%20Chouhan\Downloads\MajorFinalReport%20(Mehak%20&amp;%20Shivam).docx#_Toc40288154) 31

[Figure 6: Data Flow Diagram 1](file:///C:\Users\Tanvi%20Chouhan\Downloads\MajorFinalReport%20(Mehak%20&amp;%20Shivam).docx#_Toc40288155) 31

[Figure 7 : Data Flow Diagram 2](file:///C:\Users\Tanvi%20Chouhan\Downloads\MajorFinalReport%20(Mehak%20&amp;%20Shivam).docx#_Toc40288156) 32

[Figure 8 : Use Case Diagram](file:///C:\Users\Tanvi%20Chouhan\Downloads\MajorFinalReport%20(Mehak%20&amp;%20Shivam).docx#_Toc40288157) 33

[Figure 9 : Sequence Diagram](file:///C:\Users\Tanvi%20Chouhan\Downloads\MajorFinalReport%20(Mehak%20&amp;%20Shivam).docx#_Toc40288158) 33

[Figure 10 : Class Diagram](file:///C:\Users\Tanvi%20Chouhan\Downloads\MajorFinalReport%20(Mehak%20&amp;%20Shivam).docx#_Toc40288159) 35

[Figure 11 : ER Diagram](file:///C:\Users\Tanvi%20Chouhan\Downloads\MajorFinalReport%20(Mehak%20&amp;%20Shivam).docx#_Toc40288160) 36

[Figure 12 : Accuracy level of Different Datasets using Naive Bayes](file:///C:\Users\Tanvi%20Chouhan\Downloads\MajorFinalReport%20(Mehak%20&amp;%20Shivam).docx#_Toc40288161) 53

[Figure 13 : Confusion Matrix Result using Naive Bayes 54](Draft%20final%20report%20(1).docx)

[Figure 14 : Accuracy level of Different Datasets using SVM…………… ……………………………………………….54](Draft%20final%20report%20(1).docx)

[Figure 15 : Confusion Matrix Result using SVM………………………………………………………………………………..55 Figure 16 : Comparing Accuracy Performance Between Naïve Bayes and SVM……………………………….55 Figure 17 : Comparing Accuracy Performance Between Naïve Bayes and SVM……………………………….56 Figure 18 : ROC Curve For Gaussian Naïve Bayes…………………………………………………………………………….58 Figure 19 : ROC Curve For SVM (Support Vector Machine)……………………………………………………………. 59](Draft%20final%20report%20(1).docx)

**LIST OF TABLES**

[Table 1: Basic COCOMO Co-efficients 1](file:///C:\Users\Tanvi%20Chouhan\Downloads\MajorFinalReport%20(Mehak%20&amp;%20Shivam).docx#_Toc40279460)6

[Table 2: Project Schedule 1](file:///C:\Users\Tanvi%20Chouhan\Downloads\MajorFinalReport%20(Mehak%20&amp;%20Shivam).docx#_Toc40279461)8

[Table 3: Accuracy of Software dataset](file:///C:\Users\Tanvi%20Chouhan\Downloads\MajorFinalReport%20(Mehak%20&amp;%20Shivam).docx#_Toc40279462) 28

[Table 4: Coefficient for Intermediate COCOMO](file:///C:\Users\Tanvi%20Chouhan\Downloads\MajorFinalReport%20(Mehak%20&amp;%20Shivam).docx#_Toc40279463) 48

[Table 5: CONFUSION MATRIX for Naïve Bayes](file:///C:\Users\Tanvi%20Chouhan\Downloads\MajorFinalReport%20(Mehak%20&amp;%20Shivam).docx#_Toc40279464) 52

[Table 6: CONFUSION MATRIX for SVM](file:///C:\Users\Tanvi%20Chouhan\Downloads\MajorFinalReport%20(Mehak%20&amp;%20Shivam).docx#_Toc40279465) 53

# 

# 

# **ACKNOWLEDGEMENT**

We would like to take this opportunity to express our sincere gratitude to all those who helped us to complete this project successfully.

We are deeply indebted to our guide Mr. C. Arun, Department of Software Engineering for his excellent guidance and valuable comments. He taught us the method to successfully complete our project and present the project as clearly as possible. We would like to thank him for his support throughout.

We would also like to thank our department who gave us the opportunity to do this wonderful project on the topic Software Fault Prediction Using Machine Learning Approaches by Reduced Feature Set which helped us in gaining a lot of knowledge.

We are greatly thankful to Dr. C. Lakshmi, Head of Software Engineering Department for allowing us to choose our desired project and for her support throughout.

Finally, we would like to thank our parents and friends who directly and indirectly contributed to the successful completion of our project.

# **CHAPTER 1**

# **INTRODUCTION**

In order to have a software that is reliable enough, so that it can be used without having to think of the defect or faults in it as the issue to be solved we need a system which can predict the fault in the software and give an output as to tell us if the software is reliable enough to be used in daily life.

There can be a thousand things that can go wrong by small fault in the software, these may include failure of a big multinational company, or failure of some transaction taking place, or failure of a car software causing someone’s life.

We need to recognize all the factors that cause faults or make the software to fail. And so here is machine learning which helps us through its algorithms to predict faults in a software prior to any sort of failure. And in our project, we have used the approach to reduce the set of formulas using the selection algorithms also. Here, we have used the NASA and UCI data repository so as to test and train the models.

In our approach we have used naïve Bayes and SVM (Support vector machine) algorithm for analyzing the data and hence processing it. We have used NASA data repository in order for the model to learn from the data and hence make the system most accurate.

The main objective of this project is to develop & implement a theorem that characterizes the features of the software defect prediction revolution, and proposes a working model, from the data mining algorithms perspective using Naive Bayes Classification and SVM algorithm.

A good forecasting helps us to ensure software quality is to top notch. Pre-bug optimization uses software features such as the number of lines of code, to predict a file or commit are all the part of the future features or not. The feature selection techniques reduce the number of elements that we are working with in the model so created and selects the most important element, whereas the feature reduction minimizes the number of features by developing new, integrated features from the original ones.

# **CHAPTER 2**

# **PROJECT OVERVIEW**

## **2.1 LITERATURE SURVEY**

1. “Software Metrics for Fault Prediction Using Machine Learning Approaches”, “IEEE 2017”, “Syaeful Karim, Harco Leslie Hendric Spits Warnars, Benfano Soewito”

Content: “In this paper, software metric is proved as one efficient source to provide fault predictive model. In despite of every type of metric can be used to estimate fault proneness module, class level metric shows better prediction performance compare to method level metric.”

1. “Software Defects Prediction based on ANN and Fuzzy logic using Software Metrics”, “IEEE 2015”, “T. Ravi Kumar, Dr. T. Srinivasa Rao”

Content: “In this paper, a fuzzy logic-based model proposed for predicting software defects at each step of SDLC, this rule is applied for the metrics data”

1. “Empirical Studies of a Two-Stage Data Preprocessing Approach for Software Fault Prediction, IEEE 2011”, “Wangshu Liu, Shulong Liu, Qing Gu, Member, IEEE, Jiaqiang Chen, Xiang Chen, Member, IEEE, and Daoxu Chen, Member, IEEE”

Content: “In this paper, we use two-stage data preprocessing approach, which incorporates both feature selection and instance reduction, to improve the quality of software datasets used by classification models for software fault prediction.”

1. “Applying machine learning to predict software fault proneness using change metrics, static code metrics, and a combination of them”, “IEEE 2014”, “Yasser Ali Alshehri, Katerina Goseva-Popstojanova, Dale G. Dzielski and Thomas Devine”

Content: “This study shows us to compare different machine learning algorithm for software fault prediction. It uses different dataset from eclipse and in this paper, they found that using the reduced feature set of metrics produces slightly higher accuracy than using all other metrics.”

1. “Machine learning based software fault prediction utilizing source code metrics”, “IEEE

2018”, “Guru Prasad Bhandari, Ratneshwar Gupta”

Content: “In this proposed system, they studied on different machine learning algorithm like Decision tree, Naïve Bayes, Support Vector Machine… to check the feature selection process in order to predict the software fault prediction using different metrics. They analyzed the performance of the different machine learning approaches.”

1. “An Empirical Study on Software Defect Prediction with a Simplified Metric Set”, “Article in Information and Software Technology · February 2014”, “Peng He, Bing Li, Xiao Liu, Jun Chen, Yutao Ma”

Content: “In this paper, they aimed at predicting how a predictor based on simplified metrics set in build and used for both CPDP and WPDP. It built with a different metrics set which work well and is very useful in case of limited resources.”

1. “Dynamic Selection of Classifiers in Bug Prediction: An Adaptive Method”, “IEEE 2017”, “Dario Di Nucci, Fabio Palomba, Rocco Oliveto, and Andrea De Lucia”

Content: “In this paper, they proposed ASCI, which is an approach able to dynamically recommend the classifier to use and detect the bug prediction based on structural characteristics.”

1. “The Practice of predictive analytics in software”, “IEEE September 2015.”, “Gopala Krishna Palam”

Content: “In this paper, effective mechanisms have been used for software defect prediction by mining the data containing historical records. Here, we used Naïve Bayes, Support Vector Machine (SVM) and classifiers of software defect. In this study, we also present comparative study of different classifiers to measure the performance based on accuracy rate.”

1. “Support-vector networks", Machine Learning, vol. 20, no. 2, pp. 273-297, 1995.”

Content: “In this paper The support-vector network is a new learning machine for two-group classification problems. The machine conceptually implements the following idea: input vectors are non-linearly mapped to a very high-dimension feature space. In this feature space a linear decision surface is constructed. Special properties of the decision surface ensure high generalization ability of the learning machine. The idea behind the support-vector network was previously implemented for the restricted case where the training data can be separated without errors. We here extend this result to non-separable training data.”

1. "A Tutorial on Support Vector Machines for Pattern Recognition", “Data Mining and Knowledge Discovery, Springer, vol. 2, no. 2, pp. 121-167, 1998.”, “Christopher J.C. Burges “

Content: “In this study they describe linear Support Vector Machines (SVMs) for separable and non-separable data, working through a non-trivial example in detail. We describe a mechanical analogy, and discuss when SVM solutions are unique and when they are global. We describe how support vector training can be practically implemented, and discuss in detail the kernel mapping technique which is used to construct SVM solutions which are nonlinear in the data.”

## **2.2 PROBLEM DESCRIPTION**

* In the world where the reliability on software has increased a lot, from the car to flights to the space moving satellites all work using different software.
* Since the dependency of the human race inn software is so much, a minute mistake can cost us lives. And the software accuracy should not be questioned and should be a thing that should be relied up on.
* So, to predict if the software is accurate, or is reliable enough to be used is a big problem that we face every day.
* In order to have a system that can predict the defects in a software is what our world needs, this can save a lot of money and lives by having a prediction of the reliability of the software before sending it to live working phase.
* It is hence very important to have software that can predict faults in an existing software.

## **2.3 REQUIREMENTS GATHERING**

### **2.3.1 Brainstorming**

Brainstorming is a requirement gathering technique that involves members of an organization or group sitting together and trying to generate ideas. These ideas are then communicated to each other whereupon it is decided whether the idea is to be followed or not.

**Target Problem:**

What should be the technique used to predict the fault in the software?

**Solutions:**

We should use the naïve Bayes and the SVM algorithm. Also we should also reduce the already existing data using the feature reduction after the preprocessing is done

**Advantages:**

Promotes Creativity.

Everyone gets to air out their ideas

Helps in bolstering team work and productivity

**Disadvantages:**

Can lead to arguments or disagreements

Depends significantly on the team members ability to express their idea

### **2.3.2 Questionnaire**

A questionnaire is a requirement gathering technique which involves handing out a document with a list of questions which have to be answered. The questionnaire is an informal technique in which the answer can be in any format. The answer should be clear and to the point.

**Questions:**

What data repository are we going to use?

* NASA data repository

What is the best method already present to predict fault?

* Using naïve Bayes

Can you recommend any new and cost-effective method?

* By reducing the data set metrics used before selection is done.

Where could such a system be implemented or applied?

* Personal Laptops and Computers

**Advantages:**

It is cost efficient.

Respondents can answer completely anonymously.

It can be used to gather a large amount of information in one go.

It is practical and cost efficient.

**Disadvantages:**

Answers may be dishonest.

Questions may not be understood properly by the respondents.

### **2.3.3 Interview**

An interview is another requirement gathering technique which involves a panel or an individual asking another individual a series of questions so as to gather his/her thoughts or views and also to gather possible ideas. It is the job of the interviewer to make the interviewee feel comfortable. The interviewer must have already prepared a set of question to ask beforehand. The interviewer’s job is to understand clearly what the interviewee is talking about.

**Questions:**

In your opinion, what is the best method to implement the prediction of fault in ML?  
- Using naïve bayes

How do you make sure that there is no noise in data?

- We after taking the data have to preprocess the data.

Why do you recommend to use reduced set?

- We recommend to use reduced set as the reduced set metric produces higher accuracy than using all the change metric.

**Advantages:**

They are very personal in nature.

Gives individuals to properly communicate their ideas.

**Disadvantages:**

Time consuming

Only limited number of people will be interviewed.

### **2.3.4 Inference**

Brainstorming helped us in getting creative solution like using of inbuilt camera instead of additional hardware which helps in reducing the overall cost.

Questionnaires helped us in getting the idea how can gesture detection be made more efficient.

Interview allowed us to ask more in-depth questions so as to get a better idea.

## **2.4** **REQUIREMENTS ANALYSIS**

### **2.4.1 FUNCTIONAL REQUIREMENTS**

* Predict the fault in the software.
* Should display the results of different data sets using graphical means(bar graph).

### **2.4.2 NON-FUNCTIONAL REQUIREMENTS**

* System continues to work normal even in the presence of malicious attacks.
* System operates properly when it is requested for use
* System is durable i.e. it meets user's needs for a relatively long time.
* Should use a feature reduction algorithm.
* System is reliable as all the data stored is secure and multiple users can use all the functionalities at the same time without any discrepancies. This project can be run on all the operating systems without any failure.
* This project is efficient as it doesn't require any external resource, so all users can easily use it without having any extra hardware.

## **2.5 DATA SOURCE**

For this project we have used the datasets obtained from NASA data repository that is publicly available MDP (Metrics Data program) and also the Promise data repository from eclipse datasets.

## **2.6 COST ESTIMATION**

We are going to use COCOMO model for cost estimation:

“COCOMO stands for Constructive Cost Model and was developed by Barry W. Boehm. It is a procedural model to estimate software costs. It’s parameters are derived from fitting a regression formula using data from already implemented or historical projects.”

**Why COCOMO?**

“Basic COCOMO is good and preferred for quick estimation of the software costs. It does not comply for differences in hardware used/installed, personnel and individual quality, experience, use of tools and techniques (modern/traditional), and more.”

Project type:

Organic

Why organic?

* Small team
* Small project size

Effort (E) = a\* (KLOC)^b (person-months)

Development Time (D)= c \* (Effort)^d (months)

No. of People required (P)= E/D (people)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Software project** | **a** | **b** | **c** | **d** |
| Organic | 2.4 | 1.05 | 2.5 | 0.38 |
| Semi-detached | 3.0 | 1.12 | 2.5 | 0.35 |
| Embedded | 3.6 | 1.20 | 2.5 | 0.32 |

Table 1: Basic COCOMO Coefficients

“COCOMO means Constructive Cost Model, is a method for evaluating or estimating the cost of software development. The constant values of a, b, c and d for the basic model are for the different categories of the system.”

**Cost Estimation – Effort**

This project contains roughly ~ 0.9 KLOC (900 LOC)

By putting these values in the formula, the following would arrive:

Effort(E) = a \* (KLOC)^b

=2.4\*(0.9) ^1.05

=2.14865098401 ~ 2-man months

**Cost Estimation – Development Time**

Here E is 2.1487

Development Time (D) = c \* (E)^d

=2.5 \* (2.1487) ^0.38

=3.34323424391~ 3 months

**Cost Estimation – Average Staff Size**

Here E is 2.1487

D = 3.34323

Average Staff Size (SS) = E/D

= 2.1487 / 3.34323

= 0.6427018183 ~ 1 person

**Cost Estimation – Productivity**

Here E is 2.1487

KLOC = 0.9,

Productivity(P) = KLOC/E

= 0.9 / 2.1487

=0.41885791408

Note:

* Final cost is dependent on the salary of the individuals involved

## **2.7 PROJECT SCHEDULE**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | TOPIC | FROM DATE | TO DATE |
| 1 | LITERATURE SURVEY | 15/11/19 | 25/11/19 |
| 2 | REQ GATHERING | 25/11/19 | 30/11/19 |
| 3 | COST ESTIMATE | 30/11/19 | 05/12/19 |
| 4 | RISK ANALYSIS | 05/12/19 | 08/12/19 |
| 5 | ARCHITECTURE DESIGN | 08/12/19 | 14/12/19 |
| 6 | U I DESIGN | 14/12/19 | 18/12/19 |
| 7 | USE CASE | 18/12/19 | 22/12/19 |
| 8 | SEQUENCE AND CLASS DIAGRAM | 22/12/19 | 26/12/19 |
| 9 | STATE DIAGRAM | 26/12/19 | 30/12/19 |
| 10 | DATA BASE DESIGN | 30/12/19 | 05/01/20 |
| 11 | USER INTERFACE | 05/01/20 | 20/01/20 |
| 12 | MIDDLE WARE | 20/01/20 | 25/01/20 |
| 13 | TESTING | 25/01/20 | 10/02/20 |
| 14 | ANALYSIS | 10/02/20 | 01/03/20 |
| 15 | Mc Calls QUALITY FACTOR | 01/03/20 | 08/03/20 |
| 16 | RESULT | 08/03/20 | 15/03/20 |
| 17 | RESULT ANALYSIS | 15/03/20 | 19/03/20 |
| 18 | FUTURE ENHANCEMENT | 19/03/20 | 25/03/20 |

Table 2: Project Schedule

## **2.8 RISK ANALYSIS**

* Not having enough data
* Having biased data
* Homogeneous data
* Lack of model variability
* False error

## **2.9 Software Requirement Specification (SRS)**

### **2.9.1 INTRODUCTION**

In order to have a software that is reliable enough, so that it can be used without having to think of the defect or faults in it as the issue to be solved we need a system which can predict the fault in the software and give an output as to tell us if the software is reliable enough to be used in daily life.

There can be a thousand things that can go wrong by small fault in the software, these may include failure of a big multinational company, or failure of some transaction taking place, or failure of a car software causing someone’s life.

We need to recognize all the factors that cause faults or make the software to fail. And so here is machine learning which helps us through its algorithms to predict faults in a software prior to any sort of failure. And in our project,

have used the approach to reduce the set of formulas using the selection algorithms also. Here, we have used the NASA and UCI data repository so as to test and train the models.

In our approach we have used naïve Bayes and SVM (Support vector machine) algorithm for analyzing the data and hence processing it. We have used NASA data repository in order for the model to learn from the data and hence make the system most accurate.

The main objective of this project is to develop & implement a theorem that characterizes the features of the software defect prediction revolution, and proposes a working model, from the data mining algorithms perspective using Naive Bayes Classification and SVM algorithm.

A good forecasting helps us to ensure software quality is to top notch. Pre-bug optimization uses software features such as the number of lines of code, to predict a file or commit are all the part of the future features or not. The feature selection techniques reduce the number of elements that we are working with in the model so created and selects the most important element, whereas the feature reduction minimizes the number of features by developing new, integrated features from the original ones.

### **2.4.1 FUNCTIONAL REQUIREMENTS**

* Predict the fault in the software.
* Should display the results of different data sets using graphical means (bar graph).

### **2.4.2 NON-FUNCTIONAL REQUIREMENTS**

* System continues to work normal even in the presence of malicious attacks.
* System operates properly when it is requested for use.
* System is durable i.e. it meets user's needs for a relatively long time.
* Should use a feature reduction algorithm.
* System is reliable as all the data stored is secure and multiple users can use all the functionalities at the same time without any discrepancies. This project can be run on all the operating systems without any failure.
* This project is efficient as it doesn't require any external resource, so all users can easily use it without having any extra hardware.

### **2.9.4 INTERFACE REQUIREMENT**

Since the project does not have any other interface apart from python IDLE, So the constraint for the same is specified.

1. Hardware constraint:
   1. 1 GB RAM
   2. 80 GB Hard Disk
   3. Intel Processor
   4. LAN
2. Software constraint:

Your system should be compatible enough to run python codes and install its packages and should not lag.

# 

# **CHAPTER 3**

# **ARCHITECTURE & DESIGN**

## **3.1 SYSTEM ARCHITECTURE**

**Software DATA ANALYTICS**

Text DATA

Dimension Aggregation

Software Dataset

**PICKLE**

**Analysisof SVM classification**

**DISK**

**PICKLEE**

Data Cleaning

PICKLEEE

**Result**

Figure 1 : System Architecture

Figure 1 represents system architecture of our project. The system basically uses the different data sets from NASA data repository, the promise software engineering repository (public sets) all the data is collected. So, all the data from these datasets is collected which is the first step, further the data is pre-processed. The pre-processing or data cleaning the data is made assessable to be used, as the text data is further converted to the reduced metrices and these further are pickled that is they are given the sequence or series and then the data metrics is selected using the SVM algorithm or the naïve bayes classification. Further the model is trained and tested. Hence, the results are displayed using the bar graphs.

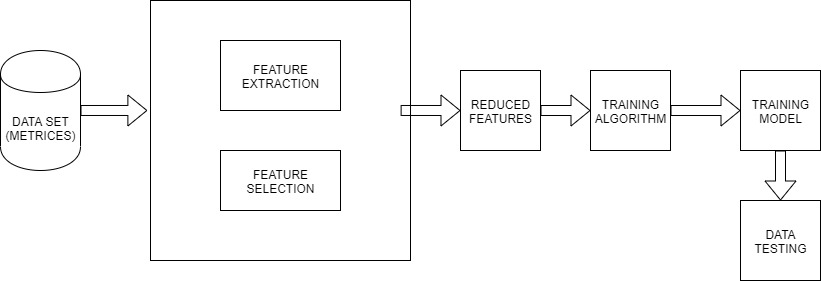


Figure 2 : Architecture Diagram

**3.1.1MODULES:**

* Data Collection
* Pre-Processing
* Feature extraction
* Classification
* Efficiency Calculation

**3.1.1.1 Data Collection**

* In connection-based communications, the data stream is a sequence of digital embedded symbols (data packets or data packets) used to transmit or receive information in transmission plans.
* We have taken our data from the NASA Promise data repository available for the public use.
* A lot of parameters are considered while predicting whether a software is buggy or not which include number of lines in the code, its complexity, the number of operators and operands used in the code and other factors. We have considered a set of 22 initial features to predict whether the software is buggy.
* For this comparison we have used 5 datasets

**3.1.1.2 Pre-Processing:**

* So, in our project we have done data pre-processing in order to avoid noise and enhance the accuracy of prediction.
* For data pre-processing, we studied our data closely and then processed the data and removed the null values. After removing the null values, we have labelled our data into two parts the observations and result part that is X and Y.
* Further we have converted the String value in Y part to 0 and 1 to make the data easier to work with.

Data packets

**Pre-processing**

**Feature Extraction**

Filter packets

Figure 3: Pre-Processing

**3.1.1.3Feature Extraction:**

* Feature extraction is the technique to extract the most important, or the feature which are independent of each other.
* Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation which converts a set of correlated variables to a set of uncorrelated variables. PCA is a most widely used tool in exploratory data analysis and in machine learning for predictive models.
* We have used Kernel Principal Component Analysis(non-linear) to reduce the dimensions from 21 to 6, this has hence improved the accuracy and made it easy for the big data to be handled.

**3.1.1.4 Classification:**

* The classification we have used Gaussian Naïve Bayse and Support Vector Machine to have a comparison.
* Bayse theorem is based on **conditional probability**. The conditional probability helps us calculating the probability that something will happen, *given that something else* has already happened
* A Gaussian Naive Bayes algorithm is a special type of NB algorithm. It’s specifically used when the features have continuous values. It’s also assumed that all the features are following a gaussian distribution i.e, normal distribution.
* A support vector machine is machine learning algorithm that analyses data for classification and regression analysis. SVM is a supervised learning method that looks at data and sorts it into one of two categories.
* So, using these two we have created the model. We divide the data into training and testing where we 80% of data we kept for training and rest 20% for testing.

**3.1.1.5 EFFICIENCY CALCULATION:**

* According to the predictions made by both the models we calculated the accuracy, precision, recall and f1 score.
* F1 scores are weighted average of Precision and Recall taking into consideration both False Positives and False Negatives.
* Finally, we got the confusion matrix for all the datasets individually and also the roc curves.

Test Results:

To evaluate classification, accuracy and AUC measures are combined. Four cases are considered to be the result of a classifier.

TP (TRUE POSITIVE): the number of samples correctly classified in that category.

TN (True Negative): the number of samples correctly rejected in that category.

FP (False Positive): the number of samples incorrectly rejected in that category.

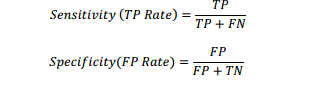
FN (False Negative): the number of samples that were incorrectly classified in that class.

“The adequacy level of the partition model is determined by the number of right and wrong segments in all the input separations. Accuracy can be calculated by using below. We have used medical data such as software and heart disk data. All of this information is derived from the UCI machine learning data [10, 12]. The purpose is to distinguish errors based on parameter selection and to compare all categories such as SVM, Naïve Bayes.”

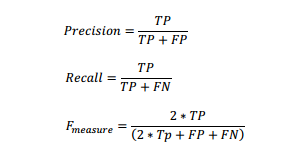
TABLE 3. ACCURACY OF SOFTWARE DATASET

|  |  |  |
| --- | --- | --- |
| **SL.NO** | **Classifiers** | **Accuracy %** |
| 1 | Support Vector Machine (SVM) | 89 |
| 2 | Naive Bayes | 86 |
|  |  |  |

According to Figure, we can clearly see that SVM gives highest accuracy 82% in case of software, Naïve Bayes gives the lowest accuracy 74% accuracy among all the classifiers in case of software defect. In case of software Support Vector Machine gives the highest accuracy measures that depict how well quite far between a case with positive and with negative class. Sensitivity is the ID of defect rate that ought to be expanded and Specificity is the false organized rate that is to be minimized for definite examination. Sensitivity and specificity can be effortlessly found by utilizing underneath formulae.



“A recognized confusion matrix (sometimes called contingency table) is acquired to estimate four measures. Confusion matrix is a matrix representation of the classification results. It contains data about real and predicted classifications done by a classification framework. The cell  
which signifies the quantity of samples orders as true while they were true (i.e., TP), and the cell that indicates the quantity of samples named false while they were quite (i.e., TN). The other two cells signify the quantity of samples misclassified. In particular, the cell signifying the quantity of samples named false while they really were true (i.e., FN), and the” cell meaning the quantity of samples named true while they really were false (i.e., FP) [20]. Once the confusion matrixes were built, the precision, recall, F-measure are effectively ascertained as:



## **3.3 DATA FLOW DIAGRAM**

Figure 4 : Data Flow Diagram

Batch sequential is a type of Data Flow Architecture and also a classical data processing model, in which a data transformation subsystem i.e. a particular module, can initiate or work on its process only after its previous module – the subsystem is completely through −

* The flow goes from one module to another as it carries a batch of data. Temporary/intermediate files helps in the communications between subsystems/interface which can be removed by the upcoming modules.

Software Dataset

Data set...

Result

Figure 5: Data Flow Diagram 0

Software dataset

Large\_vol\_of\_dataset

Check Result

Valid Not valid

DISK

Figure 6: Data Flow Diagram 1

Software Dataset

Data set files

Check by user

Search again

Result

Figure 7: Data Flow Diagram 2

## **3.4 USE CASE DIAGRAM**

**USER**

**Jupiter Tool**

**Software Data**

**Big Query Form**

**Data set collection**

**User select query**

**PICKLE...n()**

**DESK()**

**Result**

Figure 8: Use-Case Diagram

A use case diagram in UML is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of a functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

In our project, there two major actors, the main or prominent actor is the user and the other passive actor is the Jupiter notebook (Anaconda tool) that we use for the python libraries. So firstly, we get the data sets pre-processed using the tool, and apply all the algorithms further, the result in form of graphs is made using the matplotlib and NumPy of python 3.

## 

## **3.5 SEQUENCE DIAGRAM**

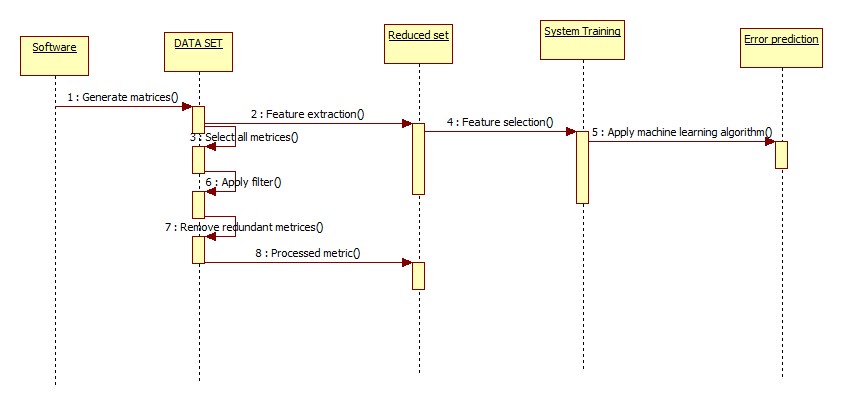


Figure 9: Sequence Diagram

A sequence diagram is one of the interaction diagrams because it explains the togetherness of a group of objects with the order they follow. Software developers and business professionals generally use sequence diagrams to understand all the requirements of a new system or for the documentation purpose of an existing process. The lifelines, or the processes and objects that live/operate simultaneously are the main focus. Also, the messages exchanged between them to perform a function before the end of a lifeline gets the specific focus.

In our Project, from the repository the software is narrowed down to datasets and hence on the data set the csv are further feature extracted and then the system is trained and tested which generates the error prediction model.

## **3.6 CLASS DIAGRAM**

**Software Data**

+Profile

+input query

+Authenticate()

+query process()

**TOOL ENGINE**

+URL

+Dataset

+Big query()

+PICKLE()

**MapReduce**

+index

+Data set

+PICKLE()

+DISK()

**Algorithms**

+SVM

+Classifier

+Dispaly()

Figure10: Class Diagram

One of the most useful types of UML diagrams in software engineering is the Class Diagram. They map out the clear structure of a particular system by modelling and mentioning its classes, attributes, operations or functions, and relationships between the objects.

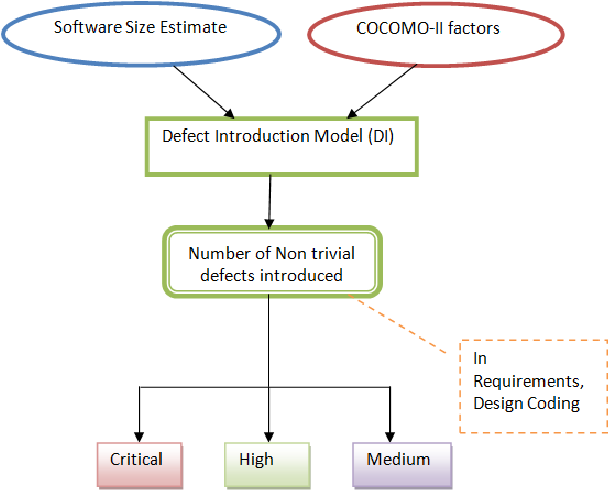
In our project, there are classes such as the Software data, tool engine, MapReduce, and algorithms. All the algorithms are used to display the final result whereas the software data class has methods like query process which will help the tool engine to process the query and hence serial and de-serial the queries. Further this pickled data is made into a disk by the MapReduce class and the final sequenced data is assessed and the result is displayed using the algorithms of ML.

# **CHAPTER 4**

# **IMPLEMENTATION**

## **4.1 DATABASE DESIGN**

### **4.1.1 ER DIAGRAM**



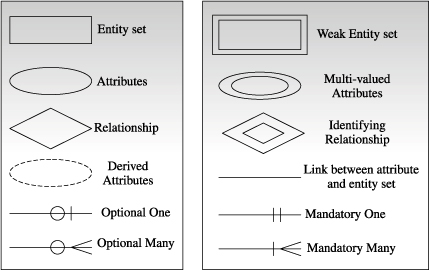


Figure 11: ER Diagram

## **4.2 User Interface**

Since in our project we having to give the result, whether the software is reliable. So, the user interface would just be an output or result graphs. The user would not have an interface as such. We will get the results in form of graphical means.

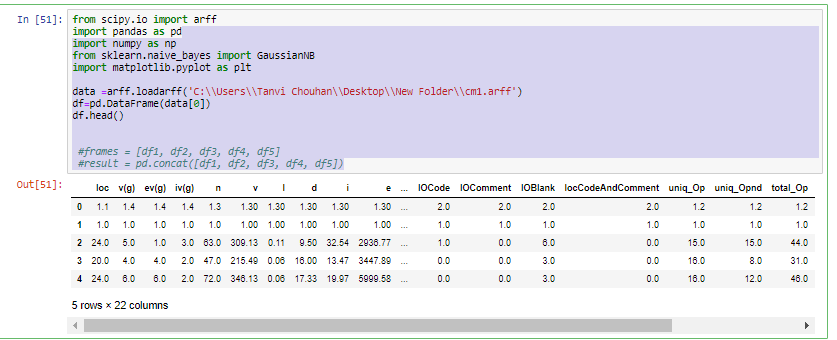
## **4.3 Middleware**

Services to software applications beyond those available from the operating system are provided by a computer software called Middleware. It acts as a connector between an OS or database and applications. The middleware used for our project are as follows:

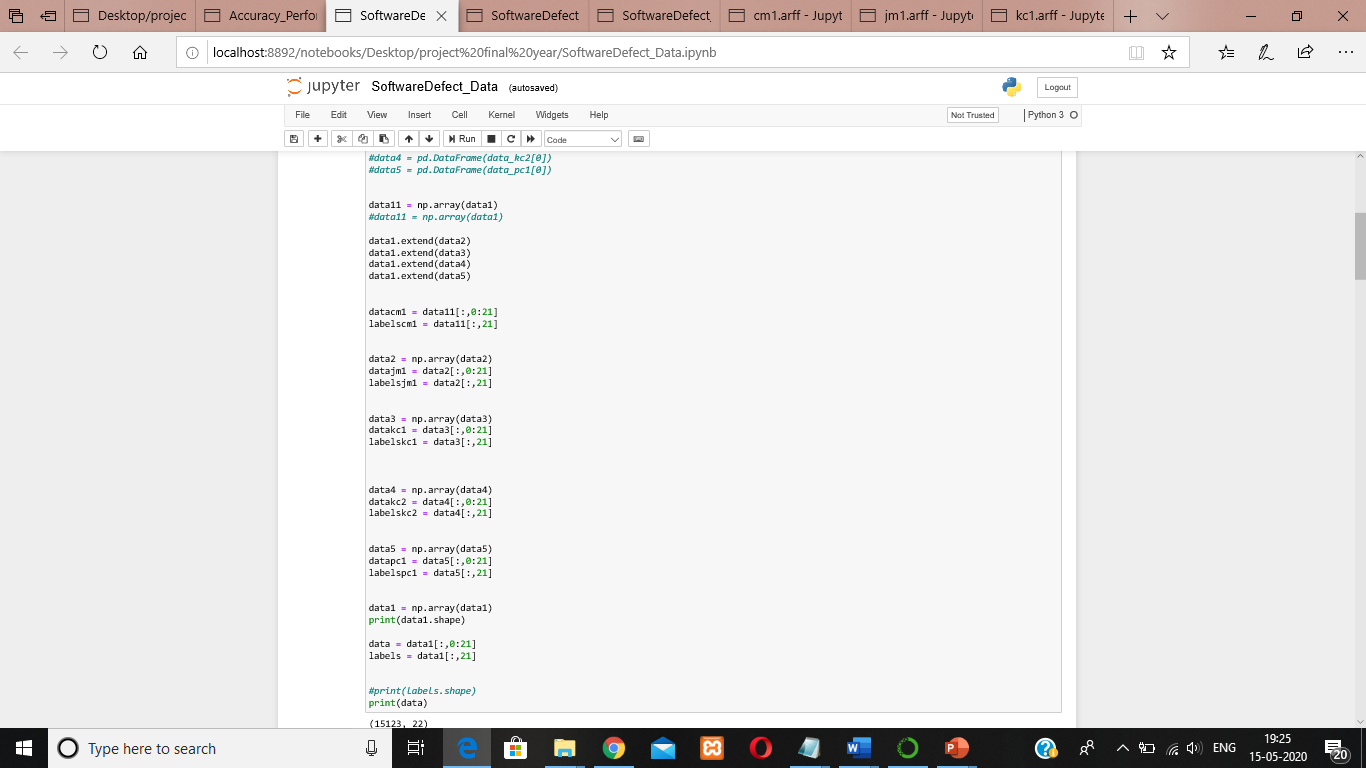
* python 3.7
* Jupiter Notebook
* Anaconda Navigator
* Star Unified Modelling Language (UML)
* pip version 20.0.2

**CODING:**

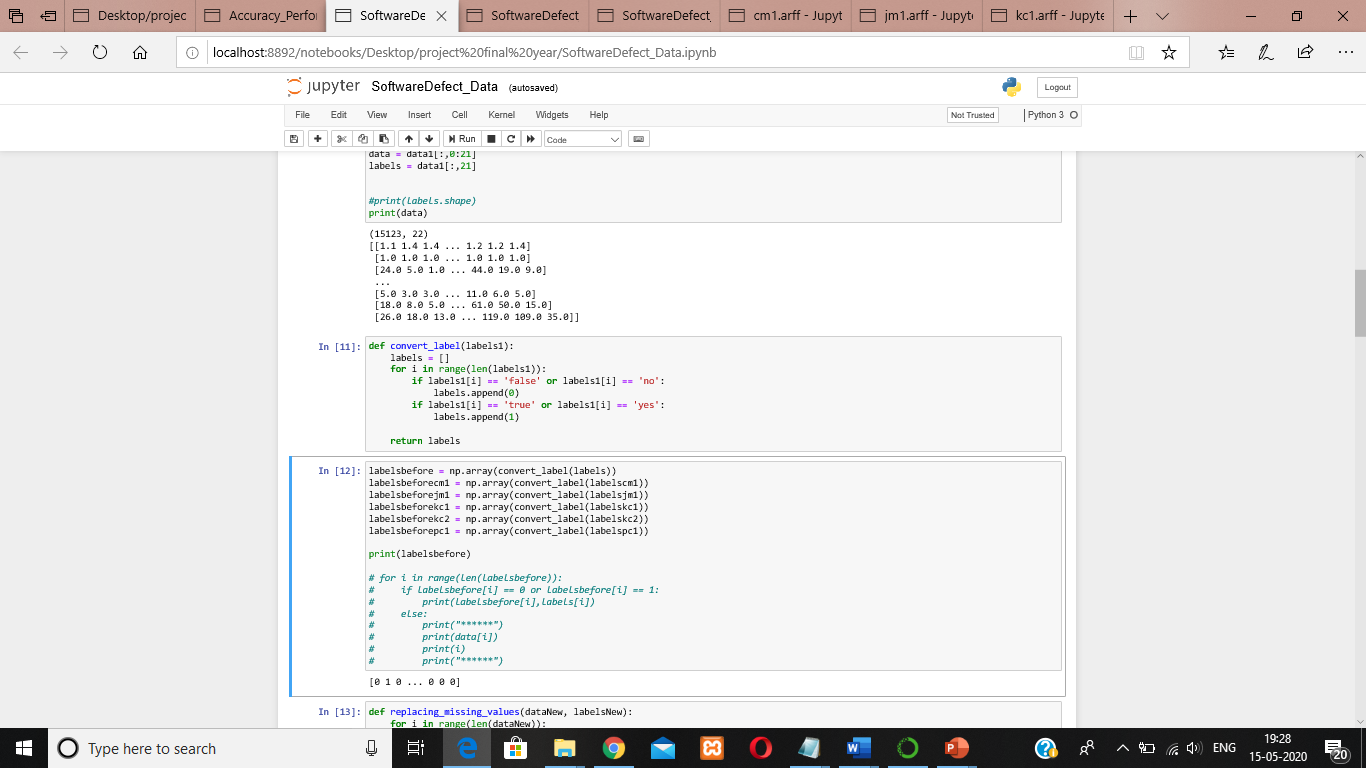
* Data Collection
* Pre-Processing
* Feature extraction
* Classification
* Efficiency Calculation
* Open data repository from NASA Promise data repository.
* **PRE-PROCESSING**: defining all the datasets

****

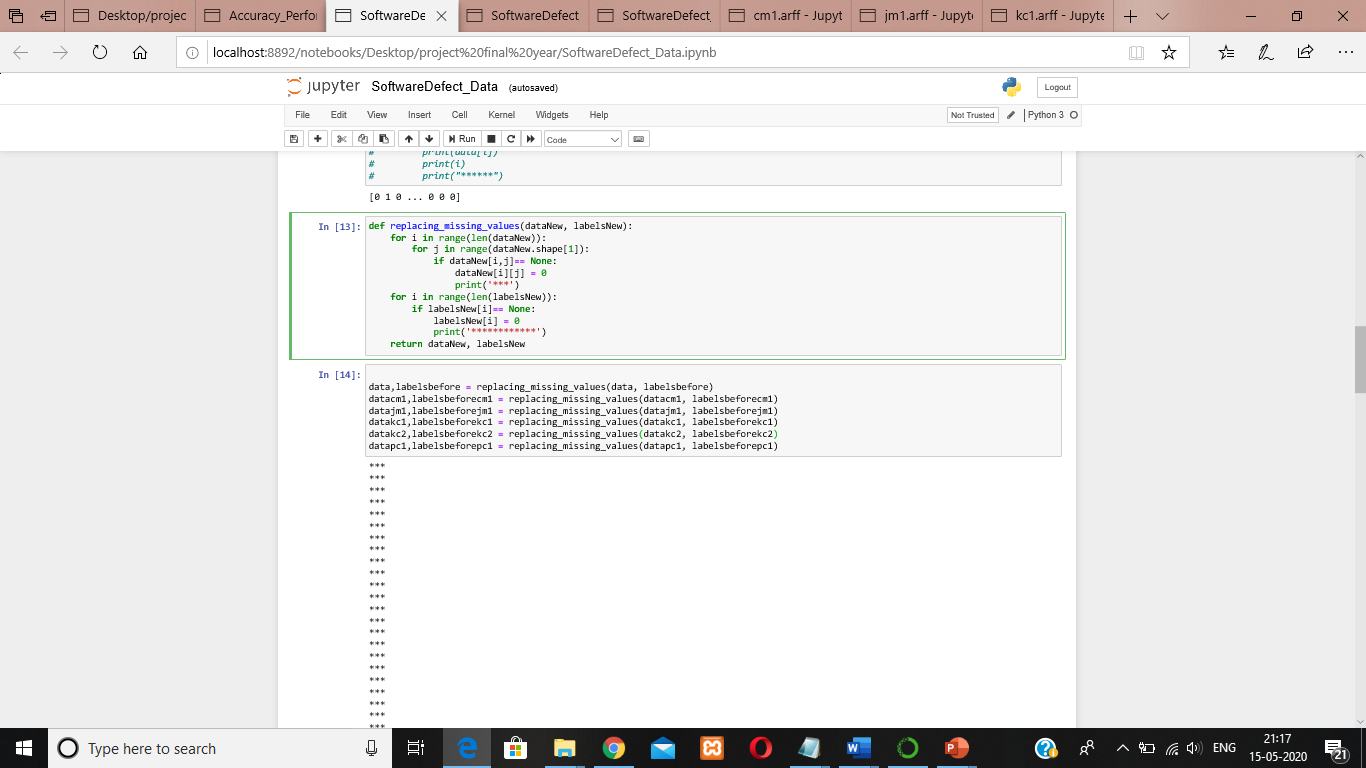
* Labelling and reducing the data



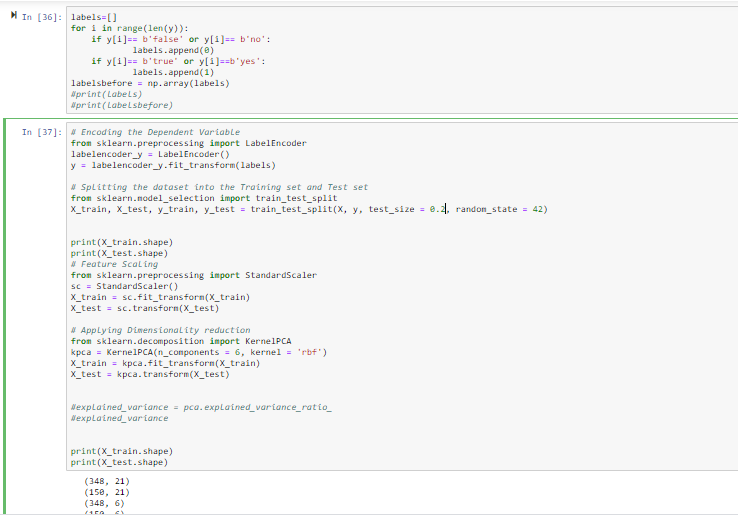
* Labelling false/true or no/yes as zero and one.
* Storing values in array.

****

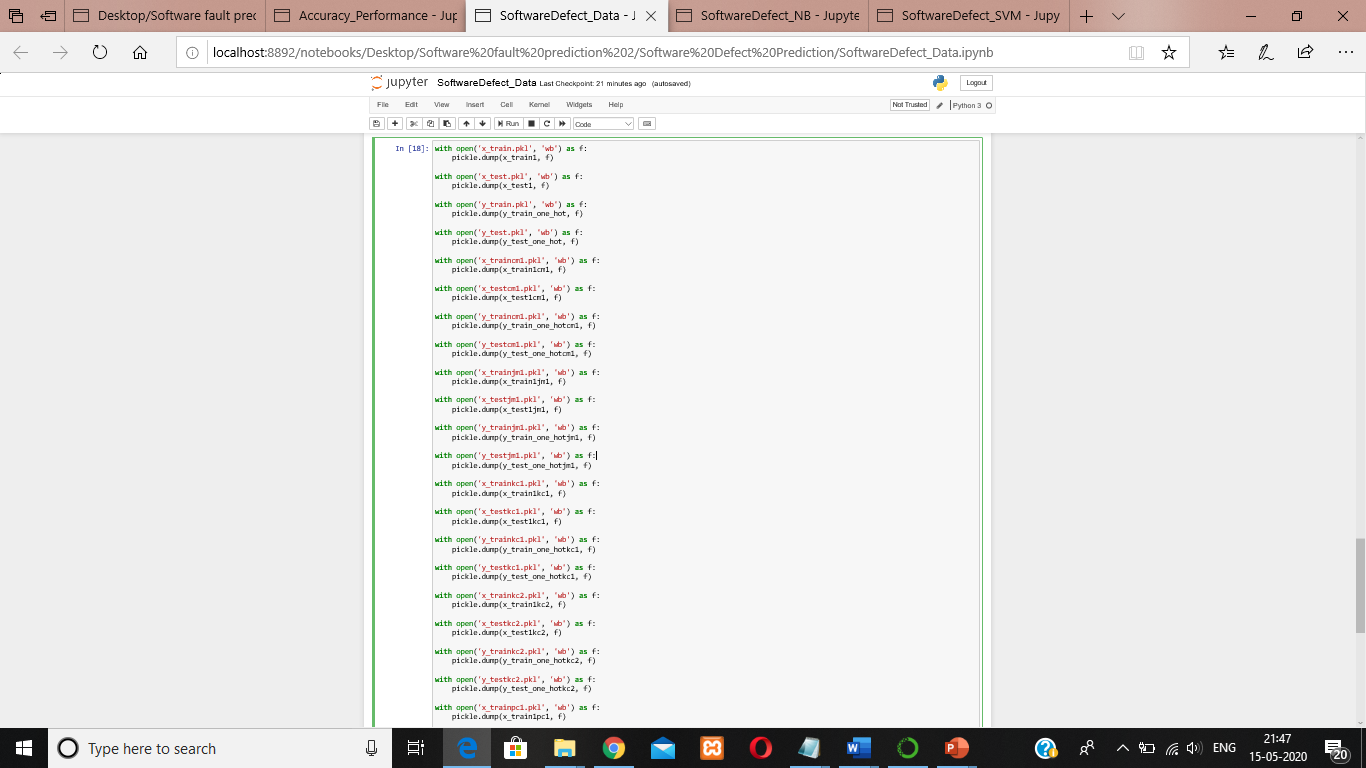
* Removing the noise. By removing null values from the data.

****

* Assigning 20% of data to testing and 80% to training. Shuffling and splitting of data.

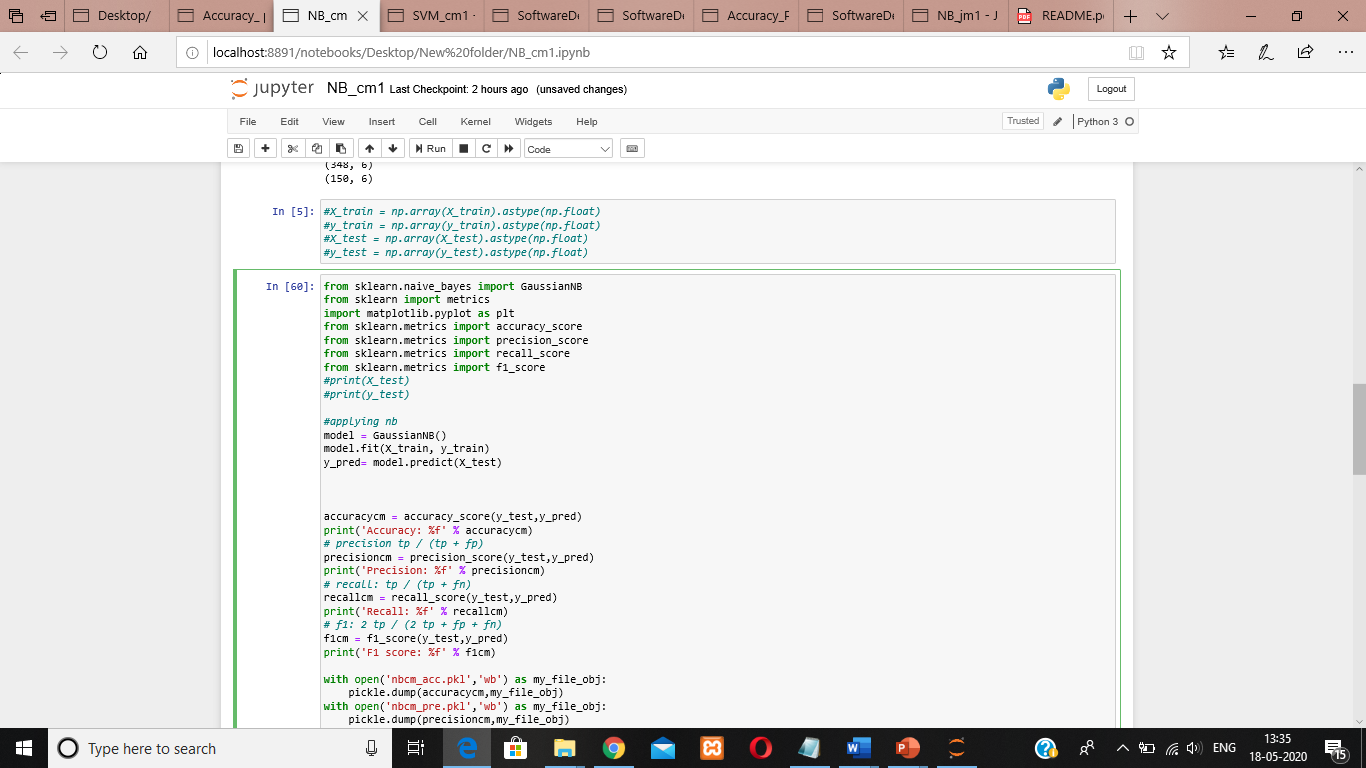


* Using pickle to serialize datasets

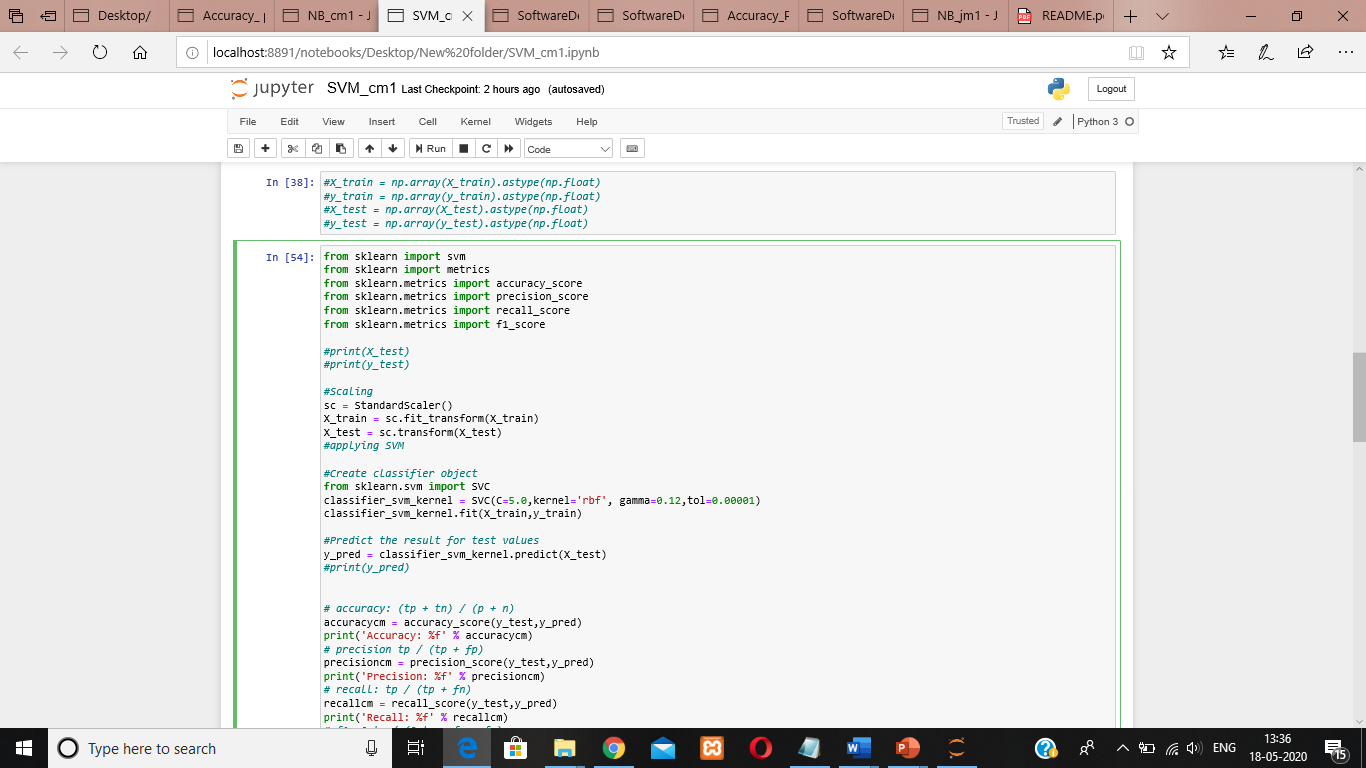
****

**Feature extraction:**

* Using the feature extraction algorithm Naïve Bayes to make the model.



* Using the feature extraction (SVM) to make the model

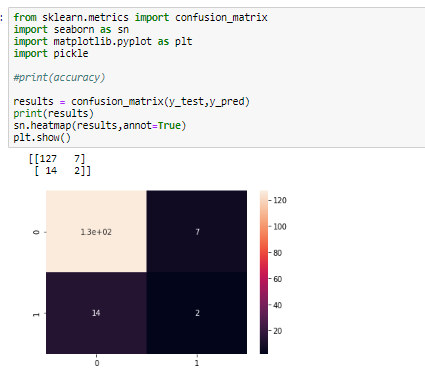


**Efficiency Calculation:**

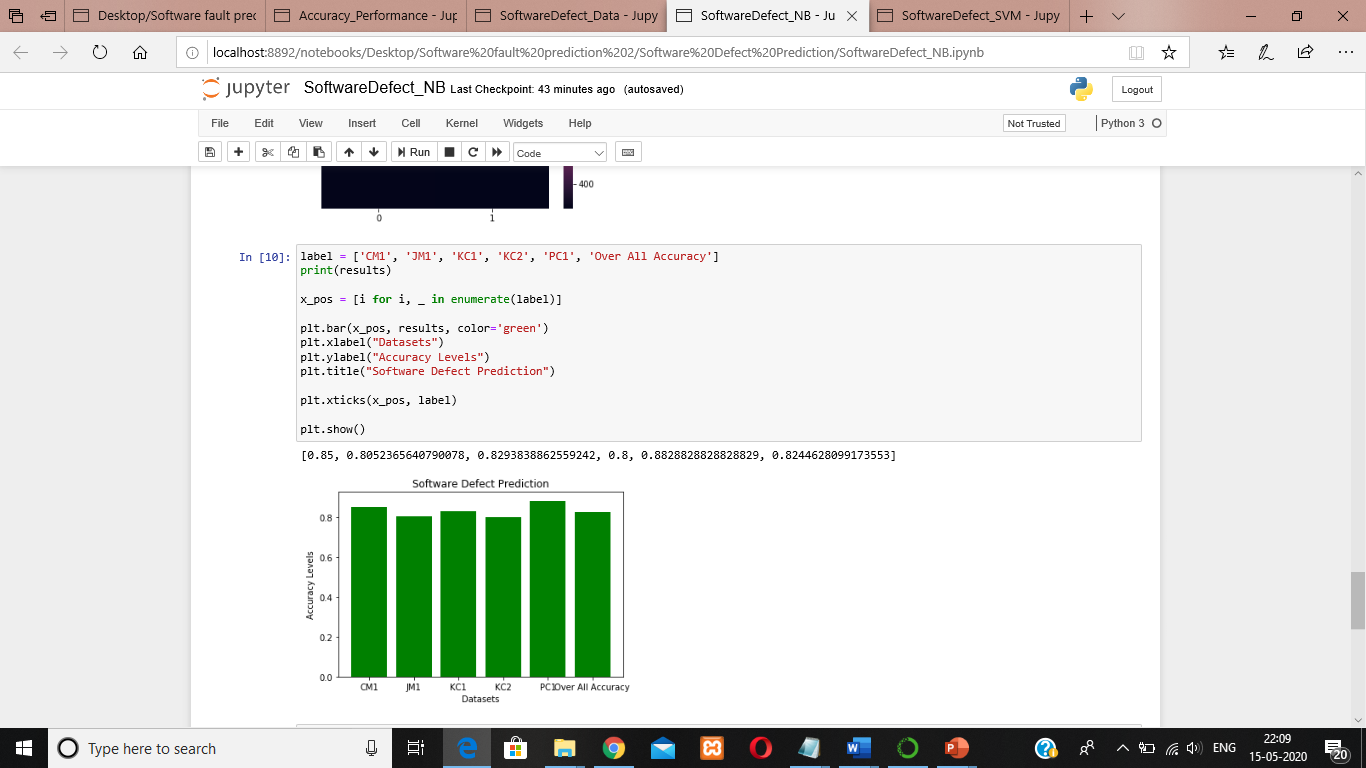
* Calculating accuracy using Naïve Bayes:



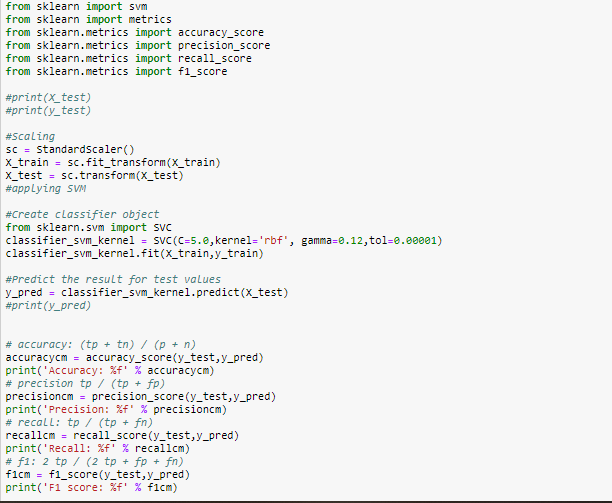
* Confusion matrix using Naïve Bayes:



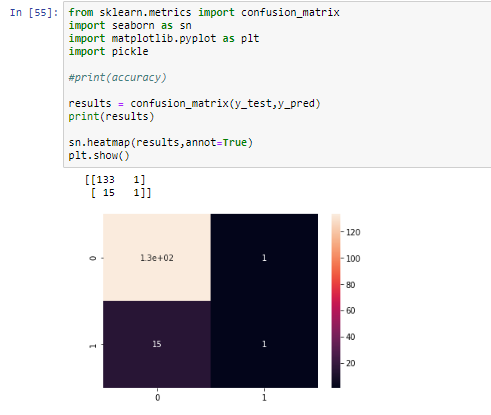
* Accuracy levels of different datasets using naïve bayes.



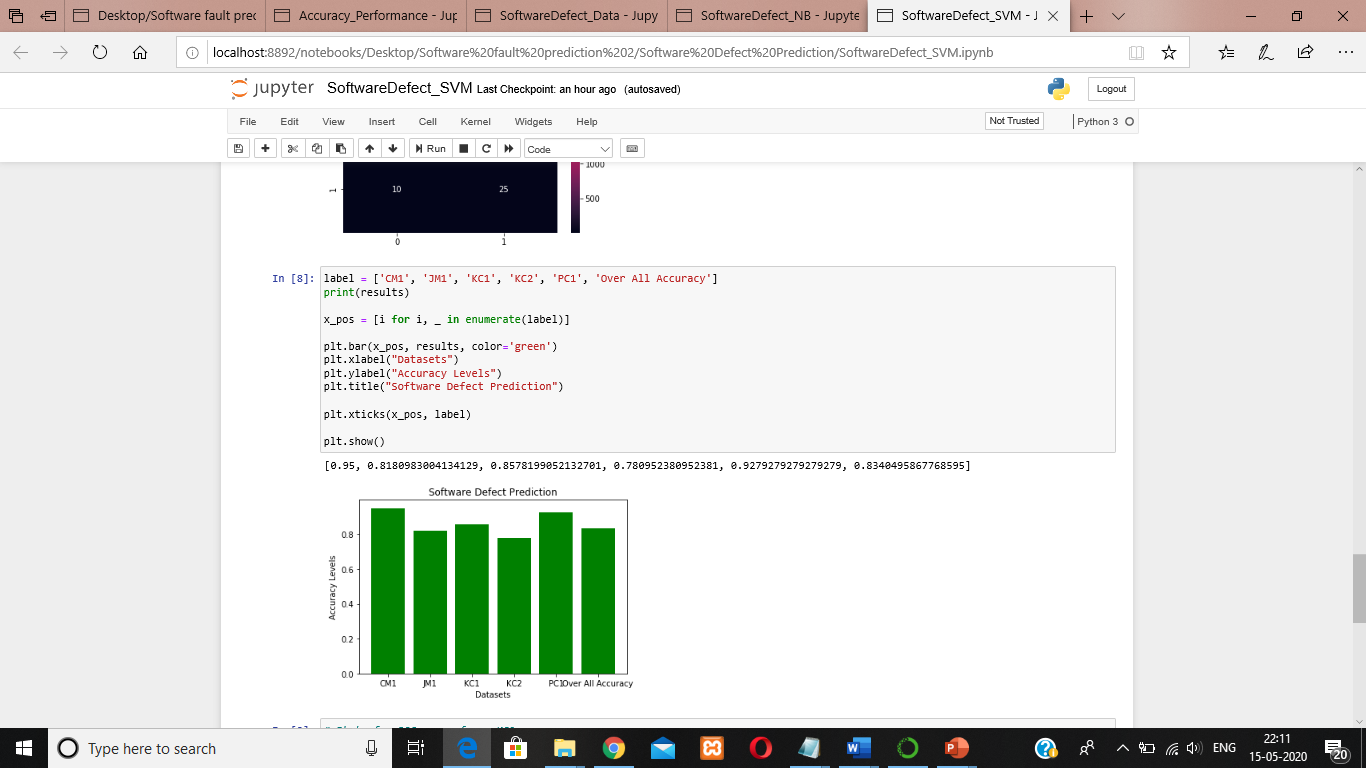
* Calculating accuracy using SVM:



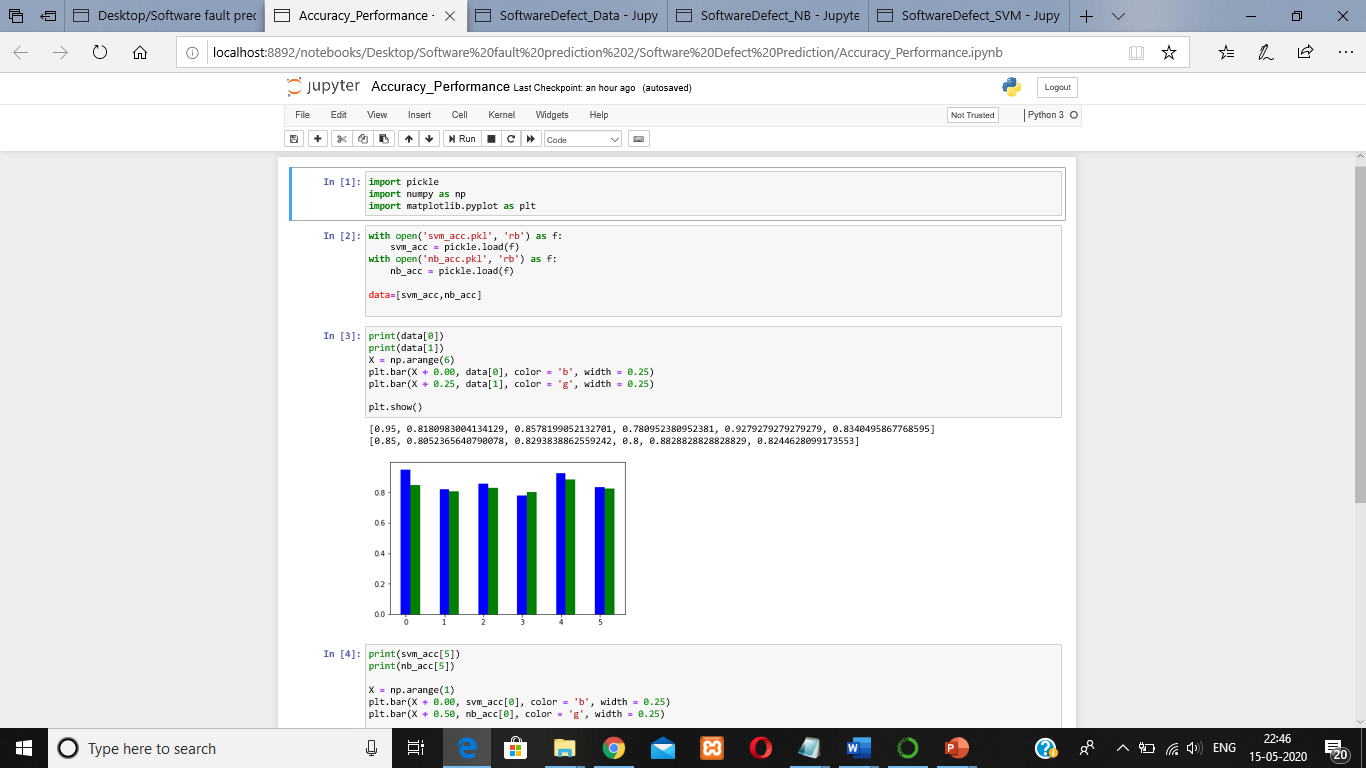
* Confusion matrix using SVM:



* Accuracy levels of different datasets using SVM.



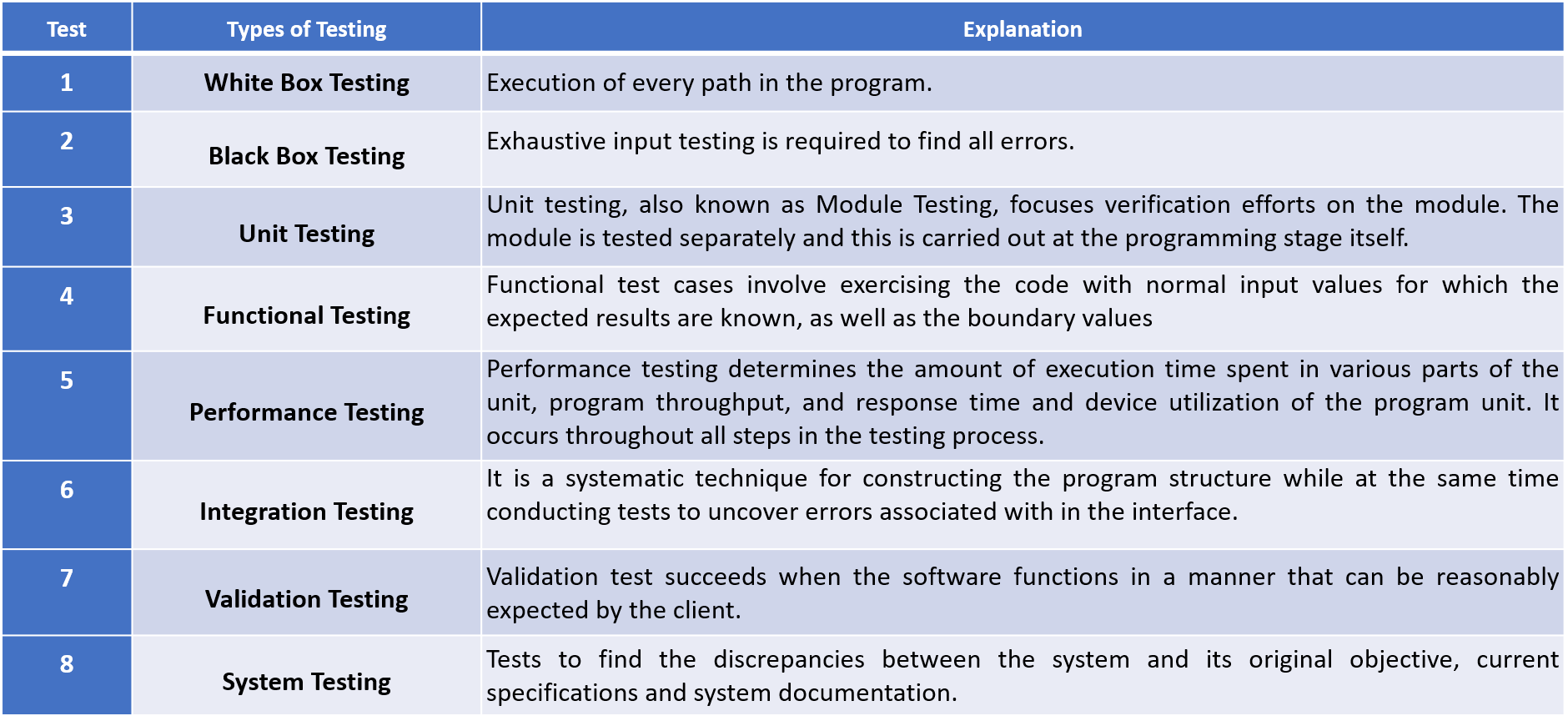
* Comparing accuracy performance between Naïve Bayes and SVM:



# 

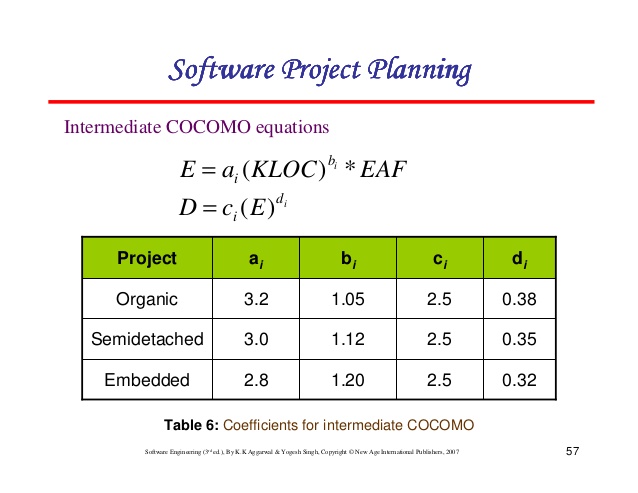
# **CHAPTER 5**

# **VERIFICATION AND VALIDATION**



## **5.4 COST ANALYSIS**

* Effort(E) = a\*(KLOC)^b (in Person-Months)
* Development Time(D) = c\*(E)^d (in months)
* Average Staff Size (SS) = E/D (in Person)
* Productivity(P) = KLOC/E (in KLOC/Person-month)



*Table 4: Coefficient for Intermediate COCOMO*

**Cost Estimation – Effort**

This project contains roughly ~ 0.9 KLOC (900 LOC)

By putting these values in the formula, the following would arrive:

Effort(E) = a \* (KLOC)^b

=2.4\*(0.9) ^1.05

=2.14865098401 ~ 2-man months

**Cost Estimation – Development Time**

Here E is 2.1487

Development Time (D) = c \* (E)^d

=2.5 \* (2.1487) ^0.38

=3.34323424391~ 3 months

**Cost Estimation – Average Staff Size**

Here E is 2.1487

D = 3.34323

Average Staff Size (SS) = E/D

= 2.1487 / 3.34323

= 0.6427018183 ~ 1 person

**Cost Estimation – Productivity**

Here E is 2.1487

KLOC = 0.9,

Productivity(P) = KLOC/E

= 0.9 / 2.1487

=0.41885791408

## **5.5 DEFECT ANALYSIS**

The Software default prediction should be accurate enough so that based on the prediction the fault can be recognized and hence failure of a project is withheld. The datasets so taken should be actual real time datasets and should have accurate data.

Further if the data is homogenous and is not reduced properly that accuracy is affected and hence fails to predict the fault data properly. If the data sets are not having enough data then it may lead to failure as well. Sometimes the data is very biased and is not up to the mark and hence it may have the risk of failure of prediction properly. The model has to be rigorously trained and tested to ensure the accuracy. We faced lack of model variability and also false errors it the accuracy is improved by the training and testing rigorously.

## **5.6 MC CALL’S QUALITY FACTORS**

### **5.6.1 Product Operation factors**

#### Correctness:

This project correctly stores the data from the dataset. Authentication of the dataset which is correctly done using different ML algorithm.

Software Fault Prediction works as desired i.e. it is showing the accuracy rate and their prediction.

Different ML Algorithms which are working properly and giving result accordingly.

#### Reliability:

Reliability which is one of the most important aspects of software systems. Software development is a complex and complicated process in which software faults are inserted into the code by mistakes during the development process or maintenance. It has been shown that the pattern of the fault’s insertion phenomena is related to measurable attributes of the software.

*Efficiency:*

This project is efficient as it doesn't require any external resource, so all users can easily use it without having any hardware. It is showing accurate result for fault prediction.

#### Usability:

Little or no training is required to use this project. The user just has to upload the dataset then he/she can use all the functionalities of the project. While using the project itself, the user will able to find that how to use this software for prediction purpose. So, it's easy to use this project.

#### Integrity:

This project uses different metrics and reduce feature set technique to authenticate where the fault can be settled and what percent of fault will be corrected so the user can find easily. Also, the functionality of this project can be altered only by the administrator i.e. the source code of the project can be seen and altered only by the admins.

### 

### **5.6.2 Product Revision Factors**

#### Maintainability:

There is no external hardware used in this project which will have to be maintained later. The code is fixed, there is no maintenance required in the code in the existing project but for improving this project changes can be made.

#### Flexibility:

This project is very flexible to changes, there will be no loss in terms of cost. Any changes can be made and any additional functionality can be added to the project whenever required.

Testability:

We have performed both unit testing and integration testing, all requirements, functionalities can be verified easily.

### **5.6.3 Product Transition Factors**

#### Portability

Since this project is coded in Python, it can easily run on any operating system and perform all the functionalities as desired.

#### Reusability

All the components of this project can be reused together or individually for any other purpose or in any other product.

#### Interoperability

This project’s interfaces are completely understood, to work with other products or systems, at present or in the future, in either implementation or access, without any restrictions.

# **CHAPTER 6**

# **EXPERIMENT RESULTS AND ANALYSIS**

## **6.1 RESULTS**

* Confusion Matrix is used for evaluating the performance of a system. In this case, Software Fault Prediction. The algorithm we use for software fault prediction is the SVM (Support Vector Machine) algorithm and Naïve Bayes algorithm. They are machine learning algorithms used to identify and predict error.
* Other functionalities of this project are already tested.
* **true positives (TP):** This is the case when dataset was detected correctly and its data is present in the database.
* **true negatives (TN):** This is the case when dataset was not detected, and its data is not present in the database.
* **false positives (FP):** This is the case when fault was detected, but its data is not present in the database.
* **false negatives (FN):** This is the case when fault is not detected, but its data is present in the database.

Confusion Matrix (for Naïve Bayes Algorithm):

Confusion Matrix is used for evaluating the performance of a system. In this case, Software fault prediction system.

No. of tests performed, n=3025 In Naïve Bayes Algorithm

|  |  |  |
| --- | --- | --- |
|  | **Detected (Positive)** | **Rejected (Negative)** |
| **Present in Database** | 2397 (TP) | 97 (FN) |
| **Not Present in Database** | 111 (FP) | 420 (TN) |

Table 5: Confusion Matrix Table for Naïve Bayes

Table 6 represents the confusion matrix for 3025 input of codes from dataset. For this, 3025 outputs are correct and 208 outputs are wrong.

Confusion Matrix (for SVM Algorithm):

Confusion Matrix is used for evaluating the performance of a system. In this case, Software fault prediction system.

No. of tests performed, n=3025 In SVM Algorithm

|  |  |  |
| --- | --- | --- |
|  | **Detected (Positive)** | **Rejected (Negative)** |
| **Present in Database** | 2498 (TP) | 25 (FN) |
| **Not Present in Database** | 10 (FP) | 492 (TN) |

Table 6: Confusion Matrix For SVM

Table 6 represents the confusion matrix for 3025 input of codes from dataset. For this, 2817 outputs are correct and 35 outputs are wrong.

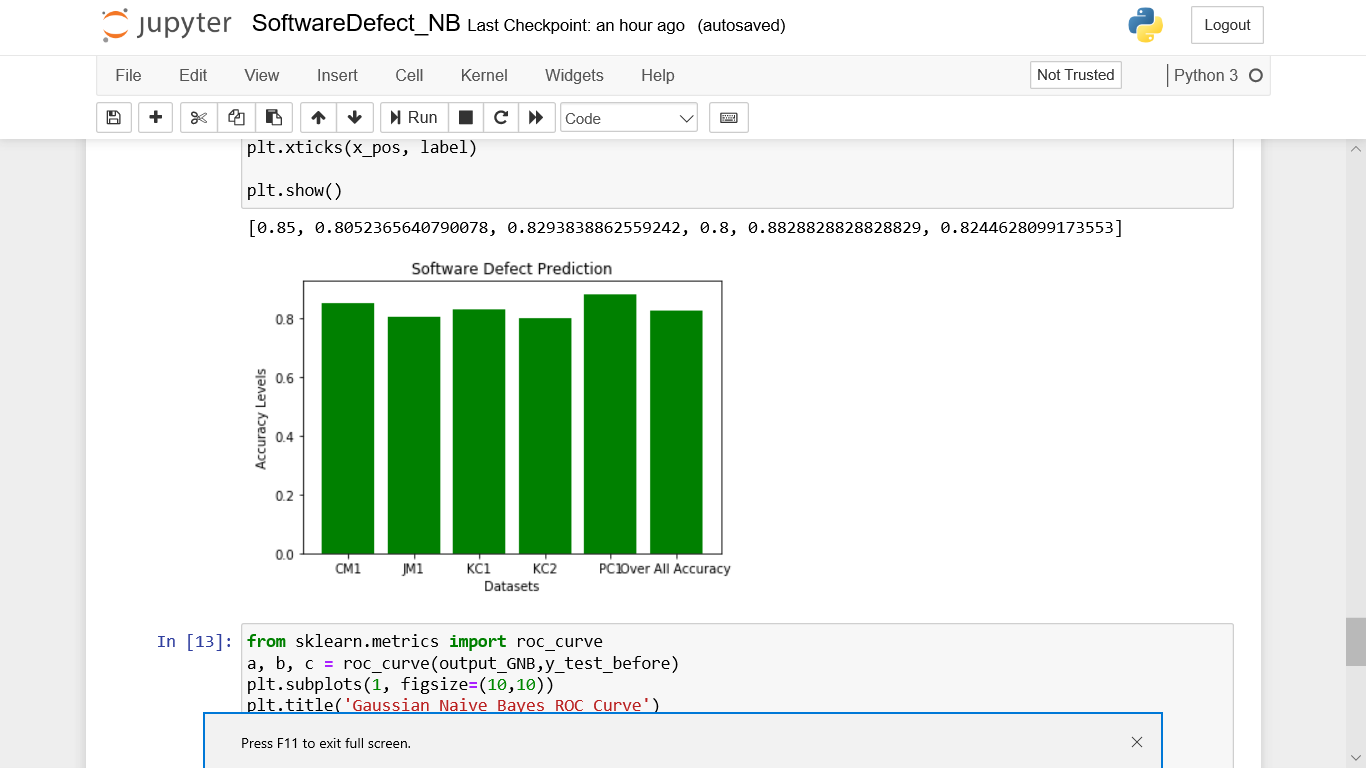
****

Figure 12: Accuracy Levels of Different datasets using Naïve Bayes

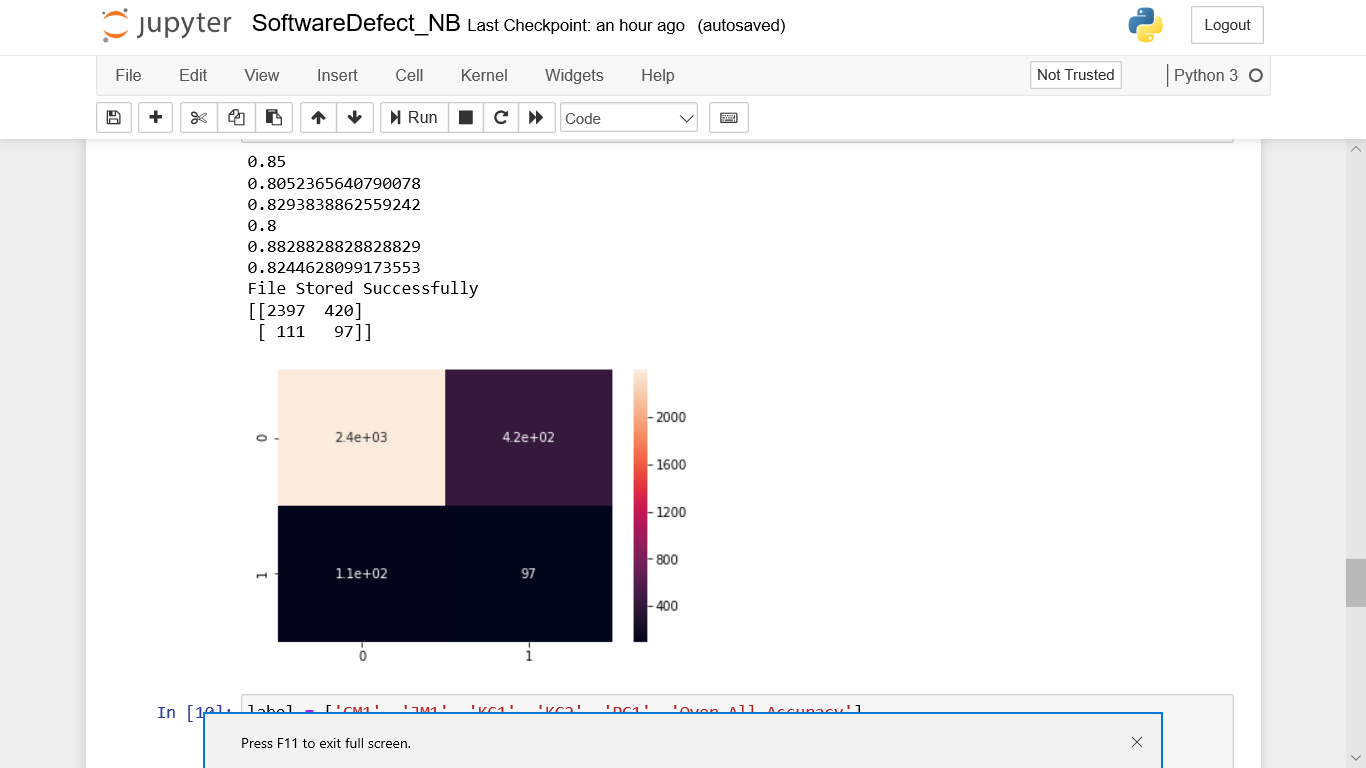
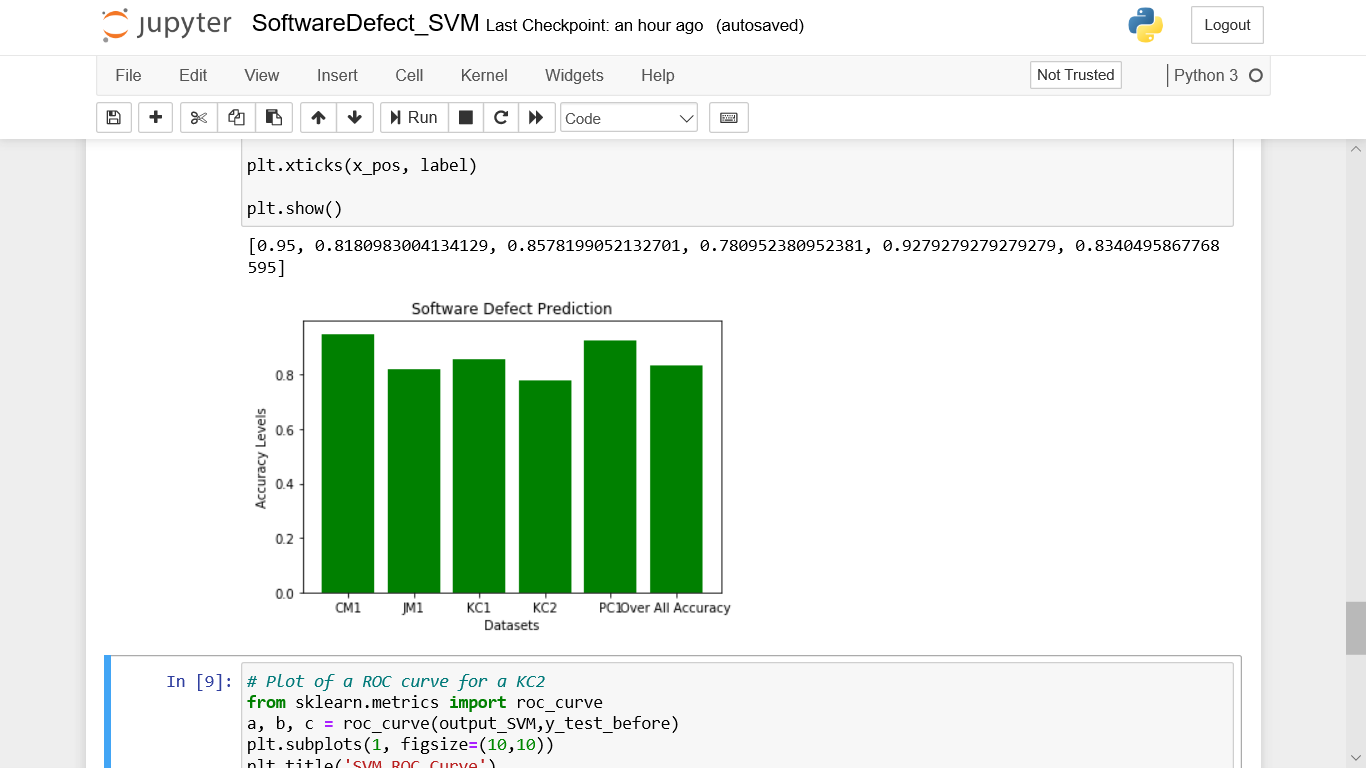


Figure 13 : Confusing Matrix Result using Naive Bayes

Figure 14: Accuracy Levels of Different datasets using SVM:

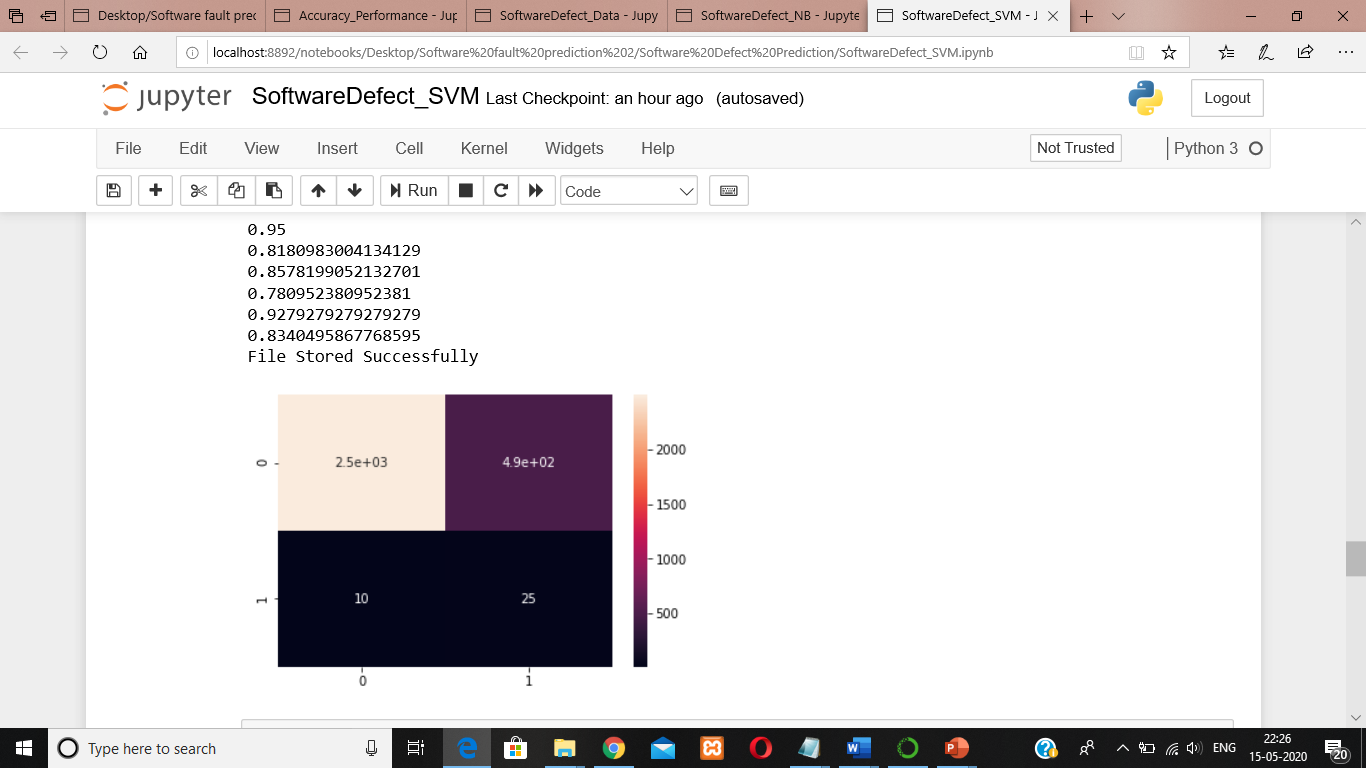


Figure 15: Confusing Matrix Result using SVM:

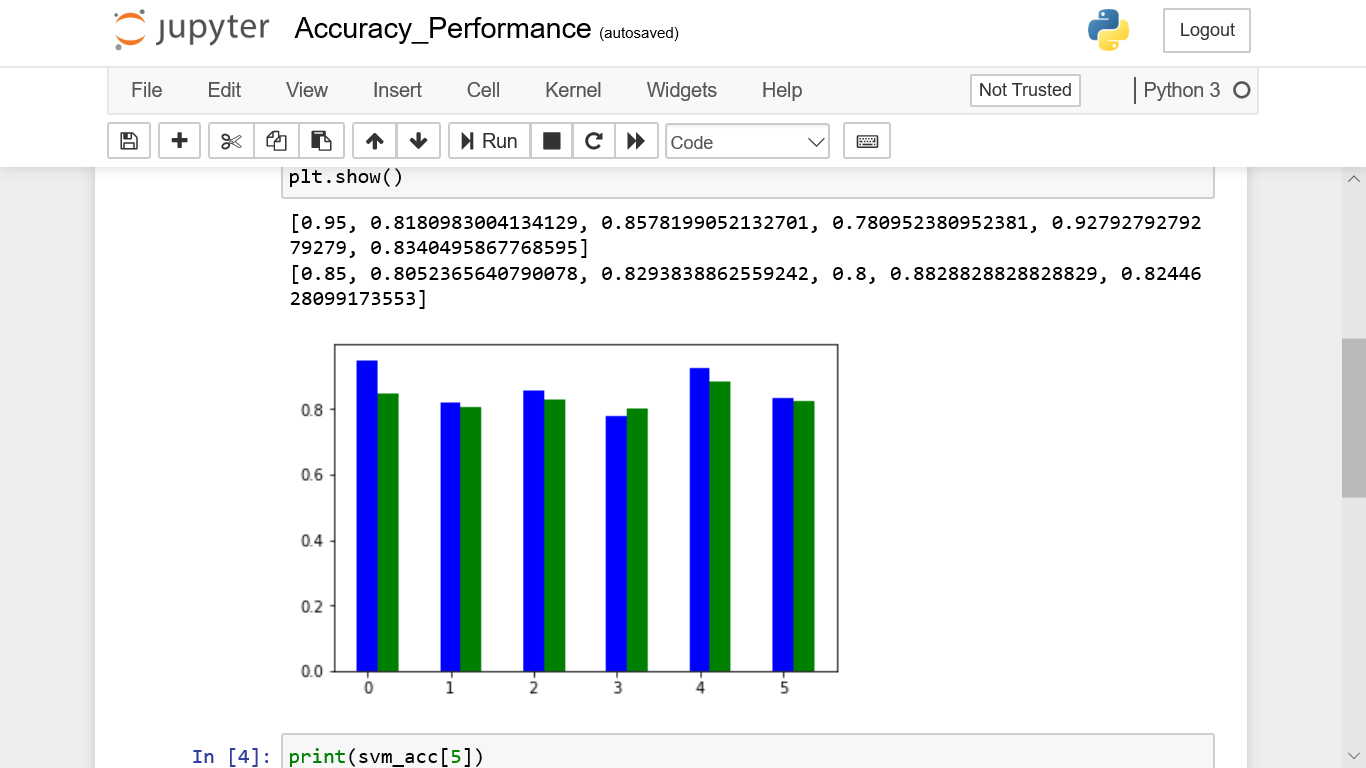
****

Figure 16: Comparing accuracy performance between naïve bayse and SVM

Blue-SVM

Green-NB

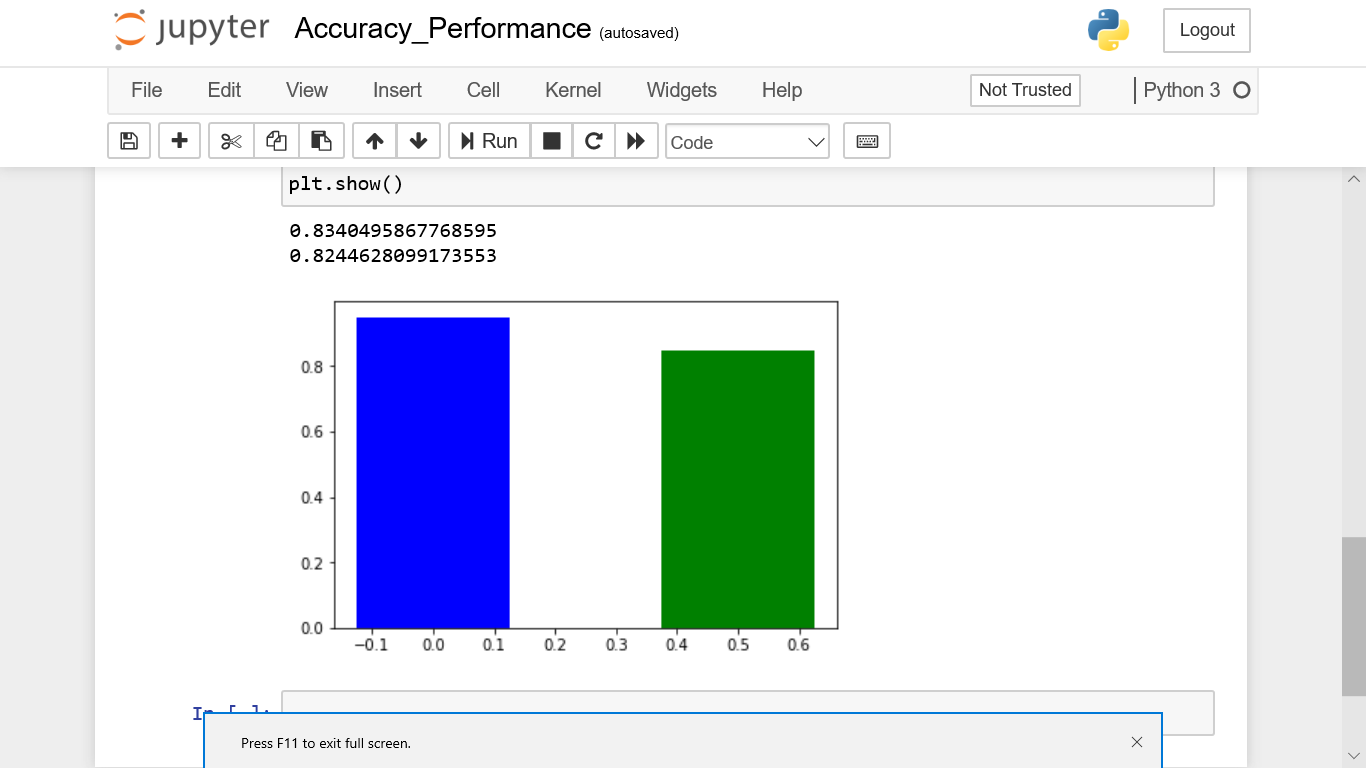


Figure 17 : Comparing accuracy performance between Naïve Bayes and SVM

## 

## **6.2 RESULT ANALYSIS**

**For Naïve Bayes Algorithm:**

Accuracy: Overall, how often is the classifier correct?

(TP+TN)/total = (127+2)/150 = 0.86

Misclassification Rate: Overall, how often is it wrong?

(FP+FN)/total = (14+7)/150 = 0.14

equivalent to 1 minus Accuracy, also known as "Error Rate"

True Positive Rate: When data is present in dataset, how often does it detect the fault?

TP/present in dataset = 127/134 = 0.947

also known as "Sensitivity" or "Recall"

False Positive Rate: When data is not present in dataset, how often does it detect the fault?

FP/not present in dataset = 14/16 = 0.875

True Negative Rate: When data is not present in dataset, how often does it not detect the fault?

TN/not present in dataset = 2/16 = 0.125

equivalent to 1 minus False Positive Rate, also known as "Specificity"

Precision: When it detects fault, how often is it correct?

TP/Face detected = 127/141 = 0.879

**For SVM Algorithm:**

Accuracy: Overall, how often is the classifier correct?

(TP+TN)/total = (133+1)/150 = 0.89

Misclassification Rate: Overall, how often is it wrong?

(FP+FN)/total = (15+1)/150 = 0.11

equivalent to 1 minus Accuracy, also known as "Error Rate"

True Positive Rate: When data is present in dataset, how often does it detect the fault?

TP/present in dataset = 133/134 = 0.992

also known as "Sensitivity" or "Recall"

False Positive Rate: When data is not present in dataset, how often does it detect the fault?

FP/not present in dataset = 15/16 = 0.9375

True Negative Rate: When data is not present in dataset, how often does it not detect the fault?

TN/not present in dataset = 1/16 = 0.0625

equivalent to 1 minus False Positive Rate, also known as "Specificity"

Precision: When it detects fault, how often is it correct?

TP/Face detected = 133/148 = 0.898

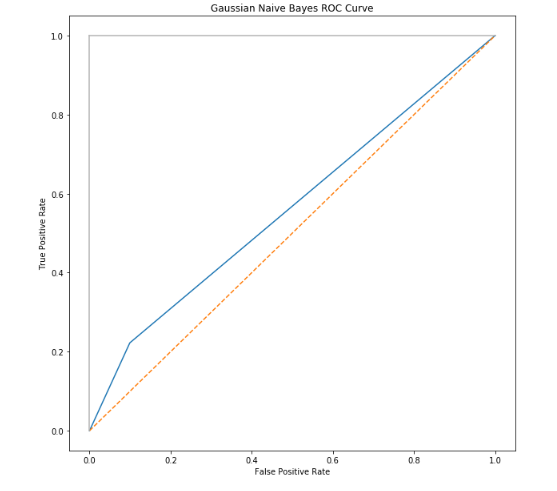


Figure 18: ROC CURVE FOR GAUSSIAN NAÏVE BAYSE

Figure 18 represents the accuracy graph. The graph shows that the accuracy increases linearly with the number of lines of data. But after reaching a threshold value it remains almost constant. The threshold value for the number of data for our project is 127.

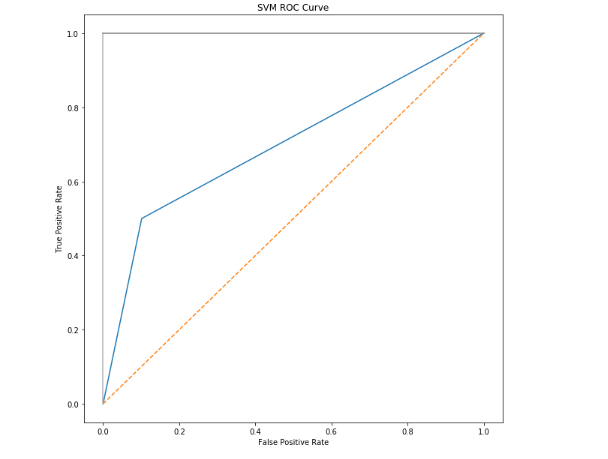


Figure 19: ROC Curve for SVM

Figure 19 represents the Receiver Operator Characteristics (ROC curve) which is drawn using multiple confusion matrix at different threshold values.

## 

## **6.3 CONCLUSION AND FUTURE WORK**

In this paper, we have used two different databases from the UCI machine learning for bad software prediction. the data is trained using different classifiers such as Naïve Bayes and SVM. From the study, it was found that SVM gives a very accurate 89% accuracy in the case of software feature and in the case of Naive Bayes classifier it provides 86% higher accuracy. We also replicated the work of Applying machine learning to predict software fault proneness using change metrics, static code metrics, and a combination of them and explored what kind of metrics provide the best accuracy. In most cases, using the combination of static code and change code metrics is better than using only static code metrics or using only change metrics.

The future work will include using the same classifiers on other datasets from Eclipse or other software projects, as well as exploring additional metrics and other classifiers.

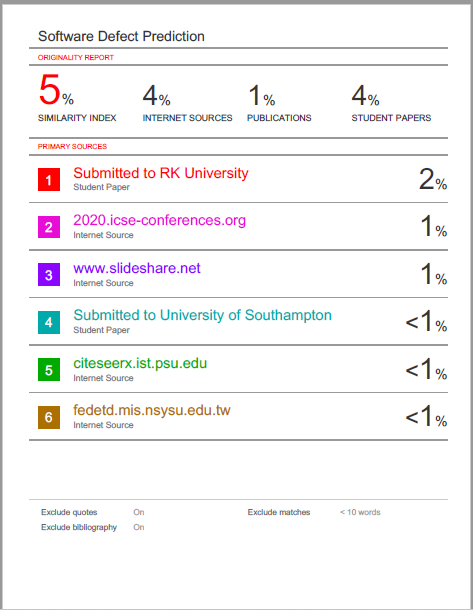
# 

# **REFRENCES**

1. X. Yang, D. Lo, X. Xia, Y. Zhang, and J. Sun, “Deep learning for just-in-time defect prediction,” in QRS’15: Proc. of the International Conference on Software Quality, Reliability and Security , 2015.
2. Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi, “A large-scale empirical study of just-in-time quality assurance,” TSE , vol. 39, no. 6, pp. 757–773, 2013.
3. T. Jiang, L. Tan, and S. Kim, “Personalized defect prediction,” in ASE , 2013, pp. 279–28.
4. M. Tan, L. Tan, S. Dara, and C. Mayeux, “Online defect prediction for imbalanced data,” in ICSE’15, pages 99–108.
5. S. Wang, T. Liu, and L. Tan, “Automaticallylearning semantic features for defect prediction,” in ICSE’16: Proc. of the International Conference on Software Engineering , 2016.
6. Y. Kamei, S. Matsumoto, A. Monden, K.-i. Matsumoto, B. Adams, and A. E. Hassan, “Revisiting common bug prediction findings using effort-aware models,” in ICSM , 2010, pp. 1–10.
7. Jian Li, Pinjia He, Jieming Zhu, and Michael R. Lyu, “Software Defect Prediction via Convolutional Neural Network”.
8. P. D. Singh and A. Chugh, “Software Defect Prediction Analysis Using Machine Learning Algorithms,” in International Conference on Cloud Computing, Data Science & Engineering – Confluence, 2017.
9. G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” Science, vol. 313, no. 5786, pp. 504–507, 2006.
10. G. E. Hinton, “Learning multiple layers of representation,” Trends in cognitive sciences, vol. 11, no. 10, pp. 428–434, 2007.

# 

# **PLAGIARISM REPORT**

****