

# **SWE-504 EMPIRICAL SOFTWARE ENGINEERING PRACTICAL FILE**

*Submitted towards the partial fulfilment  
of the requirements of the award of the degree of*

## **Master of Technology In Software Engineering**

**Submitted To:-**

**Dr. Abhilasha Sharma**

Associate Professor

Department of Software Engineering

**Submitted By:-**

Aman Chauhan (24/SWE/08)

II SEM, I YEAR



**Delhi Technological University**

(FORMERLY Delhi College of Engineering)

SHAHBAD DAULATPUR, BAWANA ROAD, DELHI – 110042

April, 2025

# INDEX

S.No.	EXPERIMENT NAME	DATE	SIGNATURE
1	Consider an empirical study of each type (Experiments, Survey Research, Systematic Review, Postmortem Analysis and Case Study). Identify the following components for an empirical study. a) Identify parametric and non parametric tests. b) Identify Independent, dependent and confounding variables. c) Is it Within-company and cross-company analysis? d) What is the name of datasets used in each empirical study? e) What type of datasets (Proprietary and open source software) are used?		
2	Defect detection activities like reviews and testing help in identifying the defects in the artifacts (deliverables). These defects must be classified into various buckets before carrying out the root cause analysis. Following are some of the defect categories: Logical, User interface, Maintainability, and Standards. In the context of the above defect categories, classify the following statements under the defect categories.		
3	Consider any prediction model of your choice. a)Analyze the dataset that is given as an input to the prediction model b) Find out the quartiles for the used dataset. c) Analyze the performance of a model using various performance metrics.		
4	Consider defect dataset and perform following feature reduction techniques using Weka tool. Validate the dataset using 10-cross validation. a. Correlation based feature evaluation b. Relief Attribute feature evaluation c. Information gain feature evaluation d. Principle Component		
5	Online loan system has two modules for the two basic services, namely Car loan service and House loan service. The two modules have been named as Car_Loan_Module and House_Loan_Module. Car_Loan_Module has 2000 lines of uncommented source code. House_Loan_Module has 3000 lines of uncommented source code. Car_Loan_Module was completely implemented by Mike. House_Loan_Module was completely implemented by John. Mike took 100 person hours to implement Car_Loan_Module. John took 200 person hours to implement House_Loan_Module. Mike's module had 5 defects. John's module had 6 defects. With respect to the context given, which among the following is an INCORRECT statement? Identify the null and alternate hypothesis for the following options. Justify and Choose one: a.John's Quality is better than Mike's Quality b.John's Productivity is more than Mike's Productivity c.John introduced more defects than Mike d.John's Effort is more than Mike's Effort.		

<b>6</b>	Statistical Hypothesis Testing in R-Statisticians use hypothesis testing to formally check whether the hypothesis is accepted or rejected. Consider an example or data of your choice and identify the following: (A) State the Hypotheses (B) Formulate an Analysis Plan (C)Analyze Sample Data (D) Interpret Results (E) Estimate type-I and type-II error		
<b>7</b>	Consider the defect dataset and implement the following statistical test using the SPSS tool. a. t-test b.Chi-Square Test c.Wilcoxon Signed Test d. Friedman Test e. Kruskal Wallis Test		
<b>8</b>	Why is version control important? How many types of version control systems are there? Demonstrate how version control is used in a proper sequence (stepwise).		
<b>9</b>	Demonstrate how Git can be used to perform version control?		
<b>10</b>	Validate the results obtained in experiment 3 using 10-cross validation, hold out validation or leave one out cross-validation.		

# EXPERIMENT 1

## AIM:

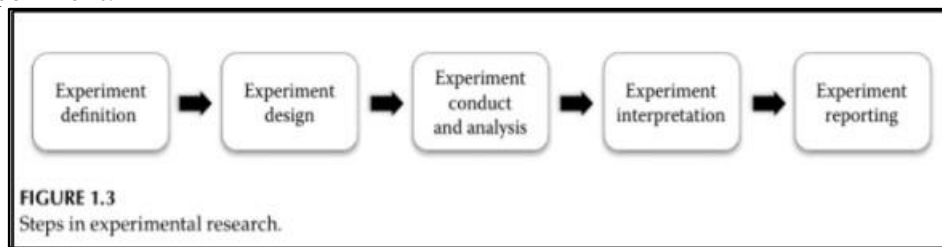
Consider an empirical study of each type (**Experiments, Survey Research, Systematic Review, Postmortem Analysis and Case Study**). Identify the following components for a empirical study.

- Identify parametric and non parametric tests.
- Identify Independent, dependent and confounding variables.
- Is it Within-company and cross-company analysis?
- What is the name of datasets used in each empirical study?
- What type of datasets (Proprietary and open source software) are used?

## 1) Experiments

### **THEORY:**

An experimental study tests the established hypothesis by finding the effect of variables of interest (independent variables) on the outcome variable (dependent variable) using statistical analysis. If the experiment is carried out correctly, the hypothesis is either accepted or rejected. Experiments are carried out in a controlled environment. The experiment must also control the confounding variables, which may affect the accuracy of the results produced by the experiment.



### **EXPERIMENT:**

The experiment used for this study is: **Experiments with Incremental Concept Formation: UNIMEM** by **MICHAEL LEBOWITZ**. In this paper, UNIMEM, an artificial intelligence system that learns by observation is presented. One domain on which we tested UNIMEM involved Congressional voting records. Instances were formed from the votes of each U.S. Congressman on a number of major issues (taken from The 1983 American Political Almanac) combined with information about the district and state represented. One advantage of this domain for research purposes is that people have strong intuitions about the kinds of generalizations that should be found.

#### **a) Identify parametric and non-parametric tests.:**

The Friedman test and Wilcoxon signed-rank tests have been used as non-parametric tests to confirm that the results obtained were significant or not.

#### **b) Identify Independent, dependent, and confounding variables.**

The independent variables are GND1, GND2, GND3, and so on that describe the generalizations among the voters. One top-level generalization, GND2, describes congressmen from agricultural states with high levels of school expenditures who voted for an education bill,

parks in Alaska, and so forth. The 24th Texas Congressional District is stored under this generalization, along with two sub-generalizations. Someone familiar with U.S. politics would describe this voting pattern as 'liberal.' Similarly, the second top-level node in this example, GND3, would be considered 'conservative.' The dependent variable describes whether a voter is “for” or “against” voting for a party.

**c) Is it Within-company and cross-company analysis?**

The analysis is a standalone experiment and has thus been performed by researchers within a single company. Although, the data used has been collected from multiple sources.

**d) What is the name of the datasets used in this empirical study?:**

In the run described here, UNIMEM has been presented with 100 instances, each containing 15 votes and about 21 other features, is We expected to find generalizations that related the various votes to each other (e.g., 'liberal' and 'conservative' ideologies), along with others that related the votes to the states and districts represented (e.g., someone representing a highly urban state would support bills that help cities). Indeed UNIMEM formed concepts of this sort.

**e) What type of datasets (Proprietary and open source) are used?**

The datasets used in this study are unique and proprietary which have been used to assess patterns in voting records.

## 2)Survey Research

### **THEORY:**

Survey research identifies features or information from a large scale of a population. For example, surveys can be used when a researcher wants to know whether the use of a particular process has improved the view of clients toward software usability features. This information can be obtained by asking the selected software testers to fill questionnaires. Surveys are usually conducted using questionnaires and interviews. The questionnaire/survey can be used to detect trends and may provide valuable information and feedback on a particular process, technique, or tool. The questionnaire/survey must include questions concerning the participant’s likes and dislikes about a particular process, technique, or tool.

Surveys are classified into three types (Babbie 1990) —descriptive, explorative, and explanatory.

### **EXPERIMENT:**

Here, I conduct the experiment on: *Evaluation and Satisfaction Survey on the Interface Usability of Online Publishing Software [2014] Ying-Jye Lee and Chia-Ju Lin*. The Usability of Online Publishing Software is related to consumers’ acceptance or intention of product purchase. The systems designed by designers are often different from the users’ imagination and understanding of such systems. The better satisfaction would please the users more.

**a) Identify parametric and non parametric tests.:** The statistical test : ANOVA is used to compare the learnability of the different Online Publishing Software ( Tintint, Hypo,

Photobook) . Analysis of variance (ANOVA) is a collection of statistical models and their associated estimation procedures (such as the "variation" among and between groups) used to analyze the differences among means in a sample.

**b) Identify Independent, dependent and confounding variables.**

The dependent variable (or response variable) is the output produced by analyzing the effect of the independent variables. The dependent variables are presumed to be influenced by the independent variables. The independent variables are the causes and the dependent variable is the effect. Apart from the independent variables, confounding variables(extraneous variables) may affect the outcome (dependent) variable. Randomization can nullify the effect of confounding variables.

In [ Ying-Jye Lee, 2014], no cause-effect relationship or dependence between variables is studied. Although it uses **Learnability** and the **number of user errors** as the evaluation criteria in the paper. With the experimental data, the representative online publishing software (Tintint, Hypo, Photobook) is evaluated and compared to the interface usability testing on the basis of the evaluation criteria. Furthermore, users' subjective satisfaction with the online publishing software interface is investigated to understand the users' demands for online publishing software to propose better interface improvement suggestions.

**c) Is it Within-company and cross-company analysis?** It is a cross-company analysis which performs a satisfaction survey on the Interface Usability of some representative Online Publishing Softwares (TinTint, Photobook, and Hypo). Designers and General Users participated in the questionnaire.

**d) What is the name of datasets used in each empirical study?:**

The Focus Group method was used in this paper to conduct the survey. **Focus Group** is often used for evaluating users' demands for objective and representative solutions. The Focus Group members are divided into two groups, with six interviewed members in each group to discuss the representative online publishing software as the research samples. The subjects were requested to freely browse the contents, software instruction, instructional video, or operation software of each software website for three minutes before the experiment, aiming to be close to the users' habits. The questionnaire is referred to **QUIS** (Questionnaire for User Interface Satisfaction), developed by Chin et al.

Subjects of the survey are classified into two groups. Part of the participants in Focus Group are **designers**, while the other part is the **users** with long-term experiences in online publishing software. Total of 30 subjects are included in the evaluation of software usability. Most of the subjects in this study are aged 20~29.

**e) What type of datasets (Proprietary and open source) are used?**

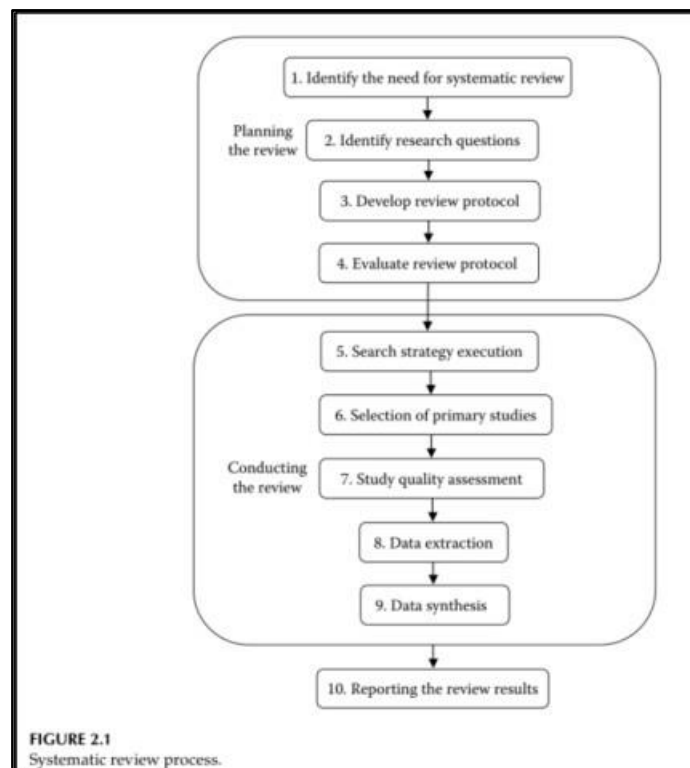
Focus Group is utilized in this study for screening the representative online publishing software (TinTint, Photobook, and Hypo) as the testing software for research. The Dataset can be considered to be Open Source.

**REFERENCES:**

### 3) Systematic Reviews (SR)

#### THEORY:

Review of existing literature is an essential step before beginning any new research. Systematic reviews (SRs) synthesize the existing research work in such a manner that can be analyzed, assessed, and interpreted to draw meaningful conclusions. The aim of conducting an SR is to gather and interpret empirical evidence from the available research with respect to formed research questions. The benefit of conducting an SR is to summarize the existing trends in the available research, identify gaps in the current research, and provide future guidelines for conducting new research. The SRs also provide empirical evidence in support or opposition of a given hypothesis.



#### EXPERIMENT:

The SR in ESE, we use here is co-authored by **Dr. Ruchika Malhotra, Dr. Kusum Lata (2020)** with a study size of 36 research papers on the topic of Software Maintainability Prediction(SMP) using Machine Learning Techniques. It systematically reviewed the prediction models from January 1990 to October 2019 for predicting software maintainability. On investigating the existing research, the authors found that various machine learning (**ML**), statistical (**ST**), and hybridized (**HB**) techniques have been applied to develop predictive models for SMP. **Software maintainability prediction(SMP)** in the earlier stages of software development involves the construction of models for the accurate estimation of maintenance effort. This systematic review analyzed and summarized the use of data sets, software metrics, techniques, performance measures, and statistical tests in the estimation of the lines of code added, deleted, and modified during the maintenance phase.

Some problems related to SMP are: studying the effect of refactoring on software maintainability, coming up with a software maintainability predictor or metrics on the basis of the OO metrics of the software modules and the maintenance methods employed for maintaining the software.

A characteristic of SRs are Quality Assessments which were analysed very nicely in **Dr.Ruchika Malhotra, Dr Kusum Lata ( 2020 )**. While planning the study, research questions identified are listed below:

Research question
Which techniques have been used for SMP? Which data sets have been used for SMP?
Which metrics have been used for SMP?
Which performance evaluators have been used for SMP?
What are the commonly used performance measures to judge performance of prediction models?
Which validation method has been used for developing models for SMP?
Which statistical tests of significance have been used to statistically examine the predictive capability of SMP models?
What is the predictive ability of different techniques used for SMP?
What is the performance of SMP models that are built with the help of ML techniques?
What is the overall performance of SMP of models built using ST techniques?
What is the overall performance of SMP models developed using HB techniques?
Is there any one technique that outperforms the other in handling data sets with varying properties?
What are the potential threats to validity in SMP?
What are advantages and disadvantages of the techniques used for SMP?

#### **a) Identify parametric and non parametric tests ?**

Parametric and Non-Parametric are not needed for SRs, instead strict quality assessments of research and case studies which will be used in the systematic review is conducted. The assessment criteria is based on the relevance of a particular study to the research questions and the quality of the processes and methods used in the study. In addition, quality assessment questions must focus on experimental design, applicability of results, and interpretation of results.

Eg: Are the research questions answered validly in the research? Is the scope of the study clear? Are limitations stated? Is experiment design stated clearly? Is it based on sound practices and ethics of research?

**b) Identify Independent, dependent and confounding variables.** The aim of conducting an SR is to gather and interpret empirical evidence from the available research with respect to finalised research questions.



**c) Is it Within-company and cross-company analysis?** SRs summarize high-quality research on a specific area. They provide the best available evidence on a particular technique or technology and produce conclusions that can be used by the software practitioners and researchers to select the best available techniques or methodologies. SR opens avenues for new research as it provides future directions for researchers based on thorough analysis of existing literature.

**d) What is the name of datasets used in Systematic Review?:**

A Systematic Review critically analyses the existing literature and research on a particular topic. During data collection, studies related to the topic are collected.

In our example,

The review (Malhota, 2020 on SMP) aimed to select the most relevant and significant papers on the subject under study to provide concrete results of the investigation. During data collection 206 papers were extracted but only research of utmost quality was ultimately used for conducting the Systematic Review. The Systematic review included research papers on 4 topics listed below:

- Empirical studies on SMP using CHANGE (line of code added, deleted, and modified) as the dependent variable.
- Empirical studies using ST techniques for developing a prediction model for SMP.
- Empirical studies using ML techniques for building a prediction model for SMP.
- Empirical studies comparing two or more techniques for SMP.

**e) What type of datasets (Proprietary and open source software) are used?**

The review (Malhota, 2020 on SMP) aimed to select the most relevant and significant papers on the subject under study to provide concrete results of the investigation. During research 206 papers were extracted but only research of utmost quality was ultimately used for conducting the Systematic Review. The Systematic review included research papers on 4 topics listed below:

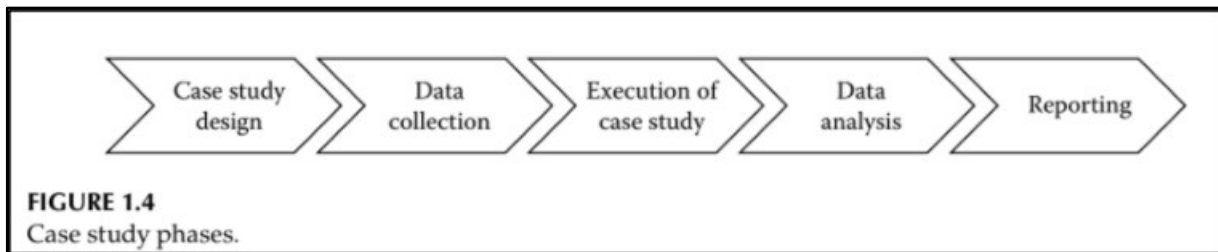
- Empirical studies on SMP using CHANGE (line of code added, deleted, and modified) as the dependent variable.
- Empirical studies using ST techniques for developing a prediction model for SMP.
- Empirical studies using ML techniques for building a prediction model for SMP.
- Empirical studies comparing two or more techniques for SMP.

#### **4) Case Study**

A case study is “an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident.” (Yin 2002). Case study research allows software industries to evaluate a tool, method, or process. The effect of a change in an organization can be studied using case study research. Case studies increase the understanding of the phenomenon under study.

For example, a case study can be used to examine whether a unified model language (UML) tool is effective for a given project or not. The initial and new concepts are analyzed and explored by exploratory case studies, whereas the already existing concepts are tested and improvised by confirmatory case studies.

The phases included in the case study are presented in figure below.



The case study design phase involves identifying existing objectives, cases, research questions, and data-collection strategies. The case may be a tool, technology, technique, process, product, individual, or software. Qualitative data is usually collected in a case study. This data is collected for the specific research problem at hand, using procedures that fit the research problem best. The sources include interviews, group discussions, or observations.

Finally, the case study is executed, the results obtained are analyzed, and the findings are reported. Case studies are appropriate where a phenomenon is to be studied for a longer period of time so that the effects of the phenomenon can be observed.

#### **EXPERIMENT:**

Case Study used: [Software Engineering for Machine Learning: A Case Study, Saleema Amershi, Andrew Begel, Christian Bird, et. al., Microsoft Research] 2019

This paper describes a study which studies how various Microsoft software teams build software applications with customer-focused AI features. In Microsoft an integrated existing Agile software engineering processes with AI-specific workflows informed by prior experiences in developing early AI and data science applications is used. For conducting the study, the authors asked Microsoft employees about how they worked through the growing challenges of daily software development specific to AI, as well as the larger, more essential issues inherent in the development of large-scale AI infrastructure and applications.

a) **Identify parametric and non parametric tests.:** The statistical test : ANOVA is used to compare the average overall effectiveness (OE) of a team's ML practices divided by application domain (anonymized). Analysis of variance (ANOVA) is a collection of statistical models and their associated estimation procedures (such as the "variation" among and between groups) used to analyze the differences among means in a sample.

No other kind of statistical tests were conducted during case study research.

b) **Identify Independent, dependent and confounding variables.:**

The dependent variable (or response variable) is the output produced by analyzing the effect of the independent variables. The dependent variables are presumed to be influenced by the independent variables. The independent variables are the causes and the dependent variable is the effect. Apart from the independent variables, confounding variables(extraneous variables) may affect the outcome (dependent) variable. Randomization can nullify the effect of confounding variables.

No cause-effect relationship has been studied in the case study. To analyze the responses, Activity Maturity Index (AMI) was used. Some findings are likely to be specific to the Microsoft teams and team members who participated in the interviews and

surveys.(confounding variable). The interviewees had different levels of experiences and were working on different AI projects across Microsoft. This variation of experience levels affects their perception of the engineering challenges to be addressed in their day-to-day practices.

**c) Is it Within-company and cross-company analysis?**

Case studies are majorly conducted by academia and research teams. The practitioners just benefit from the claims made by the case studies about new software technologies and tools. Our example, [ Software Engineering for Machine Learning: A Case Study, Saleema Amershi, Andrew Begel, Christian Bird, et. al., Microsoft Research] synthesized a set of best practices to address issues fundamental to the large-scale development and deployment of ML-based applications. It also presented a ML process maturity metric to help teams self assess how well they work with machine learning and offer guidance towards improvements. (Some findings are likely to be specific to the Microsoft teams and team members who participated in the interviews and surveys)

**d)What is the name of datasets used in each empirical study?:**

Case studies data is “collected for the specific research problem at hand, using procedures that fit the research problem best” . The data may be directly or indirectly collected from the users. Observations can be conducted in order to investigate how a certain task is conducted by software engineers. Informants were chosen from a variety of teams,to get different levels of experience and different parts of the ecosystem (products with AI components, AI frameworks and platforms, AI created for external companies). 14 software engineers, largely in senior leadership roles, were interviewed. An open-ended questionnaire whose focus was on existing work practice, challenges in that work practice, and best practices was designed to which 155 engineers responded.

**e) What type of datasets (Proprietary and open source) are used?**

Case studies data is collected for the specific research problem at hand, using procedures that fit the research problem best. Qualitative data, collected using interviews and surveys, was used for case studies. It is Proprietary data.

**CONCLUSION:**

Here, we gained a better understanding of how case studies have been applied in Software Engineering research and also learnt the importance of case study research approach. Case studies are generally conducted in social science fields like psychology, sociology, political science and business but in the last 20 years, a lot of case studies research is coming up in the field of software engineering. The analytical research paradigm is not sufficient for investigating complex real life issues, involving humans and their interactions with technology. Hence non-analytical research methodologies like case studies are being used by researchers.

**REFERENCES:**

[https://www.microsoft.com/en-us/research/uploads/prod/2019/03/amershi-icse-2019\\_Software\\_Engineering\\_for\\_Machine\\_Learning.pdf](https://www.microsoft.com/en-us/research/uploads/prod/2019/03/amershi-icse-2019_Software_Engineering_for_Machine_Learning.pdf)

## 5) Postmortem Analysis

Post-mortem analysis (PMA) is an empirical study method in software engineering. It is an important, but often forgotten, way of gathering empirical knowledge. PMA is ideally performed either soon after the most important milestones and events or at the end of a project, both in successful and unsuccessful software development projects. The benefit is that post-mortems can often reveal findings more frequently and differently than project completion reports alone.

### THEORY:

PMA is a project based learning technique. An essential component of the post-mortem analysis is the execution of a workshop on the project history. In the literature, this step is called the most important step in the process .

Depending on the size and the schedule of the project, there are many alternative ways and tools for conducting a PMA. To ensure a successful project review, it is essential to capture the right (unbiased) information and to plan the process carefully. The thorough analysis of the proposed PMA process descriptions provided a framework for creating a general post-mortem analysis process (Figure 1). It illustrates the main steps needed for all systematic post-mortem analyses.

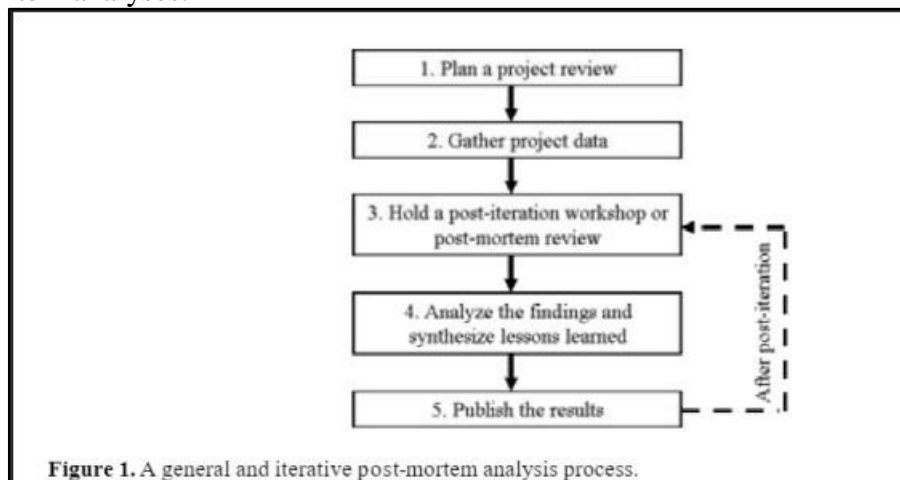


Figure 1. A general and iterative post-mortem analysis process.

The PMA process is made up of five steps:

**1. Identifying company success factors** Success factors of the company can be cost and schedule compliance, high performance quality, experienced employees, etc. The goal clarity of improvement measures is sharpened by the examination of the processes in connection with the success factors of the organization. **The Product Manager and the Project Manager act as the responsible person for the quality, costs, and deadline criteria.**

**2. Determining basic conditions:** Projects come about under different boundary conditions, demands and influencing variables. The PMA should be conducted differently depending on the size of a project. The selection of participants plays an important role in the success; it should include the following groups:

- Project Management people

- Representatives of the designers of the software

Representatives of DevOps professionals  
Representatives of developers of the software  
Representatives of the users

### **3. Designating objective and subjective data:**

Costs, deadlines and qualities count in the objective data in projects. Subjective data involves, e.g., *perceived customer satisfaction, communication between the participants, collaboration in the team, personal advancement of the team members, motivation and commitment, re-defined metrics, surveys, debriefings*,. Several methods are available for the collection of the data. The following methods are used to gain information:

- Interviews: The facilitator prepares a list of questions, such as “What characterizes the work packages that you estimated correctly?”
- Moderated group discussions
- Questionnaires
- KJ sessions: The participants write down up to four positive and negative project experiences on post-it notes. Then they present their issues and put the notes on a whiteboard.

**4. Collection of experience:** The results of the post-mortem analysis are analyzed. The assessment of the findings can result in a strength/weakness profile. This documents the specific advantages of the project course as well as the detectable deficiencies that either lower the efficiency and safety of the project development, or in the worst case, endanger the project goals.

**5. Analysing findings of the Post-mortem session:** The findings are analysed, prioritized and synthesized as lessons learned. In conclusion, the summary of the finding is published and presented in a way that enables future projects to know what processes or tools are important to continue, and to turn problems into improvement activities.

### **EXPERIMENT:**

Paper used: “Software Engineering Projects May Fail Before They Are Started: Post-Mortem Analysis of Five Cancelled Projects, Jarmo J. Ahonen, Paula Savolainen, 2010”

It reports an analysis of five cancelled software engineering projects as thoroughly as possible. . Although there were several unique characteristics in each case, the common features found were the fact that in all 5 cases the outcome of the project was supposed to be delivered to the external customer, and that mistakes were made before the project started, either during the tendering phase or after the supplier got the order but before the project start.

a) **Identify parametric and non parametric tests.** Methodologies used for Post-mortems generally do not involve the use of any statistical tests: parametric and non-parametric. Root cause analysis and cause-effect relationships are studied in post-mortem analysis mainly using diagrams and charts like: Fish-bone diagram, graphics, presentations etc.

b) **Identify Independent, dependent and confounding variables.:** Post-mortems can encompass both quantitative data and qualitative data. Quantitative data can include the variance between the hours estimated for a project and the actual hours incurred. Qualitative data will often include stakeholder satisfaction, end-user satisfaction and perceived quality of

end-deliverables. In the Postmortem Analysis we are using , we can not identify the dependent, independent and confounding variables.

**c) Is it Within-company and cross-company analysis?**

Every project was from a different company and each company was involved in only one case, either as a supplier, a customer, or a company with an in-house development unit. The companies agreed to cede data only for research purposes. It can be said to be a cross- company analysis because it involves 5 companies. Even though the post mortem analysis was conducted separately for different companies, the findings were general.

**d) What is the name of datasets used in a Post-mortem analysis empirical study?:**The PM team analyzed project documentation and interviewed people involved in the project. All types of documents : Administrative documentation, Technical documents, emails, documents that were directly related to the actual cancellation decision, design document, project completion report, SRS were studied and interpreted. During the second phase of data collection, a re-analysis of documentation was performed in order to gain new insights and avoid possible interpretational or reasoning errors. Also data was collected in the form of interviews and KJ sessions from Project Managers, employees and users: perceived customer satisfaction, communication between the participants, collaboration in the team, personal advancement of the team members, motivation and commitment,, debriefings.

**e) What type of datasets (Proprietary and open source software) are used?** Every project was from a different company and each company was involved in only one case, either as a supplier, a customer, or a company with an in-house development unit. Proprietary datasets are used. The companies agreed to cede data only for research purposesAnd the learnings of post-mortem study can be incorporated to improve the performance of other similar upcoming company projects.

**CONCLUSION:**

The post-mortem analysis can be used successfully as an instrument for risk identification and evaluation.

- One result of the PMA is the identification of process errors and failures. The findings are entered in a risk catalog for future projects.
- In addition, the post-mortem analysis also serves to support learning within the organization.
- An efficient risk management system offers the company the opportunity to improve its market position.

**REFERENCES**

- (PDF) *A Review of Small and Large Post-Mortem Analysis Methods*. Available from: [https://www.researchgate.net/publication/250884638\\_A\\_Review\\_of\\_Small\\_and\\_Large\\_Post-Mortem\\_Analysis\\_Methods](https://www.researchgate.net/publication/250884638_A_Review_of_Small_and_Large_Post-Mortem_Analysis_Methods) [accessed Jan 21 2021].
- PAPER USED : <https://ulir.ul.ie/bitstream/handle/10344/659/2010-Ahonen-Software.pdf?sequence=2>

## **EXPERIMENT 2**

### **AIM:**

Defect detection activities like reviews and testing help in identifying the defects in the artifacts (deliverables). These defects must be classified into various buckets before carrying out the root cause analysis. Following are some of the defect categories: Logical, User interface, Maintainability, and Standards.

In the context of the above defect categories, classify the following statements under the defect categories.

### **THEORY:**

RCA (Root Cause Analysis) is a mechanism of analyzing the Defects, to identify its cause. The RCA team brainstorms, reads and digs the defect to identify whether the defect was due to “testing miss”, “development miss” or was a “requirement or design miss”.

### **EXPERIMENT:**

#### **1. Divide by Zero Error is not guarded:**

##### **Logical Defect:**

Logical defects are mistakes done regarding the implementation of the code. They are related to the core of the software and happen when the programmer does not take care of the corner cases or doesn't understand the problem clearly or thinks in a wrong way. Not handling corner cases can lead to low-quality software causing crashes and other kinds of defects.

##### **Poor Test Cases:**

If this defect wasn't unearthed during the testing phase then it can be due to weak test cases. Test cases should be developed for both valid and invalid input conditions. Such test cases also help to evaluate the robustness of the software.

#### **2. Usage of 3.14 in the statement $\text{Circle\_Area} = 3.14 * \text{Radius} * \text{Radius}$ ; Logical Defect:**

Logical defects are mistakes done regarding the implementation of the code. Using 3.14 can lead to loss of precision for some applications. When the programmer doesn't understand the problem clearly or thinks in a wrong way then such types of defects happen. Also while implementing the code if the programmer doesn't take care of the corner cases then logical defects happen. It is basically related to the core of the software.

#### **3. 3500 lines of code in a single function:**

##### **Maintainability:**

Managing large monolithic codebases is tough. And modifying such codebases is tougher. Decomposing a system into subsystems reduces the complexity developers have to deal with by simplifying the parts and increasing their coherence.

#### **4. A pointer is declared but not initialized. It is used in the program for storing a value.**

##### **Logical Defect, Coding errors:**

Logical defects are mistakes done regarding the implementation of the code. Also while implementing the code if the programmer doesn't take care of the corner cases then logical defects happen. Such errors are basically related to the core of the software.

**5. A program designed to handle 1000 simultaneous users, crashed when 1001 the user logged in.**

**Insufficient Testing:**

The above program isn't robust. It is a "testing miss". Before deployment, load testing and stress testing of the software is performed. A load test, by definition, measures the performance of a system under an expected load. A stress test overloads a system in order to find the breaking point.

**Incorrect design:**

The system should have been designed to be robust and reliable. Why on adding only a single user the system crashed? Incorrect design makes software vulnerable to failures.

**Performance Defect:**

Performance defects are the defects when the system or the software application is unable to meet the desired and the expected results. It also includes the response of the system with the varying load on the system.

**6. A "while" loop never exits.**

**Logical Defect:**

Logical defects are related to the core of the software. Logical defects are mistakes done regarding the implementation of the code. These happen due to negligence on part of the developers.

**7. User interface displays "MALFUNCTION 54" when something goes wrong in the back-end.**

**User Interface defect:**

Interface defects means the defects in the interaction of the software and the users. The system may suffer different kinds of interface testing in the forms of the complicated interface, unclear interface or the platform based interface.

**No proper UI testing:**

One of the main aspects of GUI testing is checking whether correct error messages are being displayed. UI Testing is done to uncover such User Interface Bugs. UI testing has become integral to the Software development process, where we analyze an application from the user's point of view.

Since a bug-free interface is a big step forward to a successful product release, listed below are the main aspects of GUI testing checklist:

- Screen with its control of buttons, icons, images, menus is the basic element testers check
- **The correct display of error messages**
- Text check-up: readable font, color, proper text alignment
- Images are checked for clarity, proper alignment
- GUI elements are tested for compliance with different screen resolutions

(Source:<https://www.qamadness.com/common-user-interface-bugs-or-why-ui-testing-matters/>)

**8. No documentation (comments) for the source code.**

**Poor Standards:**



It is a good practice to comment while writing code. It helps other developers to understand your code. Following proper coding standards while software development has many advantages:

- Risk of project failure is reduced
- Easy to Maintain
- Easy Bug Rectification
- Minimal Complexity

## **9. Hungarian Notation is not followed while coding, even though the coding guidelines mandate to use Hungarian Notation.**

### **Logical Defect:**

When the programmer doesn't understand the problem clearly or thinks in a wrong way then such types of defects happen. Logical defects are mistakes done regarding the implementation of the code. These happen due to negligence on part of the developers.

### **Poor Standards:**

It is considered to be a good coding standard to name the functions according to what they perform.(Hungarian Notation). Coding guidelines were not followed. Not compliant with the requirement document. All coding guidelines should be followed for successful software development.

## **10. Pressing the “Tab” key moves the cursor in different fields of a web form randomly.**

### **Interface Defects:**

Interface defects means the defects in the interaction of the software and the users. The system may suffer different kinds of interface testing in the forms of the complicated interface, unclear interface or the platform based interface.

### **No proper UI testing:**

One of the main aspects of GUI testing is checking whether correct error messages are being displayed. UI Testing is done to uncover such User Interface Bugs. UI testing has become integral to the Software development process, where we analyze an application from the user's point of view.

Since a bug-free interface is a big step forward to a successful product release, listed below are the main aspects of GUI testing checklist:

- **Screen with its control of buttons, icons, images, menus** is the basic element testers check for the required position, size, width, character acceptance
- The correct display of error messages
- Text check-up: readable font, color, proper text alignment
- GUI elements are tested for compliance with different screen resolutions

(Source:<https://www.qamadness.com/common-user-interface-bugs-or-why-ui-testing-matters/>)

### **REFERENCES:**

- <https://www.geeksforgeeks.org/types-of-defects-in-software-development/>
- <https://www.softwaretestinghelp.com/root-cause-analysis/>

## **EXPERIMENT 3**

### **AIM:**

Consider any prediction model of your choice.

- a) Analyze the dataset that is given as an input to the prediction model
- b) Find out the quartiles for the used dataset.
- c) Analyze the performance of a model using various performance metrics.

### **THEORY**

The predictive model used for this experiment is from a research paper “**Software Fault Proneness Prediction Using Support Vector Machines by Yogesh Singh, Arvinder Kaur, Ruchika Malhotra**, 2009 [Proceedings of the World Congress on Engineering 2009, London, U.K.] “

**a) Analyze the dataset that is given as an input to the prediction model DATASET**

**USED:**


Public domain KC1 NASA data set (Class level data) of Object Oriented Metrics.

DATASET LINK: <http://promise.site.uottawa.ca/SERepository/datasets/kc1-class-level-numericdefect.arff>

This study makes use of the public domain data set KC1 from the NASA Metrics Data Program . The data in KC1 was collected from a storage management system for receiving/processing ground data, which was implemented in the C++ programming language. This system consists of 145 classes that comprise 2107 methods, with 40K lines of code. KC1 provides both class-level and method-level static metrics. At the class level, values of 10 metrics are computed including seven metrics given by **Chidamber and Kemerer** . The seven OO metrics taken in this study are defined in the Table below.

**METRICS STUDIED**

<b>Metric</b>	<b>Definition</b>
Coupling between Objects (CBO)	CBO for a class is count of the number of other classes to which it is coupled and vice versa.
Lack of Cohesion (LCOM)	For each data field in a class, the percentage of the methods in the class using that data field; the percentages are averaged then subtracted from 100%.
Number of Children (NOC)	The NOC is the number of immediate subclasses of a class in a hierarchy.
Depth of Inheritance (DIT)	The depth of a class within the inheritance hierarchy is the maximum number of steps from the class node to the root of the tree and is measured by the number of ancestor classes.
Weighted Methods per Class (WMC)	A count of methods implemented within a class.
Response for a Class (RFC)	A count of methods implemented within a class plus the number of methods accessible to an object class due to inheritance.
Source Lines Of Code (SLOC)	It counts the lines of code.

 `x.describe()`

	CBO	DIT	LCOM	NOC	RFC	WMC	SLOC
count	2095.000000	2095.000000	2095.000000	2095.000000	2095.000000	2095.000000	2095.000000
mean	0.132220	0.950835	2.843914	2.550358	14.619570	21.375384	20.377566
std	0.704866	3.094650	3.912850	3.386475	24.241343	21.506699	29.838073
min	0.000000	0.000000	1.000000	1.000000	0.000000	0.000000	1.000000
25%	0.000000	0.000000	1.000000	1.000000	0.000000	5.330000	3.000000
50%	0.000000	0.000000	1.000000	1.000000	5.000000	14.450000	9.000000
75%	0.000000	0.000000	3.000000	3.000000	17.000000	29.890000	24.000000
max	12.000000	44.000000	45.000000	45.000000	262.000000	193.060000	288.000000

**Statistical Summary of Dataset b) Find out the quartiles for the used dataset.**

The `quantile()` function of NumPy is used to find quartiles of the dataset for each of the seven OO metrics used.

```
L=['CBO', 'DIT', 'LCOM', 'NOC', 'RFC', 'WMC', 'SLOC']
for feature in L:
    first_quartile = np.quantile(X[feature],0.25)
    second_quartile = np.quantile(X[feature],0.5)
    third_quartile = np.quantile(X[feature],0.75)
    print(" first_quartile for", feature, " is ", first_quartile )
    print(" second_quartile for", feature, " is ", second_quartile )
    print(" third_quartile for", feature, " is ", third_quartile )
    print("\n")
```

Results in Tabular Format:

Feature	First Quartile	Second Quartile	Third Quartile
CBO	0.0	0.2	0.5
DIT	0.0	0.1	0.5
LCOM	1.0	1.0	3.0
NOC	1.0	1.2	3.0
RFC	0.0	5.0	17.0

<b>WMC</b>	<b>5.33</b>	<b>14.45</b>	<b>29.89</b>
<b>SLOC</b>	<b>3.0</b>	<b>9.0</b>	<b>24.0</b>

c) **Analyze the performance of a model using various performance metrics.** The Model evaluation metrics used are:

**Classification Accuracy:** Overall, how often is the classifier correct?

```
[42] # use float to perform true division, not integer division
print((TP + TN) / float(TP + TN + FP + FN))
print(metrics.accuracy_score(y_test, y_pred))

0.8336134453781513
0.8336134453781513
```

**Classification Error:** Overall, how often is the classifier incorrect?

```
classification_error = (FP + FN) / float(TP + TN + FP + FN)
print(classification_error)
print(1 - metrics.accuracy_score(y_test, y_pred))

0.16638655462184873
0.1663865546218487
```

**Sensitivity:** When the actual value is positive, how often is the prediction correct?

Good model aims to maximise Sensitivity..Also known as "True Positive Rate" or "Recall"  
 $TP / TP + FN$

```
sensitivity = TP / float(FN + TP)
print(sensitivity)
print(metrics.recall_score(y_test, y_pred))

0.04854368932038835
0.04854368932038835
```

**Specificity:** When the actual value is negative, how often is the prediction correct? Good model aims to maximise Specificity.

$TN / TN + FP$

```
specificity = TN / (TN + FP)
print(specificity)

0.9979674796747967
```

**False Positive Rate:** When the actual value is negative, how often is the prediction incorrect?

```

false_positive_rate = FP / float(TN + FP)
print(false_positive_rate)
print(1 - specificity)

0.0020325203252032522
0.002032520325203291

```

**Precision:** When a positive value is predicted, how often is the prediction correct?

```

[47] precision = TP / float(TP + FP)

print(precision)
print(metrics.precision_score(y_test, y_pred))

0.8333333333333334
0.8333333333333334

```

**Completeness Score:** It is the Completeness metric of a cluster labeling given a ground truth. A clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster.

```

[48] print(1-completeness_score(y_test, y_pred))

0.8107354402580748

```

Below table ( from paper) summaries the results of evaluating the performance of the model using various metrics:

TABLE 4  
RESULTS OF 10-CROSS VALIDATION OF MODELS

SUPPORT VECTOR MACHINE	
	Model
Cutoff point	0.45
Sensitivity	69.49
Specificity	82.55
Precision	77.24
Completeness	79.59
AUC	0.89

## **CONCLUSION:**

Using the Support Vector Machine Predictive Model for Software Fault Proneness Prediction using OO metrics of KC1 NASA dataset, we analysed the dataset, found out quartiles and analysed the performance of the model using various performance metrics.

## **EXPERIMENT 4**

### **AIM:**

Consider defect dataset and perform following feature reduction techniques using Weka tool. Validate the dataset using 10-cross validation.

- a. Correlation based feature evaluation
- b. Relief Attribute feature evaluation
- c. Information gain feature evaluation
- d. Principle Component

### **THEORY:**

#### **DATASET USED:**

- KC2 Dataset in .arff format.
- Author: Mike Chapman, NASA
- <https://datahub.io/machine-learning/kc2#readme>

One of the NASA Metrics Data Program defect data sets. Data from software for science data processing. Data comes from McCabe and Halstead features extractors of source code. These features were defined in the 70s in an attempt to objectively characterize code features that are associated with software quality.

#### **## Attribute Information**

1. loc :	numeric	%	M McCabe's line count of code
2. v(g) :	numeric	%	M McCabe "cyclomatic complexity"
3. ev(g) :	numeric	%	M McCabe "essential complexity"
4. iv(g) :	numeric	%	M McCabe "design complexity"
5. n :	numeric	%	M Halstead total operators + operands
6. v :	numeric	%	M Halstead "volume"
7. l :	numeric	%	M Halstead "program length"
8. d :	numeric	%	M Halstead "difficulty"
9. i :	numeric	%	M Halstead "intelligence"
10. e :	numeric	%	M Halstead "effort"
11. b :	numeric	%	M Halstead
12. t :	numeric	%	M Halstead's time estimator
13. lOCcode :	numeric	%	M Halstead's line count
14. lOCcomment :	numeric	%	M Halstead's count of lines of comments
15. lOBlank :	numeric	%	M Halstead's count of blank lines
16. lOCcodeAndComment:	numeric		unique operators
17. uniq_Op :	numeric	%	
18. uniq_Opnd :	numeric	%	unique operands
19. total_Op :	numeric	%	total operators
20. total_Opnd :	numeric	%	total operands
21. branchCount :	numeric	%	Branch Count of the flow graph
22. problems :	{ false,true }	%	module has/has not one or more reported defects

## Feature Selection in Weka

A central problem in machine learning is identifying a representative set of features from which to construct a classification model for a particular task.

Many feature selection techniques are supported in Weka.

### **STEPS:**

1. Open the Weka GUI Chooser.
2. Click the “Explorer” button to launch the Explorer.
3. Open the dataset.
4. Click the “Select attributes” tab to access the feature selection methods

Feature selection is divided into two parts: ●

Attribute Evaluator

- Search Method.

Each section has multiple techniques from which to choose. The attribute evaluator is the technique by which each attribute in the dataset is evaluated in the context of the output variable. The search method is the technique by which to try or navigate different combinations of attributes in the dataset in order to arrive on a shortlist of chosen features. Some Attribute Evaluator techniques require the use of specific Search Methods. For example, the **CorrelationAttributeEval** technique used in the next section can only be used with a Ranker Search Method, that evaluates each attribute and lists the results in a rank order.

Both the Attribute Evaluator and Search Method techniques can be configured. Once chosen, click on the name of the technique to get access to its configuration details.

### **a) Correlation Based Feature Selection**

A popular technique for selecting the most relevant attributes in your dataset is to use correlation. Correlation is more formally referred to as **Pearson’s correlation coefficient** in statistics. You can calculate the correlation between each attribute and the output variable and select only those attributes that have a moderate-to-high positive or negative correlation (close to -1 or 1) and drop those attributes with a low correlation (value close to zero).

Weka supports correlation based feature selection with the **CorrelationAttributeEval** technique that requires use of a **Ranker search method**.

### **RESULTS:**

---

Evaluator : weka.attributeSelection.CorrelationAttributeEval  
Search : weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -  
1 Relation: KC2  
Instances: 522  
Attributes: 22  
Evaluation mode: 10-fold cross-validation

---



```
=== Attribute selection 10 fold cross-validation (stratified), seed: 0
```

average merit	average rank	attribute
0.496 +- 0.015	1.4 +- 0.66	17 uniq_Op
0.49 +- 0.016	2 +- 0.77	8 d
0.481 +- 0.03	3 +- 0.77	18 uniq_Opnd
0.473 +- 0.024	3.6 +- 0.66	9 i
0.411 +- 0.032	5.5 +- 0.5	1 loc
0.407 +- 0.016	6.3 +- 2.61	15 lOBlank
0.404 +- 0.034	7 +- 0.45	20 total_Opnd
0.39 +- 0.037	8.4 +- 1.02	19 total_Op
0.39 +- 0.037	8.7 +- 0.46	13 lOCode
0.386 +- 0.035	10 +- 0.45	5 n
0.377 +- 0.028	11 +- 0.45	2 v(g)
0.367 +- 0.028	12.5 +- 0.92	21 branchCount
0.351 +- 0.019	13.1 +- 2.39	14 lOComment
0.358 +- 0.035	13.2 +- 1.17	4 iv(g)
0.339 +- 0.037	14.8 +- 0.6	11 b
0.337 +- 0.04	15.6 +- 1.2	6 v
0.316 +- 0.012	17.7 +- 0.78	7 l
0.315 +- 0.038	18.2 +- 0.98	3 ev(g)
0.306 +- 0.012	18.2 +- 1.17	16 lOCodeAndComment
0.244 +- 0.023	19.9 +- 0.3	10 e
0.223 +- 0.029	20.9 +- 0.3	12 t

From Classification scores, a ranked list of features is obtained. Experiments with choosing a select number of the highest ranked features and using them with common machine learning algorithms showed that, on average, the top three or more features are as accurate as using the original set. Each Feature's weight reflects its ability to distinguish among the class values. Features are ranked by weight and those exceed a user-specified threshold are included in the final

subset of features. Running this on the KC2 dataset suggests that the attribute (**uniq\_Op**) has the highest correlation with the output class closely followed by the attribute (**d**).

### **c) Relief Attribute feature evaluation**

Weka supports correlation based feature selection with **ReliefFAttributeEval**, the technique that requires use of a **Ranker search method**.

From Classification scores, a ranked list of features is obtained. Each Feature's weight reflects its ability to distinguish among the class values. Features are ranked by weight and those exceed a user- specified threshold are included in the final subset of features.

#### **ReliefFAttributeEval :**

Evaluates the worth of an attribute by repeatedly sampling an instance and considering the value of the given attribute for the nearest instance of the same and different class.

#### **RESULTS:**



---

Evaluator: weka.attributeSelection.ReliefFAttributeEval -M -I -D 1 -K 10 Search:  
weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -  
1 Relation: KC2  
Instances: 522

---

```
=== Attribute selection 10 fold cross-validation (stratified), seed: 0

average merit      average rank  attribute
0.05 +- 0.005      1 +- 0        7 l
0.021 +- 0.003     2 +- 0        17 uniq_Op
0.011 +- 0.004     3.1 +- 0.3    9 i
0.009 +- 0.001     4 +- 0.45    8 d
0.008 +- 0.004     5.3 +- 0.64   18 uniq_Opnd
0.006 +- 0.001     6.8 +- 3.76   16 lOCodeAndComment
0.004 +- 0.001     8.4 +- 1.74   15 lOBlank
0.004 +- 0.001     8.5 +- 3.53   14 lOComment
0.003 +- 0         9.6 +- 1.85   11 b
0.003 +- 0.001    9.7 +- 1.42   2 v(g)
0.003 +- 0.001   10.8 +- 1.25   20 total_Opnd
0.003 +- 0.001   11.1 +- 1.45   21 branchCount
0.003 +- 0.001   12.9 +- 1.37   1 loc
0.003 +- 0.001   13.6 +- 1.02   19 total_Op
0.002 +- 0.001   14.6 +- 1.11   5 n
0.002 +- 0.001   15.9 +- 2.02   13 lOCode
0.002 +- 0.001   16.1 +- 0.94   4 iv(g)
0.002 +- 0.001   17.8 +- 0.6    6 v
0.001 +- 0.001   19.3 +- 1      3 ev(g)
0.001 +- 0       19.6 +- 0.49   10 e
0.001 +- 0       20.9 +- 0.3    12 t
```

### **c) Information gain feature evaluation**

Another popular feature selection technique is to calculate the information gain.

You can calculate the information gain (also called **entropy**) for each attribute for the output variable. Entry values vary from 0 (no information) to 1 (maximum information). Those attributes that contribute more information will have a higher information gain value and can be selected, whereas those that do not add much information will have a lower score and can be removed. Weka supports feature selection via information gain using the **InfoGainAttributeEval** Attribute Evaluator. Like the correlation technique above, the **Ranker Search Method** must be used.

### **RESULTS:**

---

Evaluator: weka.attributeSelection.InfoGainAttributeEval

Search: weka.attributeSelection.Ranker -T -1.7976931348623157E308

-N -1 Relation: KC2

Instances: 522

Attributes: 22

---

```
=== Attribute selection 10 fold cross-validation (stratified), seed: 0
```

average merit	average rank	attribute
0.23 +- 0.016	1.7 +- 0.64	20 total_Opnd
0.228 +- 0.015	1.8 +- 1.47	18 uniq_Opnd
0.215 +- 0.012	3.5 +- 0.92	1 loc
0.21 +- 0.012	5.1 +- 2.47	11 b
0.212 +- 0.015	5.1 +- 2.17	19 total_Op
0.209 +- 0.011	5.5 +- 1.2	6 v
0.207 +- 0.011	6.3 +- 1	5 n
0.198 +- 0.017	9.6 +- 2.33	12 t
0.198 +- 0.014	9.9 +- 3.21	9 i
0.198 +- 0.017	10 +- 2.49	10 e
0.192 +- 0.018	11 +- 2.24	13 lOCode
0.187 +- 0.016	12.1 +- 2.3	2 v(g)
0.185 +- 0.012	13 +- 1.34	15 lOBlank
0.185 +- 0.013	13 +- 1.61	8 d
0.175 +- 0.013	15.3 +- 1.35	7 l
0.175 +- 0.013	15.6 +- 1.62	4 iv(g)
0.171 +- 0.015	16 +- 2.49	17 uniq_Op
0.17 +- 0.017	16.5 +- 1.86	21 branchCount
0.141 +- 0.009	19 +- 0	3 ev(g)
0.126 +- 0.012	20 +- 0	14 lOComment
0.051 +- 0.006	21 +- 0	16 lOCodeAndComment

#### **d) Principle Component**

Weka Explorer can be used to perform principal components analysis and transformation of the data. It is used in conjunction with a Ranker search.

#### **RESULTS:**

---

Evaluator: weka.attributeSelection.PrincipalComponents -R 0.95 -A 5

Search: weka.attributeSelection.Ranker -T -1.7976931348623157E308

-N -1 Relation: KC2

Instances: 522

Attributes: 22

Evaluation mode: evaluate on all training

data Search Method: Attribute ranking.

Attribute Evaluator (unsupervised): Principal Components Attribute Transformer

---

```
Ranked attributes:
0.1912  1  -0.241loc-0.241n-0.241lOCcode-0.241total_Op-0.241total_Opnd...
0.1066  2  0.586l-0.505uniq_Op-0.354d+0.259t+0.229e...
0.0725  3  -0.782lOCcodeAndComment-0.414lOCcomment-0.323l+0.147i-0.145d...
0.0502  4  -0.599lOCcomment+0.439lOCcodeAndComment-0.416l-0.318lOBlank+0.228ev(g) ...
0.0334  5  0.58 lOCcomment-0.443i-0.352l-0.278lOBlank-0.26lOCcodeAndComment...

Selected attributes: 1,2,3,4,5 : 5
```

## REFERENCES:

- <https://www.cs.waikato.ac.nz/~mhall/thesis.pdf>
- <https://machinelearningmastery.com/perform-feature-selection-machine-learning-data- weka/#:~:text=Weka%20supports%20correlation%20based%20feature,correlation%20with%20the%20output%20class.>

## **EXPERIMENT 5**

Online loan system has two modules for the two basic services, namely **Car loan service** and **House loan service**. The two modules have been named as **Car\_Loan\_Module** and **House\_Loan\_Module**. **Car\_Loan\_Module** has 2000 lines of uncommented source code. **House\_Loan\_Module** has 3000 lines of uncommented source code. **Car\_Loan\_Module** was completely implemented by Mike.

**House\_Loan\_Module** was completely implemented by John. Mike took 100 person hours to implement **Car\_Loan\_Module**. John took 200 person hours to implement **House\_Loan\_Module**. Mike's module had 5 defects. John's module had 6 defects. With respect to the context given, which among the following is an INCORRECT statement? Identify the null and alternate hypothesis for the following options.

Justify and Choose one:

- a. John's Quality is better than Mike's Quality
- b. John's Productivity is more than Mike's Productivity
- c. John introduced more defects than Mike
- d. John's Effort is more than Mike's Effort.

### **THEORY**

Hypothesis is an educated guess about something in the world around you. It should be testable, either by experiment or observation.

Hypothesis testing in statistics is a way for you to test the results of a survey or experiment to see if you have meaningful results. You're basically testing whether your results are valid by figuring out the odds that your results have happened by chance. If your results may have happened by chance, the experiment won't be repeatable and so has little use.

Hypothesis testing can be one of the most confusing aspects for students, mostly because before you can even perform a test, you have to know what your null hypothesis is. Often, those tricky word problems that you are faced with can be difficult to decipher. But it's easier than you think; all you need to do is:

1. Figure out your null hypothesis,
2. State your null hypothesis,
3. Choose what kind of test you need to perform,
4. Either support or reject the null hypothesis.

#### **• NULL hypothesis**

The null hypothesis states that a population parameter (such as the mean, the standard deviation, and so on) is equal to a hypothesized value. The null hypothesis is often an initial claim that is based on previous analyses or specialized knowledge.

- **Alternate hypothesis**

The alternative hypothesis states that a population parameter is smaller, greater, or different than the hypothesized value in the null hypothesis. The alternative hypothesis is what you might believe to be true or hope to prove true.

**EXPERIMENT:**

The **answer** is **b)** only option b is wrong among the given options

a)

Let John's quality be  $J_q$

Let Mike's Quality be  $M_q$

Let John's defects per line be  $J_{dl}$

Let Mike's defects per line be  $M_{dl}$

According to question quality is measured in terms on the number of bugs per line of code

$H_o = J_q > M_q \sim J_{dl} < M_{dl}$

$H_a = J_q \leq M_q \sim J_{dl} \geq M_{dl}$

Now, as  $J_{dl} = 0.002$

and  $M_{dl} = 0.0025$

$J_{dl} < M_{dl}$ , Hence the null hypothesis is correct

b)

Let John's Productivity be  $J_p$

Let Mike's Productivity be  $M_p$

$H_o = J_p > M_p$

$H_a = J_p \leq$

$M_p$  Now,  $J_p =$

$15$   $M_p = 20$

$J_p \leq M_p$ , Hence the null hypothesis is wrong

c)

Let John's defects be  $J_d$

Let Mike's defects be  $M_d$

$H_o = J_d > M_d$

$H_a = J_d \leq$

$M_d$

Now,  $J_d =$

$6$   $M_d = 5$

$J_d > M_d$  , Hence the null hypothesis is correct

d) Let John's effort be  $J_e$

Let Mike's effort be  $M_e$

$H_0 = J_e > M_e$

$H_a = J_e \leq$

$M_e$

Now,  $J_e = 200$

$M_e = 100$

$J_e > M_e$  , Hence the null hypothesis is correct

**CONCLUSION:**

Through this experiment we were able to learn about Null hypothesis, Alternative hypothesis and Hypothesis Testing.

## **EXPERIMENT 6**

### **AIM:**

Statistical Hypothesis Testing in R-Statisticians use hypothesis testing to formally check whether the hypothesis is accepted or rejected. Consider an example or data of your choice and identify the following:

- (A) State the Hypotheses
- (B) Formulate an Analysis Plan
- (C) Analyze Sample Data
- (D) Interpret Results
- (E) Estimate type-I and type-II error

### **THEORY**

1. NULL HYPOTHESIS: A **null hypothesis** is a type of hypothesis used in statistics that proposes that no statistical significance exists in a set of given observations.

2. HYPOTHESIS TESTING: All hypothesis tests are conducted the same way. The researcher states a hypothesis to be tested, formulates an analysis plan, analyzes sample data according to the plan, and accepts or rejects the null hypothesis, based on results of the analysis.

### **PROCEDURE**

**State the hypotheses.** Every hypothesis test requires the analyst to state a null hypothesis and an alternative hypothesis. The hypotheses are stated in such a way that they are mutually exclusive. That is, if one is true, the other must be false; and vice versa.

**Formulate an analysis plan.** The analysis plan describes how to use sample data to accept or reject the null hypothesis. It should specify the following elements.

**Significance level.** Often, researchers choose significance levels equal to 0.01, 0.05, or 0.10; but any value between 0 and 1 can be used.

**Test method.** Typically, the test method involves a test statistic and a sampling distribution. Computed from sample data, the test statistic might be a mean score, proportion, difference between means, difference between proportions, z-score, t statistic, chi-square, etc. Given a test statistic and its sampling distribution, a researcher can assess probabilities associated with the test statistic. If the test statistic probability is less than the significance level, the null hypothesis is rejected.

**Analyze sample data.** Using sample data, perform computations called for in the analysis plan.

**Test statistic.** When the null hypothesis involves a mean or proportion, use either of the following equations to compute the test statistic.

Test statistic = (Statistic - Parameter) / (Standard deviation of statistic) Test

statistic = (Statistic - Parameter) / (Standard error of statistic)

where *Parameter* is the value appearing in the null hypothesis, and *Statistic* is the point estimate of *Parameter*. As part of the analysis, you may need to compute the standard deviation or standard error of the statistic. Previously, we presented common formulas for the standard deviation and standard error. When the parameter in the null hypothesis involves categorical data, you may use a chi-square statistic as the test statistic. Instructions for computing a chi square test statistics are presented in the lesson on the chi-square goodness of fit test.

P-value. The P-value is the probability of observing a sample statistic as extreme as the test statistic, assuming the null hypothesis is true.

**Interpret the results.** If the sample findings are unlikely, given the null hypothesis, the researcher rejects the null hypothesis. Typically, this involves comparing the P-value to the significance level, and rejecting the null hypothesis when the P-value is less than the significance level.

**Type I and Type II errors** can be defined in terms of hypothesis testing. A **Type I error** is the probability of rejecting a true null hypothesis. A **Type II error** is the probability of failing to reject a false null hypothesis.

**Problem Taken:** Within a school district, students were randomly assigned to one of two Math teachers - Mrs. Smith and Mrs. Jones. After the assignment, Mrs. Smith had 30 students, and Mrs. Jones had 25 students.

At the end of the year, each class took the same standardized test. Mrs. Smith's students had an average test score of 78, with a standard deviation of 10; and Mrs. Jones' students had an average test score of 85, with a standard deviation of 15.

Test the hypothesis that Mrs. Smith and Mrs. Jones are equally effective teachers. Use a 0.10 level of significance. (Assume that student performance is approximately normal.)

### **OUTPUT / ANSWERS**

**State the hypotheses.** The first step is to state the null hypothesis and an alternative hypothesis.

Null hypothesis:  $\mu_1 - \mu_2 = 0$

Alternative hypothesis:  $\mu_1 - \mu_2 \neq 0$

Note that these hypotheses constitute a two-tailed test. The null hypothesis will be rejected if the difference between sample means is too big or if it is too small.



**Formulate an analysis plan.** For this analysis, the significance level is 0.10. Using sample data, we will conduct a two-sample t-test of the null hypothesis.

**Analyze sample data.** Using sample data, we compute the standard error (SE), degrees of freedom (DF), and the t statistic test statistic (t).

$$SE = \sqrt{(s_1^2/n_1) + (s_2^2/n_2)}$$

$$SE = \sqrt{(10^2/30) + (15^2/25)} = \sqrt{3.33 + 9}$$

$$SE = \sqrt{12.33} = 3.51$$

$$DF = (s_1^2/n_1 + s_2^2/n_2) / \{ [ (s_1^2/n_1) / (n_1 - 1) ] + [ (s_2^2/n_2) / (n_2 - 1) ] \}$$

$$DF = (10^2/30 + 15^2/25) / \{ [ (10^2/30) / (29) ] + [ (15^2/25) / (24) ] \}$$

$$DF = (3.33 + 9) / \{ [ (3.33) / (29) ] + [ (9) / (24) ] \} = 12.33 / (0.382 + 3.75) = 12.33 / 4.132 = 2.98 \approx 3$$

$$t = [ (x_1 - x_2) - d ] / SE = [ (78 - 85) - 0 ] / 3.51 = -7 / 3.51 = -1.99$$

where  $s_1$  is the standard deviation of sample 1,  $s_2$  is the standard deviation of sample 2,  $n_1$  is the size of sample 1,  $n_2$  is the size of sample 2,  $x_1$  is the mean of sample 1,  $x_2$  is the mean of sample 2,  $d$  is the hypothesized difference between the population means, and SE is the standard error.

Since we have a two-tailed test, the P-value is the probability that a t statistic having 40 degrees of freedom is more extreme than -1.99; that is, less than -1.99 or greater than 1.99.

We use the t Distribution Calculator to find  $P(t < -1.99) = 0.027$ , and  $P(t > 1.99) = 0.027$ . Thus, the P-value =  $0.027 + 0.027 = 0.054$ .

**Interpret results.** Since the P-value (0.054) is less than the significance level (0.10), we cannot accept the null hypothesis.

**Type 1 error.** Significance level in this case is 0.1

Specifically, the approach is appropriate because the sampling method was simple random sampling, the samples were independent, the sample size was much smaller than the population size, and the samples were drawn from a normal population.

### **LEARNING**

This experiment gives insights into the use of hypothetical testing on real-life examples.

## EXPERIMENT 7

### AIM:

Consider the defect dataset and implement the following statistical test using the SPSS tool.

- a. t-test
- b. Chi-Square Test
- c. Wilcoxon Signed Test
- d. Friedman Test
- e. Kruskal Wallis Test

### THEORY

**a) T-test:** A t-test is a type of inferential statistic used to determine if there is a significant difference between the means of two groups, which may be related in certain features. The t-test is one of many tests used for the purpose of hypothesis testing in statistics.

$$t = \frac{m - \mu}{s / \sqrt{n}}$$

$t$  = Student's t-test

$m$  = mean

$\mu$  = theoretical value

$s$  = standard deviation

$n$  = variable set size

**b) Chi-Square Test:** A chi-squared test, also written as  $\chi^2$  test, is a statistical hypothesis test that is valid to perform when the test statistic is chi-squared distributed under the null hypothesis, specifically Pearson's chi-squared test and variants thereof.

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

$\chi^2$  = chi squared

$O_i$  = observed value

$E_i$  = expected value

**c) Wilcoxon Signed:** The Wilcoxon signed-rank test is a non-parametric statistical hypothesis test used to compare two related samples, matched samples, or repeated measurements on a single sample to assess whether their population mean ranks differ.

$$W = \sum_{i=1}^{N_r} [\text{sgn}(x_{2,i} - x_{1,i}) \cdot R_i]$$

$W$  = test statistic

$N_r$  = sample size, excluding pairs where  $x_1 = x_2$

$\text{sgn}$  = sign function

$x_{1,i}, x_{2,i}$  = corresponding ranked pairs from two distributions

$R_i$  = rank  $i$

**d) Friedman Test:** The Friedman test is the non-parametric alternative to the one-way ANOVA with repeated measures. It is used to test for differences between groups when the dependent variable being measured is ordinal. It can also be used for continuous data that has violated the assumptions necessary to run the one-way ANOVA with repeated measures (e.g., data that has marked deviations from normality).

**e) Kruskal-Wallis:** The Kruskal–Wallis test by ranks, Kruskal–Wallis H test, or one-way ANOVA on ranks is a non-parametric method for testing whether samples originate from the same distribution. It is used for comparing two or more independent samples of equal or different sample sizes.

Dataset: Software Defect prediction dataset is used to evaluate these tests.

Features	Description
CBO	Coupling between objects. Counts the number of dependencies a class has.
WMC	Weight Method Class or McCabe's complexity. It counts the number of branch instructions in a class.
DIT	Depth Inheritance Tree. It counts the number of "fathers" a class has. All classes have DIT at least 1 (everyone inherits java.lang.Object).
rfc	Response for a Class. Counts the number of unique method invocations in a class.
lcom	Lack of Cohesion of Methods. Calculates LCOM metric.
totalMethods	Counts the number of methods.

totalFields	Counts the number of fields.
NOSI	Number of static invocations. Counts the number of invocations to static methods.
LOC	Lines of code. It counts the lines of count, ignoring empty lines.
returnQty	Quantity of returns. The number of return instructions.
loopQty	Quantity of loops. The number of loops (i.e., for, while, do while, enhanced for).
comparisonsQty	Quantity of comparisons. The number of comparisons (i.e., == and !=).
tryCatchQty	Quantity of try/catches. The number of try/catches.
parenthesizedExpsQty	Quantity of parenthesized expressions. The number of expressions inside parenthesis.
stringLiteralsQty	String literals. The number of string literals (e.g., "John Doe").
numbersQty	Quantity of Number. The number of numbers (i.e., int, long, double, float) literals.
assignmentsQty	Quantity of Variables. Number of declared variables.
mathOperationsQty	Quantity of Math Operations: The number of math operations (times, divide, remainder, plus, minus, left shift, right shift).
variablesQty	Quantity of Variables. Number of declared variables.
maxNestedBlocks	Max nested blocks. The highest number of blocks nested together.
uniqueWordsQty	Number of unique words. Number of unique words in the source code.

## OUTPUT

Paired T-Test performed between CBO & totalMethods

T-Test

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	cbo	27.49	6052	33.215	.427
	totalMethods	33.50	6052	53.557	.688

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	cbo & totalMethods	6052	.545	.000

Paired Samples Test

Paired Differences

		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference		t	df	Sig. (2-tailed)
					Lower	Upper			
Pair 1	cbo - totalMethods	-6.004	45.095	.580	-7.140	-4.868	-10.358	6051	<.001

Paired Samples Effect Sizes

		Standardized	Point Estimate	95% Confidence Interval	
				Lower	Upper
Pair 1	cbo - totalMethods	Cohen's d	45.095	-.133	-.108
		Hedges' correction	45.098	-.133	-.108

Independent Sample-T test performed with cbo, wmc, dit, rfc.

T-Test

Group Statistics

	totalFields	N	Mean	Std. Deviation	Std. Error Mean
cbo	1	494	14.26	18.061	.813
	2	410	15.53	19.816	.979
wmc	1	494	38.88	58.346	2.625
	2	410	34.85	59.021	2.915
dit	1	494	2.16	4.064	.183
	2	410	2.25	3.958	.195
rfc	1	494	30.86	36.929	1.662
	2	410	32.11	42.536	2.101

Independent Samples Test

Levene's Test for Equality of Variances

		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
cbo	Equal variances assumed	2.001	.158	-1.012	902	.312	-1.277	1.261	-3.752	1.198
	Equal variances not assumed			-1.004	837.242	.316	-1.277	1.272	-3.773	1.220
wmc	Equal variances assumed	4.695	.031	1.027	902	.305	4.023	3.918	-3.668	11.713
	Equal variances not assumed			1.026	867.875	.305	4.023	3.923	-3.676	11.722
dit	Equal variances assumed	.472	.492	-.342	902	.733	-.092	.268	-.618	.435
	Equal variances not assumed			-.343	879.321	.732	-.092	.268	-.617	.434
rfc	Equal variances assumed	.508	.476	-.473	902	.636	-1.251	2.644	-6.439	3.937
	Equal variances not assumed			-.467	815.896	.641	-1.251	2.678	-6.508	4.006

Independent Samples Effect Sizes					
		Standardizer <sup>a</sup>	Point Estimate	95% Confidence Interval	
				Lower	Upper
cbo	Cohen's d	18.877	-.068	-.199	.063
	Hedges' correction	18.893	-.068	-.198	.063
	Glass's delta	19.816	-.064	-.195	.067
wmc	Cohen's d	58.653	.069	-.062	.200
	Hedges' correction	58.702	.069	-.062	.199
	Glass's delta	59.021	.068	-.063	.199
dit	Cohen's d	4.016	-.023	-.154	.108
	Hedges' correction	4.019	-.023	-.154	.108
	Glass's delta	3.958	-.023	-.154	.108
rfc	Cohen's d	39.570	-.032	-.163	.099
	Hedges' correction	39.603	-.032	-.162	.099
	Glass's delta	42.536	-.029	-.160	.102
a. The denominator used in estimating the effect sizes. Cohen's d uses the pooled standard deviation. Hedges' correction uses the pooled standard deviation, plus a correction factor. Glass's delta uses the sample standard deviation of the control group.					

## Chi-Square Test

NPar Tests								
Descriptive Statistics								
	N	Mean	Std. Deviation	Minimum	Maximum	25th	Percentiles 50th (Median)	75th
cbo	6052	27.49	33.215	0	419	9.00	18.00	34.00
wmc	6052	86.18	136.077	0	1714	18.00	45.00	100.00
dit	6052	4.60	9.288	1	285	1.00	2.00	4.00
rfc	6052	68.95	85.772	0	1203	18.00	44.00	89.00

# Chi-Square Test

## Frequencies

cbo			
	Observed N	Expected N	Residual
0	23	30.7	-7.7
1	95	30.7	64.3
2	130	30.7	99.3
3	161	30.7	130.3
4	181	30.7	150.3
5	219	30.7	188.3
6	196	30.7	165.3
7	220	30.7	189.3
8	185	30.7	154.3
9	260	30.7	229.3
10	191	30.7	160.3
11	166	30.7	135.3
12	145	30.7	114.3
13	177	30.7	146.3
14	143	30.7	112.3
15	134	30.7	103.3
16	160	30.7	129.3
17	163	30.7	132.3
18	149	30.7	118.3
19	142	30.7	111.3
20	107	30.7	76.3

dit			
	Observed N	Expected N	Residual
1	2245	89.0	2156.0
2	1386	89.0	1297.0
3	557	89.0	468.0
4	371	89.0	282.0
5	240	89.0	151.0
6	190	89.0	101.0
7	149	89.0	60.0
8	126	89.0	37.0
9	99	89.0	10.0
10	97	89.0	8.0
11	73	89.0	-16.0
12	76	89.0	-13.0
13	58	89.0	-31.0
14	40	89.0	-49.0
15	24	89.0	-65.0
16	20	89.0	-69.0
17	30	89.0	-59.0
18	17	89.0	-72.0
19	23	89.0	-66.0
20	17	89.0	-72.0

rfc			
	Observed N	Expected N	Residual
0	147	15.8	131.2
1	35	15.8	19.2
2	96	15.8	80.2
3	55	15.8	39.2
4	87	15.8	71.2
5	63	15.8	47.2
6	66	15.8	50.2
7	84	15.8	68.2
8	74	15.8	58.2
9	90	15.8	74.2
10	96	15.8	80.2
11	107	15.8	91.2
12	93	15.8	77.2
13	72	15.8	56.2
14	83	15.8	67.2
15	73	15.8	57.2
16	97	15.8	81.2
17	73	15.8	57.2
18	77	15.8	61.2
19	63	15.8	47.2
20	66	15.8	50.2

wmc			
	Observed N	Expected N	Residual
0	16	14.1	1.9
1	54	14.1	39.9
2	44	14.1	29.9
3	54	14.1	39.9
4	92	14.1	77.9
5	77	14.1	62.9
6	119	14.1	104.9
7	104	14.1	89.9
8	99	14.1	84.9
9	113	14.1	98.9
10	95	14.1	80.9
11	113	14.1	98.9
12	100	14.1	85.9
13	84	14.1	69.9
14	98	14.1	83.9
15	77	14.1	62.9
16	86	14.1	71.9
17	70	14.1	55.9
18	73	14.1	58.9
19	76	14.1	61.9
20	71	14.1	56.9

Test Statistics				
	cbo	wmc	dit	rfc
Chi-Square	18016.244 <sup>a</sup>	14453.945 <sup>b</sup>	79129.146 <sup>c</sup>	12848.642 <sup>d</sup>
df	196	428	67	383
Asymp. Sig.	.000	.000	.000	.000
a. 0 cells (0.0%) have expected frequencies less than 5. The minimum expected cell frequency is 30.7. b. 0 cells (0.0%) have expected frequencies less than 5. The minimum expected cell frequency is 14.1. c. 0 cells (0.0%) have expected frequencies less than 5. The minimum expected cell frequency is 89.0. d. 0 cells (0.0%) have expected frequencies less than 5. The minimum expected cell frequency is 15.8.				

## Wilcoxin Signed Test

### NPar Tests

### Wilcoxon Signed Ranks Test

Ranks				
		N	Mean Rank	Sum of Ranks
totalMethods - cbo	Negative Ranks	3026 <sup>a</sup>	2661.35	8053241.00
	Positive Ranks	2796 <sup>b</sup>	3182.23	8897512.00
	Ties	230 <sup>c</sup>		
	Total	6052		
dit - wmc	Negative Ranks	5954 <sup>d</sup>	3015.31	17953128.5
	Positive Ranks	40 <sup>e</sup>	347.16	13886.50
	Ties	58 <sup>f</sup>		
	Total	6052		
lcom - rfc	Negative Ranks	2864 <sup>g</sup>	2036.71	5833127.50
	Positive Ranks	3104 <sup>h</sup>	3859.01	11978368.5
	Ties	84 <sup>i</sup>		
	Total	6052		

- a. totalMethods < cbo
- b. totalMethods > cbo
- c. totalMethods = cbo
- d. dit < wmc
- e. dit > wmc
- f. dit = wmc
- g. lcom < rfc
- h. lcom > rfc
- i. lcom = rfc

Test Statistics <sup>a</sup>			
	totalMethods - cbo	dit - wmc	lcom - rfc
Z	-3.292 <sup>b</sup>	-66.949 <sup>c</sup>	-23.084 <sup>b</sup>
Asymp. Sig. (2-tailed)	<.001	.000	<.001

- a. Wilcoxon Signed Ranks Test
- b. Based on negative ranks.
- c. Based on positive ranks.



## Friedman Test

### NPar Tests

#### Friedman Test

##### Ranks

	Mean Rank
cbo	13.90
wmc	18.14
dit	6.98
rfc	17.45
lcom	15.52
totalMethods	13.91
totalFields	9.80
nosi	4.73
loc	21.62
returnQty	11.69
loopQty	5.80
comparisonsQty	9.76
tryCatchQty	4.50
parenthesizedExpsQty	5.00
stringLiteralsQty	11.82
numbersQty	10.40
assignmentsQty	16.64
mathOperationsQty	8.00
variablesQty	15.37
maxNestedBlocks	7.48
uniqueWordsQty	20.79
defect	3.68

##### Test Statistics<sup>a</sup>

N	6052
Chi-Square	91337.006
df	21
Asymp. Sig.	.000

a. Friedman Test

## Kruskal Wallis Test

## NPar Tests

### Kruskal-Wallis Test

#### Ranks

	totalFields	N	Mean Rank
cbo	1	494	1655.32
	2	410	1734.75
	3	411	1697.18
	4	417	2221.85
	5	356	2060.77
	6	268	2309.46
	7	258	2345.04
	8	212	2564.77
	9	170	2661.93
	10	182	2846.79
	11	143	2889.40
	12	145	2907.52
	13	147	3161.90
	14	120	3088.93
	15	105	3280.45
	16	94	3533.00
	17	97	3692.43
	18	117	3536.93
	19	69	3605.61
	20	88	3439.35

dit	1	494	1849.87
	2	410	1862.83
	3	411	1821.44
	4	417	2097.34
	5	356	2097.12
	6	268	2414.12
	7	258	2397.76
	8	212	2535.19
	9	170	2544.13
	10	182	3219.52
	11	143	2659.31
	12	145	2943.32
	13	147	2984.28
	14	120	3226.70
	15	105	3224.72
	16	94	3441.59
	17	97	3583.77
	18	117	3371.88
	19	69	3335.65
	20	88	3070.97

wmc	1	494	1588.96
	2	410	1503.97
	3	411	1535.29
	4	417	1995.41
	5	356	2081.31
	6	268	2128.48
	7	258	2324.39
	8	212	2548.22
	9	170	2569.37
	10	182	2715.01
	11	143	2915.40
	12	145	3057.42
	13	147	3310.17
	14	120	3169.70
	15	105	3209.66
	16	94	3537.80
	17	97	3655.73
	18	117	3737.84
	19	69	3670.29
	20	88	3858.55

rfc	1	494	1552.31
	2	410	1571.10
	3	411	1636.04
	4	417	2189.11
	5	356	2091.46
	6	268	2135.43
	7	258	2427.61
	8	212	2507.00
	9	170	2576.94
	10	182	2752.69
	11	143	2900.47
	12	145	2968.47
	13	147	3315.47
	14	120	3234.03
	15	105	3173.82
	16	94	3557.42
	17	97	3640.56
	18	117	3719.88
	19	69	3677.68
	20	88	3658.43

Test Statistics <sup>a,b</sup>				
	cbo	wmc	dit	rfc
Kruskal-Wallis H	1932.136	2483.392	1772.069	2253.392
df	110	110	110	110
Asymp. Sig.	.000	.000	<.001	.000

a. Kruskal Wallis Test

b. Grouping Variable: totalFields

## **CONCLUSION**

Through this experiment, we learned how to perform hypothesis tests such as parametric test and non-parametric test which includes Kruskal Wallis, paired t-test, chi-square test, Wilcoxon signed rank test. We also explored the SPSS tool (by IBM) to evaluate these tests on Software defect prediction dataset.

## **EXPERIMENT 8**

### **AIM:**

Why is version control important? How many types of version control systems are there? Demonstrate how version control is used in a proper sequence (stepwise).

### **THEORY**

First, what is version control?

In software engineering, version control (also known as revision control, source control, or source code management) is a class of systems responsible for managing changes to computer programs, documents, large web sites, or other collections of information. Version control is a component of software configuration management.

This includes version control software, version control systems, or version control tools. Version control is a component of software configuration management. It's sometimes referred to as VCS programming.

As mentioned above, version control is sometimes referred to as revision control or source control. It's an important component of software configuration management.

Why is version control important?

Version control is important to keep track of changes — and keep every team member working off the latest version. You should use version control software for all code, files, and assets that multiple team members will collaborate on.

It needs to do more than just manage and track files. It should help you develop and ship products faster. This is especially important for teams practicing DevOps.

That's because using the right one:

- Improves visibility.
- Helps teams collaborate around the world.
- Accelerates product delivery. When you work on a development team, you may be touching similar parts of the code throughout a project. As a result, changes made in one part of the source can be incompatible with those made by another developer working at the same time.

Version control helps solve these kinds of problems and provides:

- A complete history of every file, which enables you to go back to previous versions to analyze the source of bugs and fix problems in older versions.
- The ability to work on independent streams of changes, which allow you to merge that work back together and verify that your changes conflict.
- The ability to trace each change with a message describing the purpose and intent of the change and connect it to project management and bug tracking software.

How many types of version control systems are there?

There are three types of version control: centralized and distributed.

### **Local Version Control Systems**

It is one of the simplest forms and has a database that keeps all the changes to files under revision control. RCS is one of the most common VCS tools. It keeps patch sets (differences between files) in a special format on disk. By adding up all the patches it can then re-create what any file looked like at any point in time.

### **Centralized version control**

With centralized version control systems, you have a single “central” copy of your project on a server and commit your changes to this central copy. You pull the files that you need, but you never have a full copy of your project locally. Some of the most common version control systems are centralized, including Subversion (SVN) and Perforce.

### **Distributed version control**

With distributed version control systems (DVCS), you don't rely on a central server to store all the versions of a project's files. Instead, you clone a copy of a repository locally so that you have the full history of the project. Two common distributed version control systems are Git and Mercurial. While you don't have to have a central repository for your files, you may want one "central" place to keep your code so that you can share and collaborate on your project with others. That's where Bitbucket comes in. Keep a copy of your code in a repository on Bitbucket so that you and your teammates can use Git or Mercurial locally and to push and pull code.

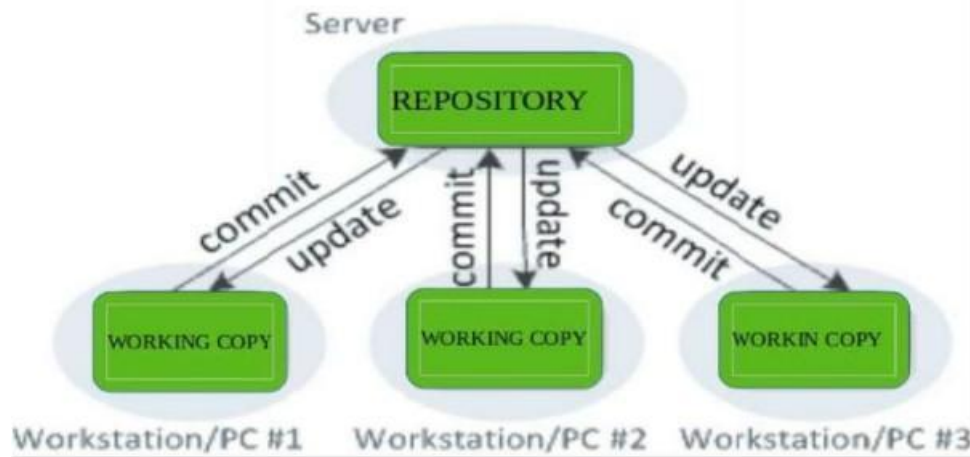
Here are a few of the most popular types of VCS:

- Helix Core (Perforce)
- Git
- SVN (Subversion)
- ClearCase
- Mercurial
- TFS (Team Foundation Server)

Version control System stepwise proper sequence.

1. Proper stepwise sequence to use VCS in Centralized Version Control System.

## Centralized version control

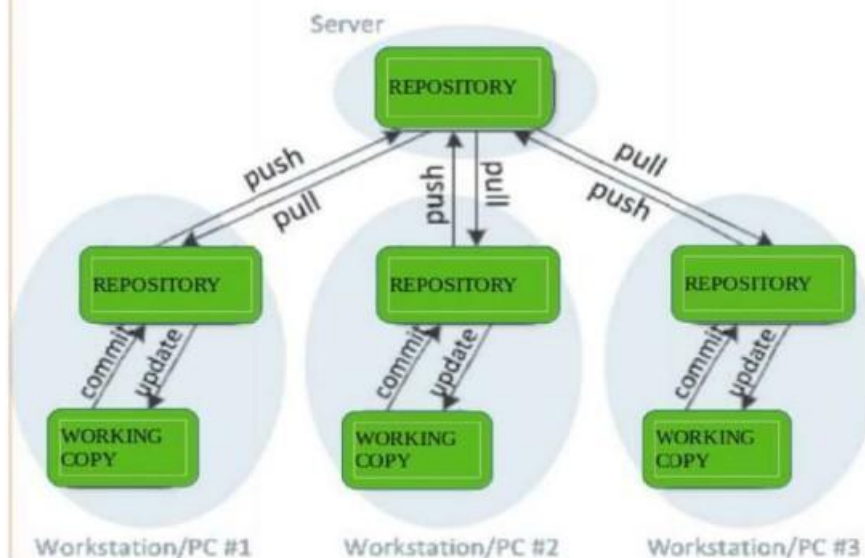


First when you make changes in your code you have to add the code to the commit docs and then commit.

After commit changes will be shown to the main central repository and the main team lead or head will change the version and push the changes if valid and then the new code and changes will be pulled to every other user connected to the centralized system (in this system every change made by any user will be reflected to all user).

2. Proper stepwise sequence to use VCS in Distributed Version Control System.

## Distributed version control



In this system every user has their own repository (branch/copy of the main code/data) where they make changes as per their role.

Then in the next step whenever they make changes to the code they push and comment what they have changed in their own repository and after that.

Main head or lead will review the comment and check the code and then commit and push the

code to the main server.

And then after any changes done to the main server code every user can pull the new code from their and start working on that or make their own changes which can be progressive or different then main code (in this system every user can have their own repository (set of data or code) which can be partially or fully different from the main code).

## **EXPERIMENT 9**

**AIM:** Demonstrate how Git can be used to perform version control?

**THEORY :** Version control is important to keep track of changes and keep every team member working off the latest version. We should use version control software for all code, files, and assets that multiple team members will collaborate on. It needs to do more than just manage and track files. It should help you develop and ship products faster. This is especially important for teams practicing DevOps.

That's because using the right one:

- Improves visibility.
- Helps teams collaborate around the world.
- Accelerates product delivery.

Git is a distributed version control software which you need to install on your local system in order to use it.

### **Git File Workflow**

#### **1. git init**

Usage: git init [repository name]

This command is used to start a new repository.

#### **2. Workspace Copy**

Users' active directory simply creates new files in this space and this will be tracked by the Git.

#### **3. Stage Area**

It is a place where all the modified files marked to be committed are placed.

#### **4. Local Repository**

User's copy of the version database or file and access all the files through local repos and push the change made to remote.

#### **5. Remote Repository**

It is a server where all the collaborators upload changes made to files.

#### **6. The Workflow**

The GitHub flow is a lightweight, branch-based workflow built around core Git commands used by teams around the globe—including ours.

The GitHub flow has six steps, each with distinct benefits when implemented:

1. **Create a branch:** Topic branches created from the canonical deployment branch (usually main) allow teams to contribute to many parallel efforts. Short-lived topic branches, in particular, keep teams focused and result in quick ships.
2. **Add commits:** Snapshots of development efforts within a branch create safe, revertible points in the project's history.
3. **Open a pull request:** Pull requests publicize a project's ongoing efforts and set the tone for a transparent development process.
4. **Discuss and review code:** Teams participate in code reviews by commenting, testing, and reviewing open pull requests. Code review is at the core of an open



and participatory culture.

5. **Merge:** Upon clicking merge, GitHub automatically performs the equivalent of a local 'git merge' operation. GitHub also keeps the entire branch development history on the merged pull request.
6. **Deploy:** Teams can choose the best release cycles or incorporate continuous integration tools and operate with the assurance that code on the deployment branch has gone through a robust workflow.

### Contribute to an existing repo:

```
# download a repository on GitHub.com to our machine
git clone https://github.com/me/repo.git

# change into the `repo` directory
cd repo

# create a new branch to store any new changes
git branch my-branch

# switch to that branch (line of development)
git checkout my-branch

# make changes, for example, edit `file1.md` and `file2.md` using the text editor

# stage the changed files
git add file1.md file2.md

# take a snapshot of the staging area (anything that's been added)
git commit -m "my snapshot"

# push changes to github
git push --set-upstream origin my-branch
```

### Start a new repo and publish it to GitHub

```
# create a new directory, and initialize it with git-specific functions
git init my-repo

# change into the `my-repo` directory
cd my-repo

# create the first file in the project
touch README.md

# git isn't aware of the file, stage it
git add README.md

# take a snapshot of the staging area
git commit -m "add README to initial commit"
```

### Contribute to an existing branch on GitHub

```

# assumption: a project called `repo` already exists on the machine

# change into the `repo` directory
cd repo

# update all remote tracking branches, and the currently checked out branch
git pull

# change into the existing branch called `feature-a`
git checkout feature-a

# make changes, for example, edit `file1.md` using the text editor

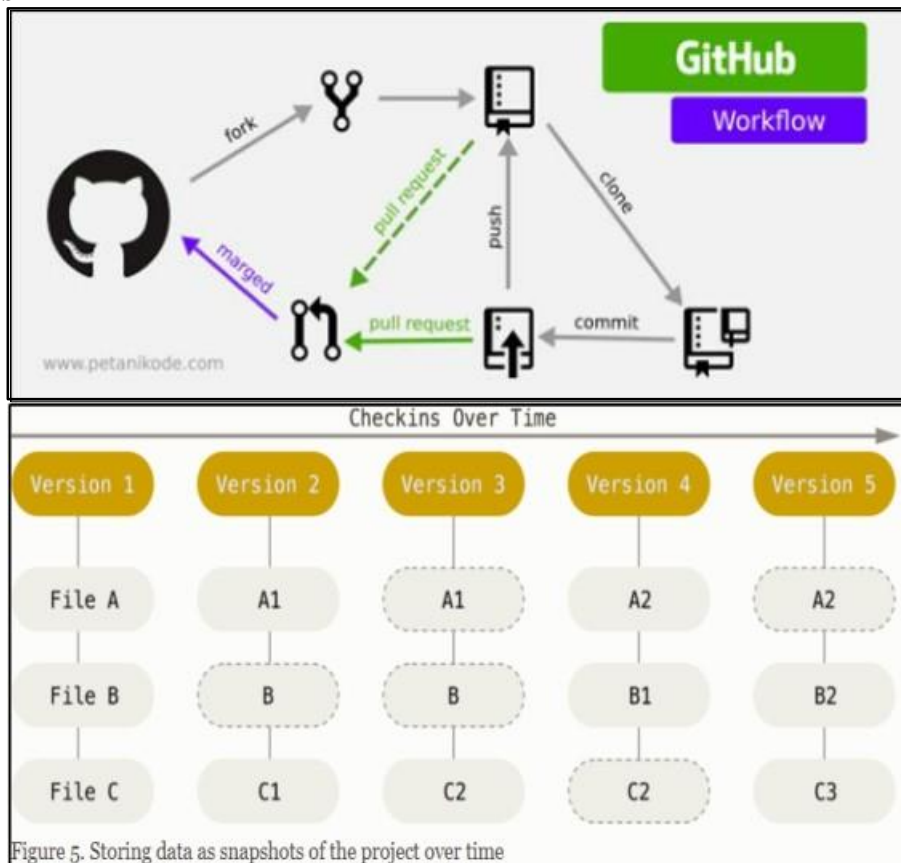
# stage the changed file
git add file1.md

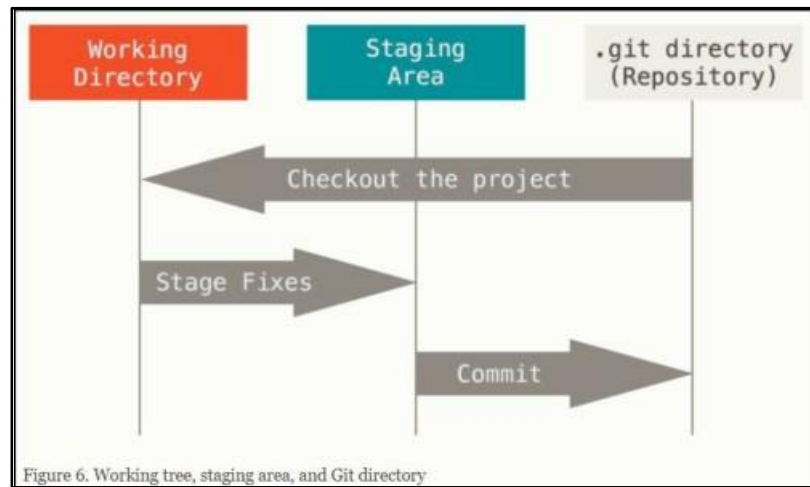
# take a snapshot of the staging area
git commit -m "edit file1"

# push changes to github
git push

```

## Diagrams





### Conclusions and Results:

In this Experiment we learned about how to work with git and understand its workflow.

## **EXPERIMENT 10**

**AIM:** Validate the results obtained in experiment 3 using 10-cross validation, hold out validation or leave one out cross-validation.

### **THEORY:**

**10-CROSS VALIDATION:** Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it. In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once. For classification problems, one typically uses stratified k-fold cross-validation, in which the folds are selected so that each fold contains roughly the same proportions of class labels. In repeated cross-validation, the cross-validation procedure is repeated n times, yielding n random partitions of the original sample. The n results are again averaged (or otherwise combined) to produce a single estimation.

### **PROCEDURE OF 10-FOLD CROSS VALIDATION:**

1. Shuffle the dataset randomly.
  2. Split the dataset into 10 groups
  3. For each unique group:
    1. Take the group as a hold out or test data set
    2. Take the remaining groups as a training data set
    3. Fit a model on the training set and evaluate it on the test set
    4. Retain the evaluation score and discard the model.
  4. Summarize the skill of the model using the sample of model evaluation scores
- Importantly, each observation in the data sample is assigned to an individual group and stays in that group for the duration of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model 9 times.

### **HOLDOUT VALIDATION:**

In the holdout method, we randomly assign data points to two sets  $d_0$  and  $d_1$ , usually called the training set and the test set, respectively. The size of each of the sets is arbitrary although typically the test set is smaller than the training set. We then train (build a model) on  $d_0$  and test (evaluate its performance) on  $d_1$ . In typical cross validation, results of multiple runs of model-testing are averaged together; in contrast, the holdout method, in isolation, involves a single run. It should be used with caution because without such averaging of multiple runs, one may achieve highly misleading results. One's indicator of predictive accuracy ( $F^*$ ) will tend to be unstable since it will not be smoothed out by multiple iterations (see below).

Similarly, indicators of the specific role played by various predictor variables (e.g., values of regression coefficients) will tend to be unstable. While the holdout method can be framed as "the simplest kind of cross-validation", many sources instead classify holdout as a type of simple validation, rather than a simple or degenerate form of cross-validation.

## LEAVE-ONE-OUT CROSS VALIDATION :

Leave-one-out cross-validation, or LOOCV, is a configuration of k-fold cross validation where k is set to the number of examples in the dataset. We then average ALL of these folds and build our model with the average. Because we would get a big number of training sets (equals to the number of samples), this method is very computationally expensive and should be used on small datasets. If the dataset is big, it would most likely be better to use a different method, like kfold.

## RESULTS

### 10 FOLD CROSS VALIDATION

```
# Use Decision tree as the prediction Model
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, accuracy_score, classification_report
import matplotlib.pyplot as plt
import math
import sklearn
from sklearn import datasets
from numpy import array
from sklearn.model_selection import KFold
iris = datasets.load_iris()
X = iris.data[:, :4] # we only take the first two features.
y = iris.target

decision_tree_classifier = DecisionTreeClassifier(criterion = 'gini',
random_state = 0)
kfold = KFold(10, True, 1)
# enumerate splits
i=1
for train, test in kfold.split(X,y):
    #print('train: %s, test: %s' % (X[train], y[train]))
    model=decision_tree_classifier.fit(X[train], y[train])
    y_pred = model.predict(X[test])
    print ("ACCURACY for ",i," :",accuracy_score(y[test], y_pred)*100)
    i=i+1
```

```
ACCURACY for 1 : 100.0
ACCURACY for 2 : 93.33333333333333
ACCURACY for 3 : 93.33333333333333
ACCURACY for 4 : 100.0
ACCURACY for 5 : 100.0
ACCURACY for 6 : 93.33333333333333
ACCURACY for 7 : 100.0
ACCURACY for 8 : 86.66666666666667
ACCURACY for 9 : 93.33333333333333
ACCURACY for 10 : 80.0
```

## HOLD OUT VALIDATION

```
# Use Decision tree as the prediction Model
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, accuracy_score, classification_report
import matplotlib.pyplot as plt
import math
import sklearn
from sklearn import datasets
from numpy import array
from sklearn.model_selection import KFold
iris = datasets.load_iris()
X = iris.data[:, :4]
y = iris.target
decision_tree_classifier = DecisionTreeClassifier(criterion = 'gini', random_state = 0)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4, random_state = 0)
model = decision_tree_classifier.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("ACCURACY :", accuracy_score(y_test, y_pred)*100)
cm = confusion_matrix(y_test, y_pred)
print("confusion matrix is:\n", cm)

ACCURACY : 95.0
confusion matrix is:
[[16  0  0]
 [ 0 22  1]
 [ 0  2 19]]
```

## LEAVE ONE OUT VALIDATION

```
# Use Decision tree as the prediction Model
from sklearn.model_selection import train_test_split, LeaveOneOut
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, accuracy_score, classification_report
import matplotlib.pyplot as plt
import math
import sklearn
from sklearn import datasets
from numpy import array
from sklearn.model_selection import KFold
iris = datasets.load_iris()
X = iris.data[:, :4] # we only take the first two features.
y = iris.target
decision_tree_classifier = DecisionTreeClassifier(criterion = 'gini',
random_state = 0)
loo = LeaveOneOut()
y_true, y_pred = list(), list()
for train, test in loo.split(X, y):

    model = decision_tree_classifier.fit(X[train], y[train])
    y_value = model.predict(X[test])
    y_true.append(y[test])
    y_pred.append(y_value)
acc = accuracy_score(y_true, y_pred)
print('Accuracy: %.3f' % (acc*100))
cm = confusion_matrix(y_true, y_pred)
print("confusion matrix is:\n", cm)
```

Output:

```
Accuracy: 95.333
confusion matrix is:
[[50  0  0]
 [ 0 46  4]
 [ 0  3 47]]
```