Empirical Software Engineering (SE-404)

LAB A1-G2

Laboratory Manual

Department of Software Engineering

DELHI TECHNOLOGICAL UNIVERSITY(DTU)

Shahbad Daulatpur, Bawana Road, Delhi-110042

Submitted to: -

Mr. Sanjay Patidar

Submitted by:-

Name: ASHISH KUMAR

Roll number: 2K18/SE/041

INDEX

S.No.	EXPERIMENT	DATE	REMARKS
10.	Perform a comparison of the following data analysis tools. WEKA, KEEL, SPSS, MATLAB, R.	04-01-2022	
1.	Consider any empirical study of your choice (Experiments, Survey Research, Systematic Review, Postmortem analysis and case study). Identify the following components for an empirical study: a. Identify parametric and nonparametric tests b. Identify Independent, dependent and confounding variables c. Is it Within-company and cross-company analysis? d. What type of dataset is used? Proprietary and open-source software	18-01-2022	
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.	25-01-2022	
3.	Consider any prediction model of your choice. a. Analyze the dataset that is given as a 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.	25-01-2022	
8.	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).	01-02-2022	
9.	Demonstrate how Git can be used to perform version control?	01-02-2022	
11.	Validate the results obtained in experiment 3 using 10-cross validation, hold out validation or leave one out cross-validation.	15-02-2022	
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	23-02-2022	
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 hypothesos for the followings 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.		
6.	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		
7.	Consider defect dataset and implement following statistical test using SPSS tool. a. t-test b. Chi-Square Test c. Wilcoxon Signed Test d. Friedman Test e. Kruskal Wallis Test		

Empirical Software Engineering LAB – A1 G2
EXPERIMENT 10

Experiment Objective:- Perform a comparison of the following data analysis tools.

a. WEKA
b. KEEL
c. SPSS
d. MATLAB
e. R

Introduction:-

1. **WEKA:** Also known as Waikato Environment is a machine learning software developed at the University of Waikato in New Zealand. WEKA is a free and open source software package that assembles a wide range of data mining and model building algorithms.
   It is best suited for data analysis and predictive modeling. It contains algorithms and visualization tools that support machine learning. WEKA has a GUI that facilitates easy access to all its features. It is written in JAVA programming language and runs on almost any platform. WEKA supports major data mining tasks including data mining, processing, visualization, regression etc. It works on the assumption that data is available in the form of a flat file. WEKA can provide access to SQL Databases through database connectivity and can further process the data/results returned by the query.

[Source: screenshot of software]

WEKA supports four different frameworks for developing and executing the methods for DM tasks:

- Explorer: An environment for exploring data with WEKA.
- Experimenter: An environment for performing experiments and conducting statistical tests between learning schemes.
- KnowledgeFlow: This environment supports essentially the same functions as the Explorer but with a drag-and-drop interface. One advantage is that it supports incremental learning.
- SimpleCLI: Provides a simple command-line interface that allows direct execution of WEKA commands for operating systems that do not provide their own command line interface

Advantages of WEKA include:

- Free availability under the GNU General Public License.
- Portability, since it is fully implemented in the Java programming language and thus runs on almost any modern computing platform.

- A comprehensive collection of data preprocessing and modeling techniques.
- Ease of use due to its graphical user interfaces.

Disadvantages of WEKA include:

- It is not capable of multi-relational data mining.
- It can only handle small datasets. Whenever a set is bigger than a few megabytes an "Out Of Memory" error occurs.

2. **KEEL:** KEEL is a free software Java tool which empowers the user to assess the behavior of evolutionary learning and soft computing based techniques for different kind of data mining problems: regression, classification, clustering, pattern mining and so on. KEEL is a data mining tool used by many EDM researchers. For instance, KEEL has extremely extensive support for discretization algorithms, but has limited support for other methods for engineering new features out of existing features. It has excellent support for feature selection, with a wider range of algorithms than any other package. It also has extensive support for imputation of missing data, and considerable support for data re-sampling. KEEL is open-source and free for use under a GNU license.

   For modeling, KEEL has an extensive set of classification and regression algorithms; with a large focus on evolutionary algorithms. Its support for other types of data mining algorithms, such as clustering and factor analysis, is more limited than other packages. Support for association rule mining is decent, though not as extensive as some other packages.

[Source: screenshot of software]

The main features of KEEL are:

- It contains a large collection of evolutionary algorithms for predicting models, preprocessing methods (evolutionary feature and instance selection among others) and postprocessing procedures (evolutionary tuning of fuzzy rules). It also presents many state-of-the-art methods for different areas of data mining such as decision trees, fuzzy rule based systems or crisp rule learning.
- It includes around 100 data preprocessing algorithms proposed in the specialized literature: data transformation, discretization, instance and feature selection, noise filtering and so forth.
- It incorporates a statistical library to analyze the results of the algorithms.
- It comprises a set of statistical tests for analyzing the suitability of the results and for performing

parametric and nonparametric comparisons among the algorithms.

- It provides a user-friendly interface, oriented to the analysis of algorithms.
- KEEL also allows creating experiments in on-line mode, aiming an educational support in order to learn the operation of the algorithms included.
- The software is aimed to create experimentations containing multiple datasets and algorithms to obtain results. Experiments are independently script-generated from the user interface for an off-line run in any machine that supports a Java Virtual Machine.

3. **SPSS:** SPSS stands for "Statistical Package for the Social Sciences". It is an IBM tool. This tool first launched in 1968. This is a software package which is mainly used for statistical analysis of the data.
SPSS is mainly used in the following areas like healthcare, marketing, and educational research, market researchers, health researchers, survey companies, education researchers, government, marketing organizations, data miners, and many others. It provides data analysis for descriptive statistics, numeral outcome predictions, and identifying groups. This software also gives data transformation, graphing and direct marketing features to manage data smoothly.

Features of SPSS are:

• The data from any survey collected via Survey Gizmo gets easily exported to SPSS for detailed and good analysis.
• In SPSS, data gets stored in .SAV format. These data mostly comes from surveys. This makes the process of manipulating, analyzing and pulling data very simple.
• SPSS have easy access to data with different variable types. These variable data is easy to understand. SPSS helps researchers to set up model easily because most of the process is automated.
• SPSS allows Opening data files, either in SPSS' own file format or many others.
• SPSS allows editing data such as computing sums and means over columns or rows of data. SPSS has outstanding options for more complex operations as well.
• SPSS has option for creating tables and charts containing frequency counts or summary statistics over (groups of) cases and variables.
• SPSS has a unique way to get data from critical data also. Trend analysis, assumptions, and predictive models are some of the characteristics of SPSS.
• SPSS is easy for you to learn, use and apply.
• It helps in to get data management system and editing tools handy.
• SPSS offers you in-depth statistical capabilities for analyzing the exact outcome.
• SPSS helps us to design, plotting, reporting and presentation features for more clarity.

Limitations are:
• SPSS is expensive to purchase for students.
• Usually involves added training to completely exploit all the available features.
• The graph features are not as simple as of Microsoft Excel.
• Documentation about algorithms is sometimes difficult or impossible to find.
• Information about effect size and confidence intervals is missing for many techniques.

4. **MATLAB:** MATLAB stands for Matrix laboratory. It was developed by Mathworks, and it is a multipurpose (or as we say it Multi-paradigm) programming language. It allows matrix manipulations and helps us to plot different types of functions and data. It can also be used for the analysis and design as such as the control systems.

Features of MATLAB:

- It is a high-level language for numerical computation, visualization and application development.
- It also provides an interactive environment for iterative exploration, design and problem solving.

- It provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.
- It provides built-in graphics for visualizing data and tools for creating custom plots.
- MATLAB's programming interface gives development tools for improving code quality maintainability and maximizing performance.
- It provides tools for building applications with custom graphical interfaces.
- It provides functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET and Microsoft Excel.

Advantages of MATLAB are:

- Easy to use interface: A user-friendly interface with features you want to use is one click away.
- A large inbuilt database of algorithms: MATLAB has numerous important algorithms you want to use already built-in, and you just have to call them in your code.
- Extensive data visualization and processing: We can process a large amount of data in MATLAB and visualize them using plots and figures.
- Debugging of codes easy: There are many inbuilt tools like analyzer and debugger for analysis and debugging of codes written in MATLAB.
- Easy symbolic manipulation: We can perform symbolic math operations in MATLAB using the symbolic manipulation algorithms and tools in MATLAB

Disadvantages of MATLAB are:

- MATLAB is slow since it is an interpreted language that is MATLAB programs are not converted into Machine language but are run by external software, so it can sometimes be slow.
- We cannot create the OUTPUT file in MATLAB.
- One cannot use graphics in MATLAB with -nojvm option, on doing so, we will get a runtime error.
- We cannot make functions in one single .m file as we have in the case of other programming languages. We have to create different files for different functions.
- Sometimes, the error messages are not much informative, so you have to figure out the error yourself.

5. R: R is a popular and powerful open source programming language and software environment for statistical computing and graphics representation. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team. R implements various statistical techniques like linear and non-linear modeling, machine learning algorithms, time series analysis, and classical statistical tests and so on. R consists of a language and a run-time environment with graphics, a debugger, access to certain system functions, and the ability to run programs stored in script files. R allows integration with the procedures written in the C, C++, .Net, Python or FORTRAN languages for efficiency.

[Source: Wikipedia]

Features of R:

- R is a well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.
- R has an effective data handling and storage facility,
- R provides a suite of operators for calculations on arrays, lists, vectors and matrices.
- R provides a large, coherent and integrated collection of tools for data analysis.
- R provides graphical facilities for data analysis and display either directly at the computer or printing at the papers.

Advantages of R:

1) Open Source: R is an open-source language. We can contribute to the development of R by optimizing our packages, developing new ones, and resolving issues.

2) Platform Independent: R is a platform-independent language or cross-platform programming language which means its code can run on all operating systems. R can run quite easily on Windows, Linux, and Mac.

3) Machine Learning Operations: R allows us to do various machine learning operations such as classification and regression. R is used by the best data scientists in the world.

4) Exemplary support for data wrangling: R allows us to perform data wrangling. R provides packages such as dplyr, readr which are capable of transforming messy data into a structured form.

5) Quality plotting and graphing: R simplifies quality plotting and graphing. R libraries such as ggplot2 and plotly advocates for visually appealing and aesthetic graphs which set R apart from other programming languages.

6) Statistics: R is mainly known as the language of statistics. It is the main reason why R is predominant than other programming languages for the development of statistical tools.
7) Highly Compatible: R is highly compatible and can be paired with many other programming languages like C, C++, Java, and Python. It can also be integrated with technologies like Hadoop and various other database management systems as well.

Disadvantages of R:

1) Data Handling: In R, objects are stored in physical memory. It is in contrast with other programming languages like Python. R utilizes more memory as compared to Python. It requires the entire data in one single place which is in the memory. It is not an ideal option when we deal with Big Data.

2) Basic Security: R lacks basic security. Because of this, there are many restrictions with R as it cannot be embedded in a web-application.

3) Complicated Language: R is a very complicated language, and it has a steep learning curve. The people who don't have prior knowledge or programming experience may find it difficult to learn R.

4) Weak Origin: The another disadvantage of R is, it does not have support for dynamic or 3D graphics. The reason behind this is its origin. It shares its origin with a much older programming language "S."

5) Lesser Speed: R programming language is much slower than other programming languages such as MATLAB and Python. In comparison to other programming language, R packages are much slower.

Learning from experiment:- We have successfully discussed and learnt various data analysis tools. We have compared 5 different analysis tools WEKA, KEEL, SPSS, MATLAB, R and mentioned their advantages and disadvantages as well.

Empirical Software Engineering LAB – A1 G2
EXPERIMENT 1

Experiment Objective:- Consider any empirical study of your choice (Experiments, Survey Research, Systematic Review, Postmortem analysis and case study). Identify the following components for an empirical study:
a. Identify parametric and nonparametric tests.
b. Identify Independent, dependent and confounding variables.
c. Is it Within-company and cross-company analysis?
d. What type of dataset is used? Proprietary and open-source software.

Introduction:-

- **Parametric and non-parametric tests:** Parametric tests are used for data samples having normal distribution (bell-shaped curve), whereas non-parametric tests are used when the distribution of data samples is highly skewed.

- **Independent variables:** Independent variables (or predictor variables) are input variables that are manipulated or controlled by the researcher to measure the response of the dependent variable.

- **Dependent 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.

- **Confounding variables:** A confounding variable is a third variable that influences both the independent and dependent variables. Failing to account for confounding variables can cause you to wrongly estimate the relationship between your independent and dependent variables.

- **Within-company analysis:** In within-company analysis, the empirical study collects the data from the old versions/ releases of the same software, predicts models, and applies the predicted models to the future versions of the same project.

- **Cross-company analysis:** The process of validating the predicted model using data collected from different projects from which the model has been derived is known as cross-company analysis.

- **Proprietary software:** Proprietary software is licensed software owned by a company. For example, Microsoft.

- **Open source software:** Open source software is usually a freely available software, developed by many developers from different places in a collaborative manner. For example, Google Chrome, Android operating system, and Linux operating system.

I have chosen **Naïve Bayes Classifier** as a case study for this experiment.
Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

To start with, let us consider a dataset.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit ("Yes") or unfit ("No") for playing golf.

Here is a tabular representation of chosen dataset:

	Outlook	Temperature	Humidity	Windy	Play Golf
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

[Source: Geeksforgeeks]

Result:-

In the given case study of Naive Bayes' Classifier, following are the identified attributes:

1. **Parametric Test:** Since the attributes mentioned in the dataset have normal distribution. So, parametric test can be used is t-test since dataset is small and have normal distribution of data.
2. **Non-Parametric Test:** None
3. **Independent Variables: '**Outlook', 'Temperature', 'Humidity' and 'Windy'.
4. **Dependent Variables:** 'Play Golf'
5. **Cofounding variables:** None as no variable is there that influences both the independent and dependent variables.
6. **Within Company and cross-company analysis:** Since the data is taken from single source, hence it is within company.
7. **Dataset:** The dataset is an open-source dataset, publicly available on the Geeksforgeeks website. This dataset describes the weather conditions whether for playing golf is fit ("Yes") or unfit ("No").

Learning from experiment:- We have successfully learned about parametric and non-parametric test. I was able to identify dependent and independent variables in chosen case study. There is no cofounding variable. It was open-source software and has conducted i.e. parametric and non-parametric tests as well. We have also learned about the Difference between within-company and cross-company analysis.

Empirical Software Engineering LAB – A1 G2
EXPERIMENT 2

Experiment Objective:- 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.

Introduction:-

Root cause analysis (RCA) is the process of discovering the root causes of problems in order to identify appropriate solutions. **The first goal** of root cause analysis is to discover the root cause of a problem or event. **The second goal** is to fully understand how to fix, compensate, or learn from any underlying issues within the root cause. **The third goal** is to apply what we learn from this analysis to systematically prevent future issues or to repeat successes.

Result:-

a. 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.

Dividing a number by Zero is a mathematical error (not defined) and we can use exception handling to gracefully overcome such operations.

b. Usage of 3.14 in the statement Circle_Area = 3.14 * Radius * Radius;

Logical Defect:

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. Solution would be 3.14 can be declared as macro.

c. 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.

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

Logical Defect and Standards:

A pointer is a variable that holds the address of another variable. So solution would be to initialize the pointer properly and assign a value to it.

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

Logical Defect, User Interface:

Solution would be to do proper load and stress testing before hosting to the internet for user usage.

f. A "while" loop never exits

Logical Defect:

Solution would be to write appropriate condition to end the loop otherwise the program would run infinite times.

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

User Interface:

Solution would be to do proper error handling and make an appropriate display message so that user would get to know what exact problem is. 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.

h. No documentation (comments) for the source code

Standards:

Solution: It is considered to be a good coding standard to name the functions according to what they perform and to add comments explaining what function are actually doing and proper documentations should be done.

i. Hungarian Notation not followed while coding, even though the coding guidelines mandate to use Hungarian Notation

Standards:

Solution: It is always good to follow standards i.e. Hungarian Notation while coding. All coding guidelines should be followed for successful software development.

j. Pressing of "Tab" key moves the cursor in different fields of a web form randomly.

User Interface:

Solution would be to design forms properly and to perform form-based testing before hosting to internet.

Learning from experiment:- We have successfully learned about RCA and from which we learned different types of testing and found different defects in the different scenarios. We also discussed solutions to prevent these defects.

Empirical Software Engineering LAB – A1 G2
EXPERIMENT 3

Experiment Objective:- Consider any prediction model of your choice.
a. Analyze the dataset that is given as a 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.

Introduction:-

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.].
Here is the link of the research paper:
http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=47D7383DC75311874ED04A78F4B955A1?doi=10.1.1.149.3711&rep=rep1&type=pdf

Code and Result:-

a) Link of the dataset used for this experiment: 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.

Feature Used as the Response Variable:
==========
NUMDEFECTS: The number of defects recorded for the class,

Features at Class Level Originally
==========
PERCENT_PUB_DATA: The percentage of data that is public and protected data in a class. In general, lower values indicate greater encapsulation. It is measure of encapsulation.

ACCESS_TO_PUB_DATA: The amount of times that a class's public and protected data is accessed. In general, lower values indicate greater encapsulation. It is a measure of encapsulation.

COUPLING_BETWEEN_OBJECTS: The number of distinct non-inheritance-related classes on which a class depends. If a class that is heavily dependent on many classes outside of its hierarchy is introduced into a library, all the classes upon which it depends need to be introduced as well. This may be acceptable, especially if the classes which it references are already part of a class library and are even more fundamental than the specified class.

DEPTH: The level for a class. For instance, if a parent has one child the depth for the child is two. Depth indicates at what level a class is located within its class hierarchy. In general, inheritance increases when depth increases.

LACK_OF_COHESION_OF_METHODS: 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%. The LOCM metric indicates low or high percentage of cohesion. If the percentage is low, the class is cohesive. If it is high, it may indicate that the class could be split into separate classes that will individually have greater cohesion.

NUM_OF_CHILDREN: The number of classes derived from a specified class.

DEP_ON_CHILD: Whether a class is dependent on a descendant.

FAN_IN: This is a count of calls by higher modules.

RESPONSE_FOR_CLASS: A count of methods implemented within a class plus the number of methods accessible to an object class due to inheritance. In general, lower values indicate greater polymorphism.

WEIGHTED_METHODS_PER_CLASS: A count of methods implemented within a class (rather than all methods accessible within the class hierarchy). In general, lower values indicate greater polymorphism.

Note:- I have used python language to analyze the dataset and used pandas, numpy libraries to extract and analyze the data.

Below is the screenshot of the Statistical Summary of dataset after extracting it from arff file:

jupyter ESE Expt3 Last Checkpoint: 8 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [15]:
```
from scipy.io import arff
import pandas as pd
import numpy as np
import csv
import urllib.request
import io # for io.StringIO()
```

In [16]:
```
url = "http://promise.site.uottawa.ca/SERepository/datasets/kc1-class-level-numericdefect.arff"
ftpstream = urllib.request.urlopen(url)
data = arff.loadarff(io.StringIO(ftpstream.read().decode('utf-8')))
df = pd.DataFrame(data[0])
```

In [17]:
```
df.rename(columns = {'COUPLING_BETWEEN_OBJECTS':'CBO'}, inplace = True)
df.rename(columns = {'LACK_OF_COHESION_OF_METHODS':'LCOM'}, inplace = True)
df.rename(columns = {'NUM_OF_CHILDREN':'NOC'}, inplace = True)
df.rename(columns = {'DEP_ON_CHILD':'DOC'}, inplace = True)
df.rename(columns = {'RESPONSE_FOR_CLASS':'RFC'}, inplace = True)
df.rename(columns = {'WEIGHTED_METHODS_PER_CLASS':'WMC'}, inplace = True)
```

In [18]:
```
df.describe()
```

Out[18]:

	PERCENT_PUB_DATA	ACCESS_TO_PUB_DATA	CBO	DEPTH	LCOM	NOC	DOC	FAN_IN	RFC	WMC	...	sun
count	145.000000	145.0	145.000000	145.000000	145.000000	145.000000	145.000000	145.000000	145.000000	145.000000	...	
mean	14.400000	0.0	8.317241	2.000000	68.724138	0.213793	0.013793	0.634483	34.379310	17.420690	...	
std	32.532975	0.0	6.376719	1.258306	36.888624	0.699069	0.117036	0.695360	36.202952	17.449001	...	
min	0.000000	0.0	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
25%	0.000000	0.0	3.000000	1.000000	58.000000	0.000000	0.000000	0.000000	10.000000	8.000000	...	
50%	0.000000	0.0	8.000000	2.000000	84.000000	0.000000	0.000000	1.000000	28.000000	12.000000	...	
75%	0.000000	0.0	14.000000	2.000000	96.000000	0.000000	0.000000	1.000000	44.000000	22.000000	...	
max	100.000000	0.0	24.000000	7.000000	100.000000	5.000000	1.000000	3.000000	222.000000	100.000000	...	

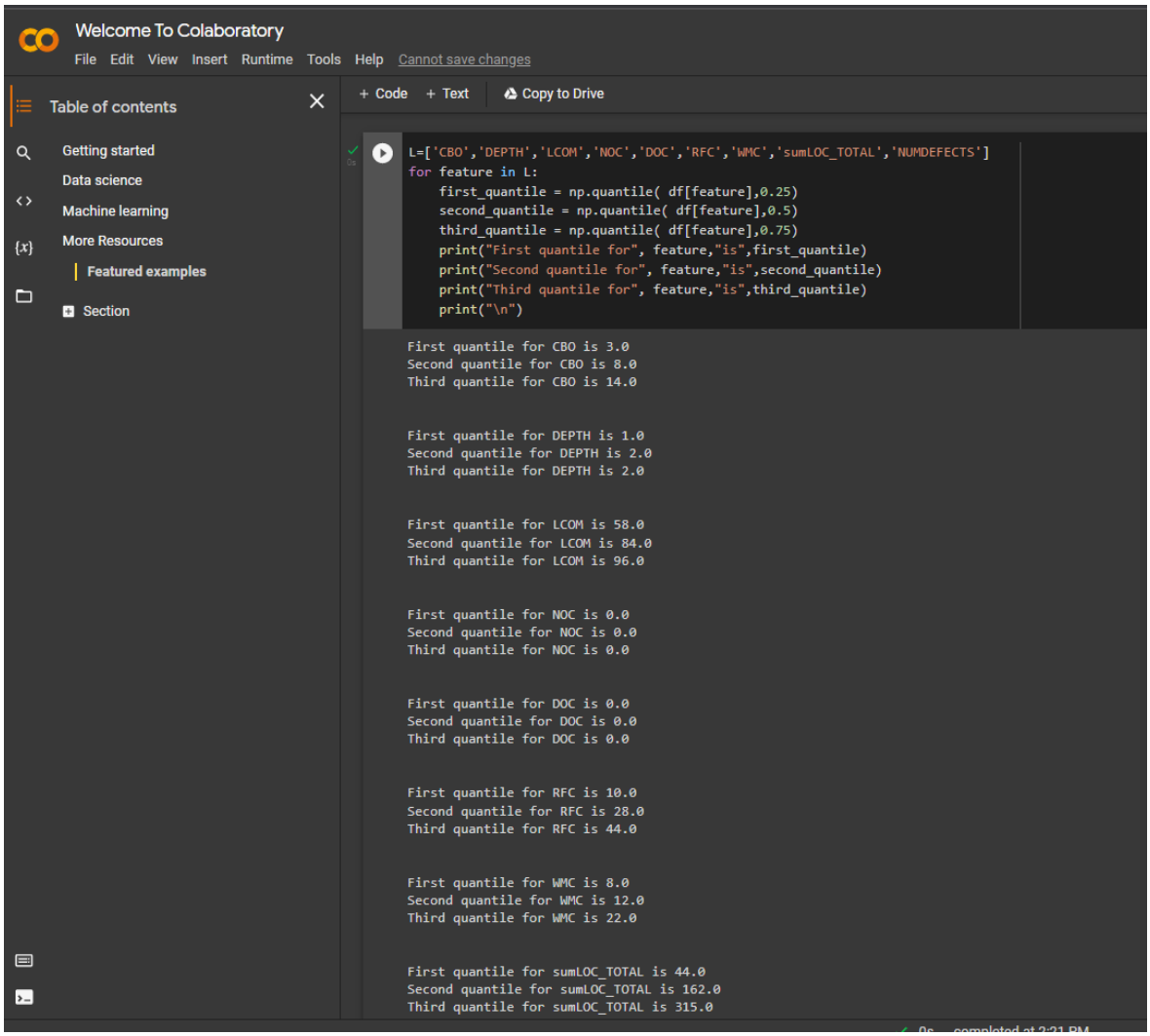
8 rows × 95 columns

b)
The quantile( ) function of NumPy is used to find quartiles of the dataset for each of the metrics or attributes used. Quantiles are the set of values/points that divides the dataset into groups of equal size.

Note: I have used Google Colab for this because jupyter notebook is showing error for quantile function.

Below is the code of it and screenshot of output:

```
L=['CBO','DEPTH','LCOM','NOC','DOC','RFC','WMC','sumLOC_TOTAL','NUMDEFECTS']
for feature in L:
    first_quantile = np.quantile( df[feature],0.25)
    second_quantile = np.quantile( df[feature],0.5)
    third_quantile = np.quantile( df[feature],0.75)
    print("First quantile for", feature,"is",first_quantile)
    print("Second quantile for", feature,"is",second_quantile)
    print("Third quantile for", feature,"is",third_quantile)
    print("\n")
```



c) The Model evaluation metrics used are:

Classification Accuracy: Classification accuracy is the number of correct predictions made as a ratio of all predictions made. This is the most common evaluation metric for classification problems.

Below is the code of it and output:

Note: Classification Accuracy comes out to be 0.7297297297297297 i.e. 72%.

jupyter ESE Expt3 Last Checkpoint: 23 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted

Run Code

In [45]:
```
feature_cols = ['COUPLING_BETWEEN_OBJECTS', 'DEPTH', 'LACK_OF_COHESION_OF_METHODS', 'NUM_OF_CHILDREN','DEP_ON_CHILD']

X = df[feature_cols]

y = df.loc[:,'PERCENT_PUB_DATA']
```

In [46]:
```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

In [47]:
```
from sklearn.linear_model import LogisticRegression

# instantiate model
logreg = LogisticRegression()

# fit model
logreg.fit(X_train, y_train)
```

Out[47]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [48]:
```
y_pred_class = logreg.predict(X_test)
```

In [50]:
```
from sklearn import metrics
print("Classification Accuracy is",metrics.accuracy_score(y_test, y_pred_class))
```

```
Classification Accuracy is 0.7297297297297297
```

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

- Also known as "Misclassification Rate"

In [62]:
```
classification_error = (FP + FN) / float(TP + TN + FP + FN)

print(classification_error)
print(1 - metrics.accuracy_score(y_test, y_pred_class))
```

```
0.03571428571428571
0.2702702702702703
```

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

- How "sensitive" is the classifier to detecting positive instances?

```
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?

- Something we want to maximize
- How "specific" (or "selective") is the classifier in predicting positive instances?

In [64]:
```
specificity = TN / (TN + FP)

print(specificity)
```

```
1.0
```

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

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

```
In [67]: false_positive_rate = FP / float(TN + FP)

         print(false_positive_rate)
         print(1 - specificity)

         0.0020325203252032522
         0.002032520325203291

In [68]: precision = TP / float(TP + FP)

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

         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.

```
In [78]: #Completeness Score
         from sklearn.metrics.cluster import completeness_score
         print(1-completeness_score(y_test,y_pred_class))

         0.9384092289227242

In [ ]:
```

Note: I have used this site as a source to evaluate c point:
https://www.ritchieng.com/machine-learning-evaluate-classification-model/

Learning from experiment:- Using the Support Vector Machine Predictive Model for Software Fault Proneness Prediction using OO metrics of KC1 NASA dataset, we analyzed the dataset, found out quartiles and analyzed the performance of the model using various performance metrics using Python language.

Empirical Software Engineering LAB – A1 G2
EXPERIMENT 8

Experiment Objective:- 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).

Introduction:- 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.
Version control allows you to keep track of your work and helps you to easily explore the changes you have made, be it data, coding scripts, notes, etc. With version control software such
as Git, version control is much smoother and easier to implement. Using an online platform like Github to store your files means that you have an online backup of your work, which is beneficial for both you and your collaborators.
For example: If you are a graphic or web designer and want to keep every version of an image or layout (which you would most certainly want to), a Version Control System (VCS) is a very wise thing to use. It allows you to revert selected files back to a previous state, revert the entire project back to a previous state, compare changes over time, and see who last modified and more.

Version control helps solve these kinds of problems and provides:

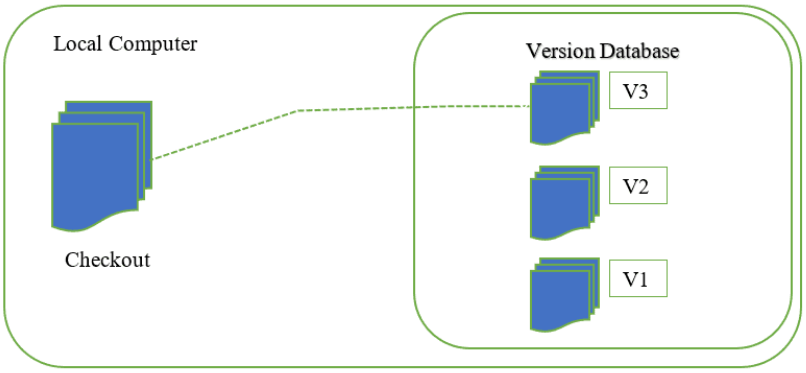
- A complete history of every file, which enables you to go back to previous versions to analyze the source of bugs and fix problems in older versions.
- The ability to work on independent streams of changes, which allow you to merge that work back together and verify your changes conflict.
- The ability 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.

Benefits of the version control system:

a) Enhances the project development speed by providing efficient collaboration,
b) Leverages the productivity, expedite product delivery, and skills of the employees through better communication and assistance,
c) Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small change,
d) Employees or contributor of the project can contribute from anywhere irrespective of the different geographical locations through this VCS,
e) For each different contributor of the project a different working copy is maintained and not merged to the main file unless the working copy is validated. A most popular example is Git, Helix core, Microsoft TFS,
f) Helps in recovery in case of any disaster or contingent situation.

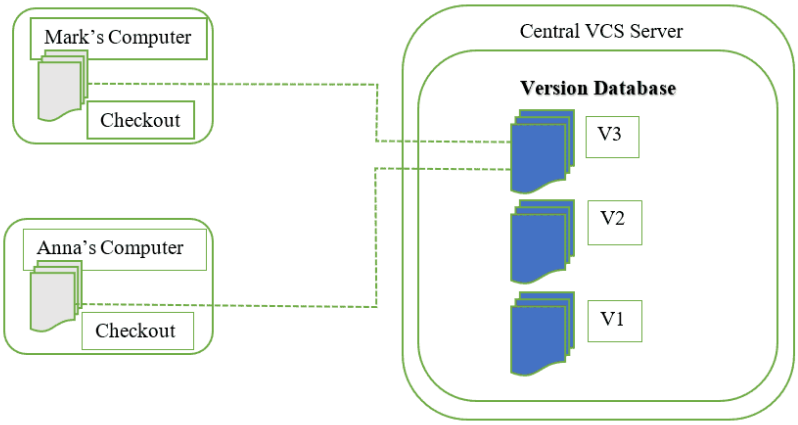
Types of version control Systems:
There are three types of version control: centralized and distributed.

1. **Local Version Control Systems:** It is one of the simplest forms and has a database that kept 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.



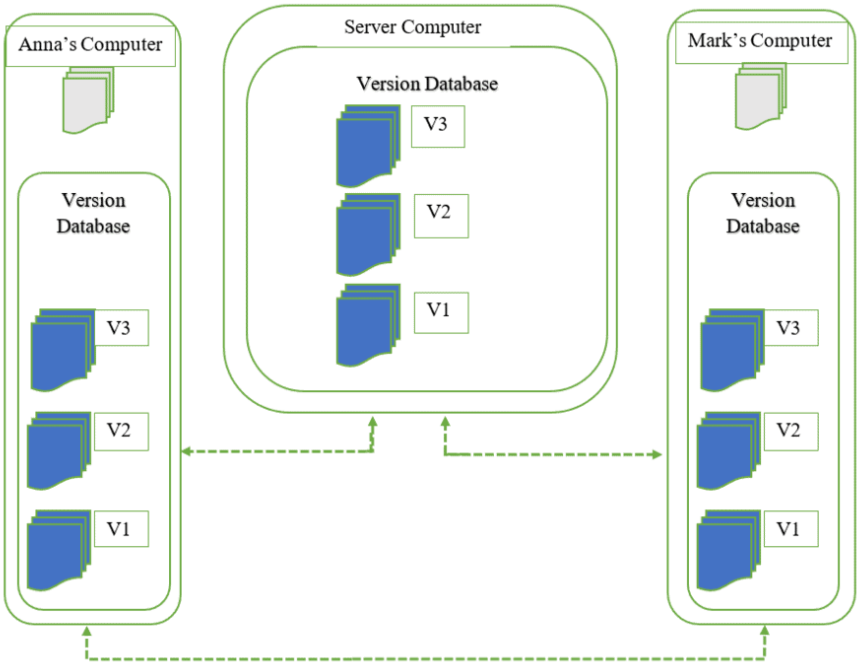
[Fig: Local VCS]

2. **Centralized version control System:** 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.



[Fig: Centralized VCS]

3. **Distributed version control systems**: 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. With this model, if the server becomes unavailable or dies, any of the client repositories can send a copy of the project's version to any other client or back onto the server when it becomes available. Git is the most well-known example of distributed version control systems.



[Fig: Distributed VCS]

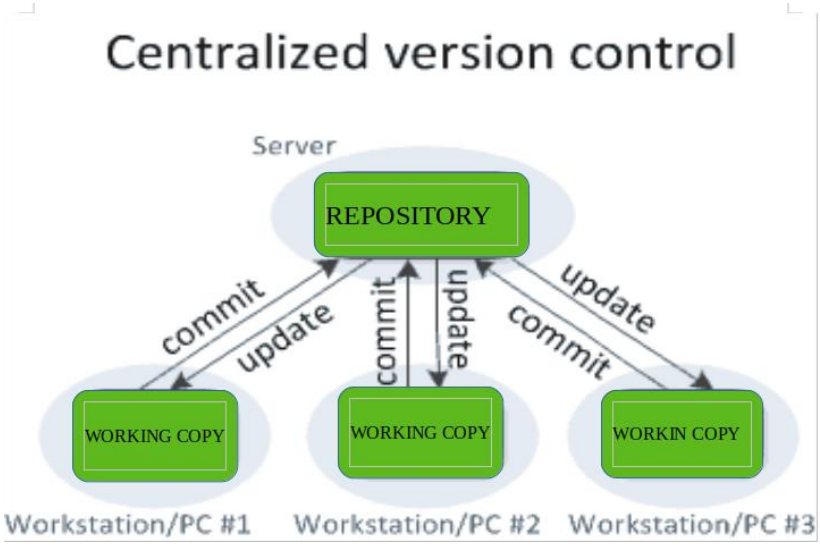
Here are a few of the most popular types of VCS:

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

Note: I have used this source: https://serengetitech.com/tech/introduction-to-git-and-types-of-version-control-systems/

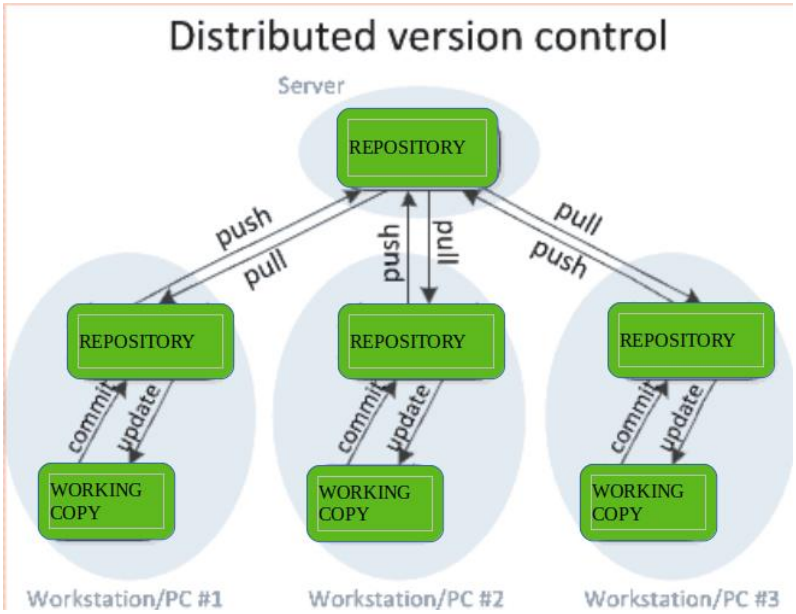
Version control System stepwise proper sequence:

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



[Source: Geeksforgeeks]

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



[Source: Geeksforgeeks]

Learning from experiment:- We have successfully learned about the version control system(VCS) and its benefits. We have also learned about the type of VCS and stepwise sequence of VCS.

Empirical Software Engineering LAB – A1 G2
EXPERIMENT 9

Experiment Objective:- Demonstrate how Git can be used to perform version control?

Introduction:- Version control allows you to keep track of your work and helps you to easily explore the changes you have made, be it data, coding scripts, notes, etc. With version control software such as Git, version control is much smoother and easier to implement. Using an online platform like Github to store your files means that you have an online backup of your work, which is beneficial for both you and your collaborators.

Benefits of using GIT as a version control Tool

Having a GitHub repo makes it easy for you to keep track of collaborative and personal projects - all files necessary for certain analyses can be held together and people can add in their code, graphs, etc. as the projects develop. Each file on GitHub has a history, making it easy to explore the changes that occurred to it at different time points. You can review other people's code, add comments to certain lines or the overall document, and suggest changes. For collaborative projects, GitHub allows you to assign tasks to different users, making it clear who is responsible for which part of the analysis. You can also ask certain users to review your code. For personal projects, version control allows you to keep track of your work and easily navigate among the many versions of the files you create, whilst also maintaining an online backup.

GITHUB WORKFLOW:

The GitHub workflow can be summarized by the commit-pull push" mantra.

● **Commit:** Once you've saved your files, you need to commit them - this means the changes you have made to files in your repo will be saved as a version of the repo, and your changes are now ready to go up on GitHub (the online copy of the repository).

● **Pull:** Now, before you send your changes to Github, you need to pull, i.e. make sure you are completely up to date with the latest version of the online version of the files - other people could have been working on them even if you haven't. You should always pull before you start editing and before you push.

● **Push:** Once you are up to date, you can push your changes – at this point in time your local copy and the online copy of the files will be the same.

How to get started?

Step 1: Sign up and installation!

First register on the **Github website**.

If you are on a personal Windows machine, download and install Git for your operating system.

Install Git on Windows:

1. Navigate to the latest Git for Windows installer and download the latest version.
2. Once the installer has started, follow the instructions as provided in the **Git Setup** wizard screen until the installation is complete.
3. Open the windows command prompt (or **Git Bash** if you selected not to use the standard Git Windows Command Prompt during the Git installation).
4. Type git version to verify Git was installed.

- If git is successfully installed in your computer, then you will see something like this when you type this command in cmd:
  git

git –version

Now you're ready to start using Git on your computer!

To get started, you can create a new repository on the GitHub website or perform a `git init` tocreate a new repository from your project directory.

From the terminal

Here's how you can get started right from the terminal:

- If you have a project directory, just go to your terminal and in your project directory runthe command:

  git init

- If you want to initialize your project with all of the files in your project directory, run following command to include everything

  git init .

- Let's say you have a folder for your project called "new_project." You could head on overto that folder in your terminal window and add a local repository to it by running:

  cd new_projectgit init

- Now you can add files to the staging area one by one with:

  git add<filename>

  or run

  git add .

- to add all of your files to the staging area. You can commit these changes with thecommand:

  git commit –m "<add a commit message here>"

- Now, copy the remote repository URL provided by github to you when you published your repository on GitHub.

  git remote add origin https://github.com/yourusername/your-repo-name.git
- In the last step, use the below command line in your terminal to push the local repository to GitHub. It will upload the file or project on github.

  git push origin master

- You can check whether or not you have changes to push through any time by running

  git status

- Pull the desired branch from the upstream repository. This method will retain the commit history without modification.

  git pull origin master

That's it! You can now initialize a repository, commit files, commit changes, and push themthrough to the master branch.

Learning from experiment:- In this Experiment we learned about how to work with git and understand its workflow.

Empirical Software Engineering LAB – A1 G2
EXPERIMENT 11

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

Introduction:- **Cross-Validation** also referred to **as out of sampling technique** is an essential element of a data science project. It is a resampling procedure used to evaluate machine learning models and access how the model will perform for an independent test dataset.

1. Leave p-out cross-validation: Leave p-out cross-validation (LpOCV) is an exhaustive cross-validation technique, that involves using p-observation as validation data, and remaining data is used to train the model. This is repeated in all ways to cut the original sample on a validation set of *p* observations and a training set.

2. Leave-one-out cross-validation: Leave-one-out cross-validation (LOOCV) is an exhaustive cross-validation technique. It is a category of LpOCV with the case of p=1.

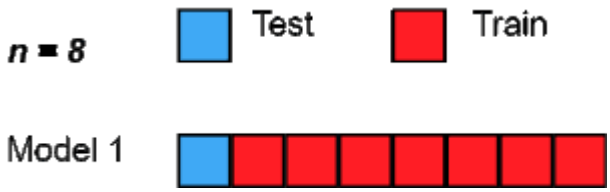


Fig. LOOCV operations

For a dataset having n rows, 1st row is selected for validation, and the rest (n-1) rows are used to train the model. For the next iteration, the 2nd row is selected for validation and rest to train the model. Similarly, the process is repeated until n steps or the desired number of operations.

Both the above two cross-validation techniques are the types of exhaustive cross-validation. Exhaustive cross-validation methods are cross-validation methods that learn and test in all possible ways. They have the same pros and cons discussed below:

Pros:

1. Simple, easy to understand, and implement.

Cons:

1. The model may lead to a low bias.
2. The computation time required is high.

3. Holdout cross-validation: The holdout technique is an exhaustive cross-validation method, that randomly splits the dataset into train and test data depending on data analysis.

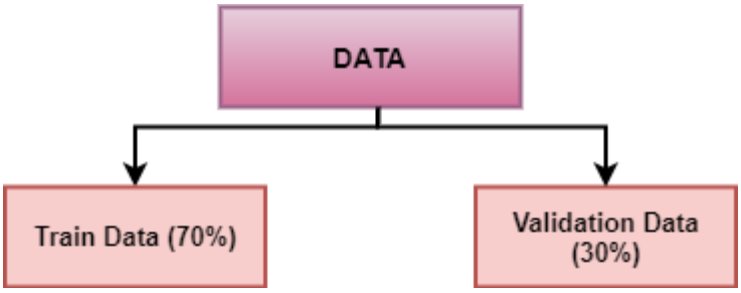


Fig. 70:30 split of Data into training and validation data respectively

In the case of holdout cross-validation, the dataset is randomly split into training and validation data. Generally, the split of training data is more than test data. The training data is used to induce the model and validation data is evaluates the performance of the model.

The more data is used to train the model, the better the model is. For the holdout cross-validation method, a good amount of data is isolated from training.

Pros:

1. Same as previous.

Cons:

1. Not suitable for an imbalanced dataset.
2. A lot of data is isolated from training the model.

4. k-fold cross-validation:

In k-fold cross-validation, the original dataset is equally partitioned into k subparts or folds. Out of the k-folds or groups, for each iteration, one group is selected as validation data, and the remaining (k-1) groups are selected as training data.

Fig. k-fold cross-validation

The process is repeated for k times until each group is treated as validation and remaining as training data.

The final accuracy of the model is computed by taking the mean accuracy of the k-models validation data.

$$\mathrm{acc_{cv}} = \sum_{i=1}^{k} \frac{\mathrm{acc_i}}{k}$$

LOOCV is a variant of k-fold cross-validation where k=n.

Pros:

1. The model has low bias

2. Low time complexity

3. The entire dataset is utilized for both training and validation.

Cons:

1. Not suitable for an imbalanced dataset.

Note:- I have used diabetes.csv dataset which contains 768 observations and 9 variables, as described below:

1. pregnancies - Number of times pregnant.
2. glucose - Plasma glucose concentration.
3. diastolic - Diastolic blood pressure (mm Hg).
4. triceps - Skinfold thickness (mm).
5. insulin - Hour serum insulin (mu U/ml).
6. bmi - BMI (weight in kg/height in m).
7. dpf - Diabetes pedigree function.
8. age - Age in years.
9. diabetes - "1" represents the presence of diabetes while "0" represents the absence of it. This is the target variable.

CODE (in python):-

```
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn

# Import necessary modules
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold
from sklearn.model_selection import LeaveOneOut
from sklearn.model_selection import LeavePOut
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import confusion_matrix
dat = pd.read_csv('diabetes.csv')
print(dat.shape)
dat.describe().transpose()
x1 = dat.drop('class', axis=1).values
y1 = dat['class'].values
# Evaluate using a train and a test set
# Holdout Validation Approach - Train and Test Set Split
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(x1, y1, test_size=0.30, random_state=100)
model = LogisticRegression()
```

```
model.fit(X_train, Y_train)
result = model.score(X_test, Y_test)
print("Accuracy: %.2f%%" % (result*100.0))

#K-fold Cross-Validation
kfold = model_selection.KFold(n_splits=10, random_state=100)
model_kfold = LogisticRegression()
results_kfold = model_selection.cross_val_score(model_kfold, x1, y1, cv=kfold)
print("Accuracy: %.2f%%" % (results_kfold.mean()*100.0))
y_pred=model.predict(X_test)
cm=confusion_matrix(Y_test,y_pred)
print("Confusion matrix is:\n",cm)

#Leave One Out Cross-Validation (LOOCV)
loocv = model_selection.LeaveOneOut()
model_loocv = LogisticRegression()
results_loocv = model_selection.cross_val_score(model_loocv, x1, y1, cv=loocv)
print("Accuracy: %.2f%%" % (results_loocv.mean()*100.0))
```

Output:-

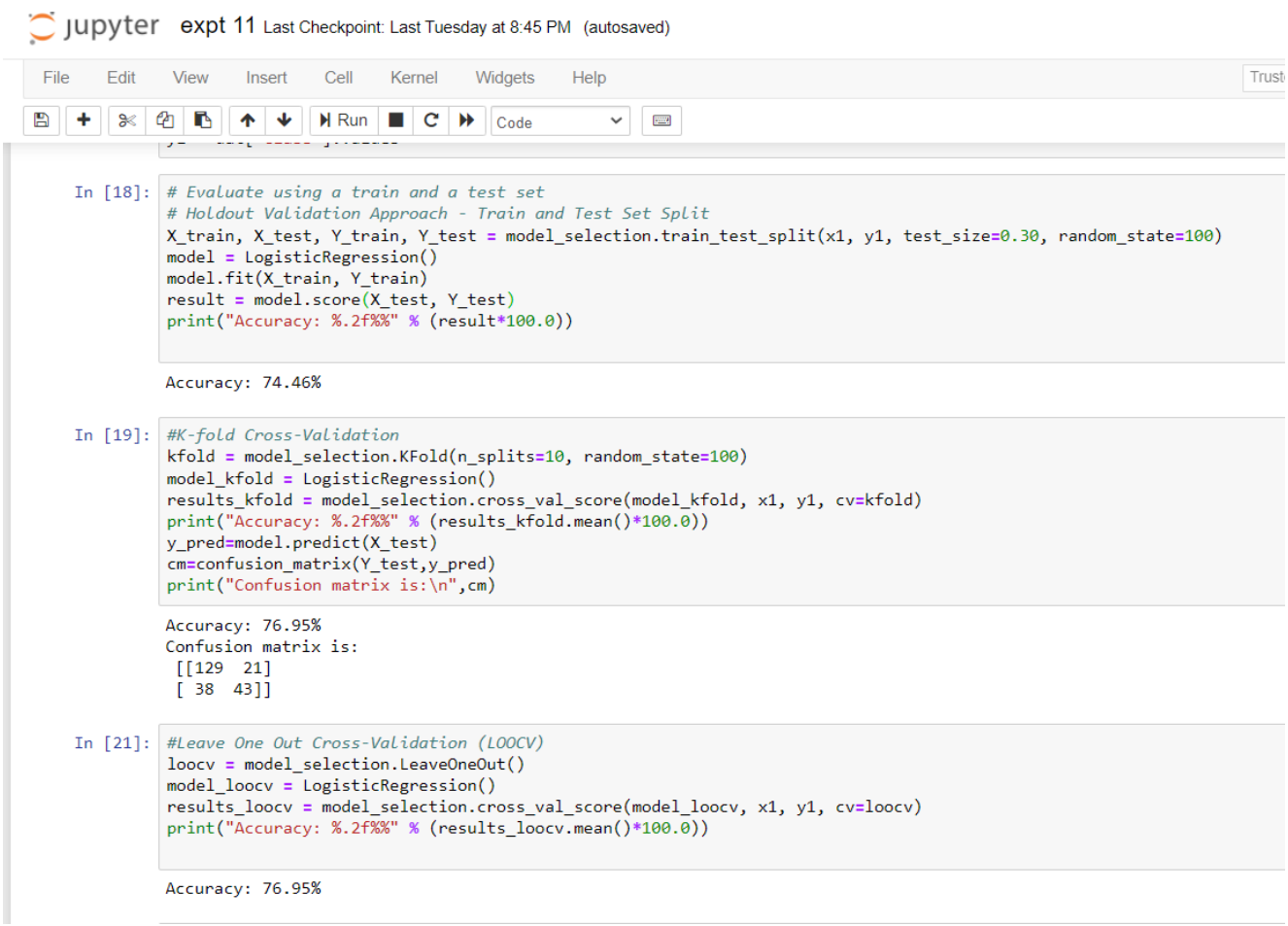
In [13]:
```
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn

# Import necessary modules
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold
from sklearn.model_selection import LeaveOneOut
from sklearn.model_selection import LeavePOut
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import confusion_matrix
dat = pd.read_csv('diabetes.csv')
print(dat.shape)
dat.describe().transpose()
```

```
(768, 9)
```

Out[13]:

	count	mean	std	min	25%	50%	75%	max
preg	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
plas	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
pres	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
skin	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
insu	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
mass	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
pedi	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00



Learning from experiment:- We have successfully learned about 10-cross validation, hold out validation, leave one out cross-validation techniques and its pros and cons. We have also successful in using this techniques to analyze a given dataset.

Empirical Software Engineering LAB – A1 G2
EXPERIMENT 4

Experiment Objective:- 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

Introduction:-

DATASET USED:

- KC2 Dataset in .arff format.
- Author: Mike Chapman, NASA
- Link to dataset: 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 McCabe's line count of code
2. v(g) : numeric McCabe "cyclomatic complexity"
3. ev(g) : numeric McCabe "essential complexity"
4. iv(g) : numeric McCabe "design complexity"
5. n: numeric Halstead total operators + operands
6. v: numeric Halstead "volume"
7. l: numeric Halstead "program length"
8. d: numeric Halstead "difficulty"
9. i: numeric Halstead "intelligence"
10. e: numeric Halstead "effort"
11. b: numeric Halstead
12. t: numeric Halstead's time estimator
13. lOCode : numeric % Halstead's line count
14. lOComment : numeric % Halstead's count of lines of comments
15. lOBlank : numeric % Halstead's count of blank lines
16. lOCodeAndComment: numeric
17. uniq_Op : numeric unique operators
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

Procedure:

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

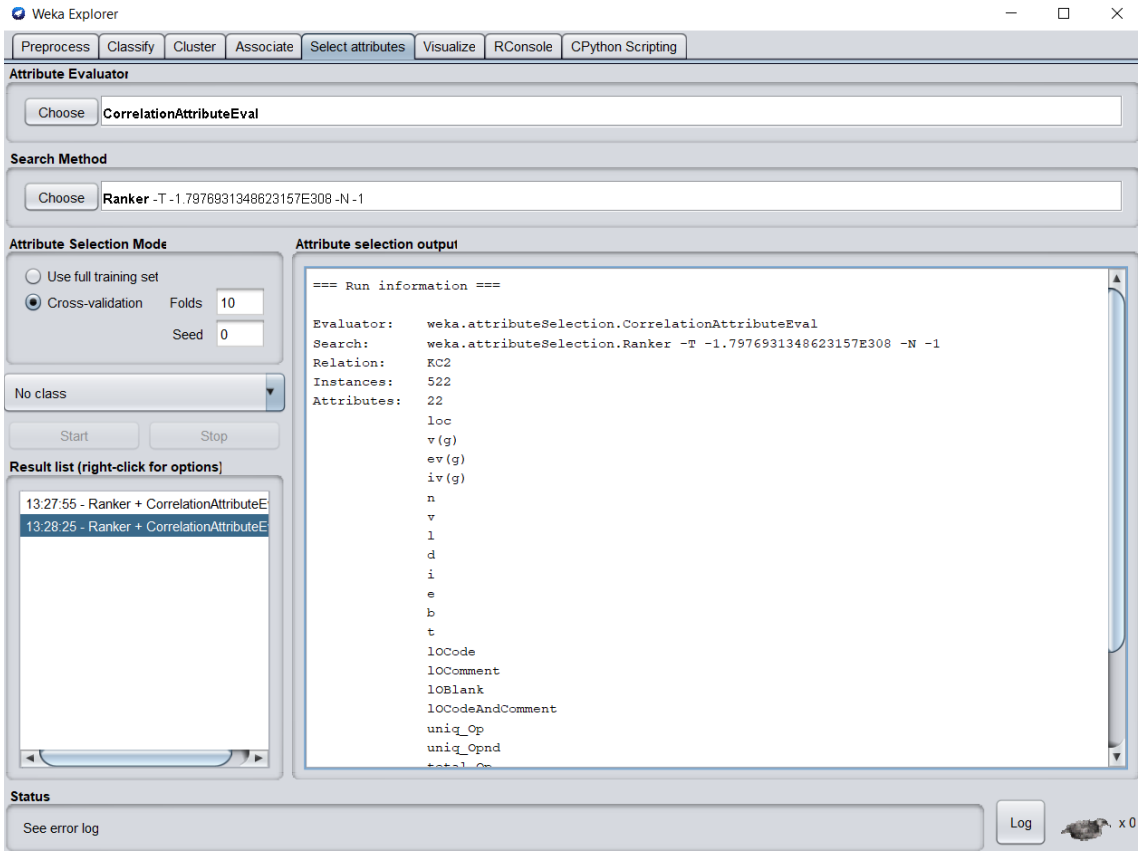
Feature selection is divided into two parts:

- Attribute Evaluator
- Search Method.

Result:-

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

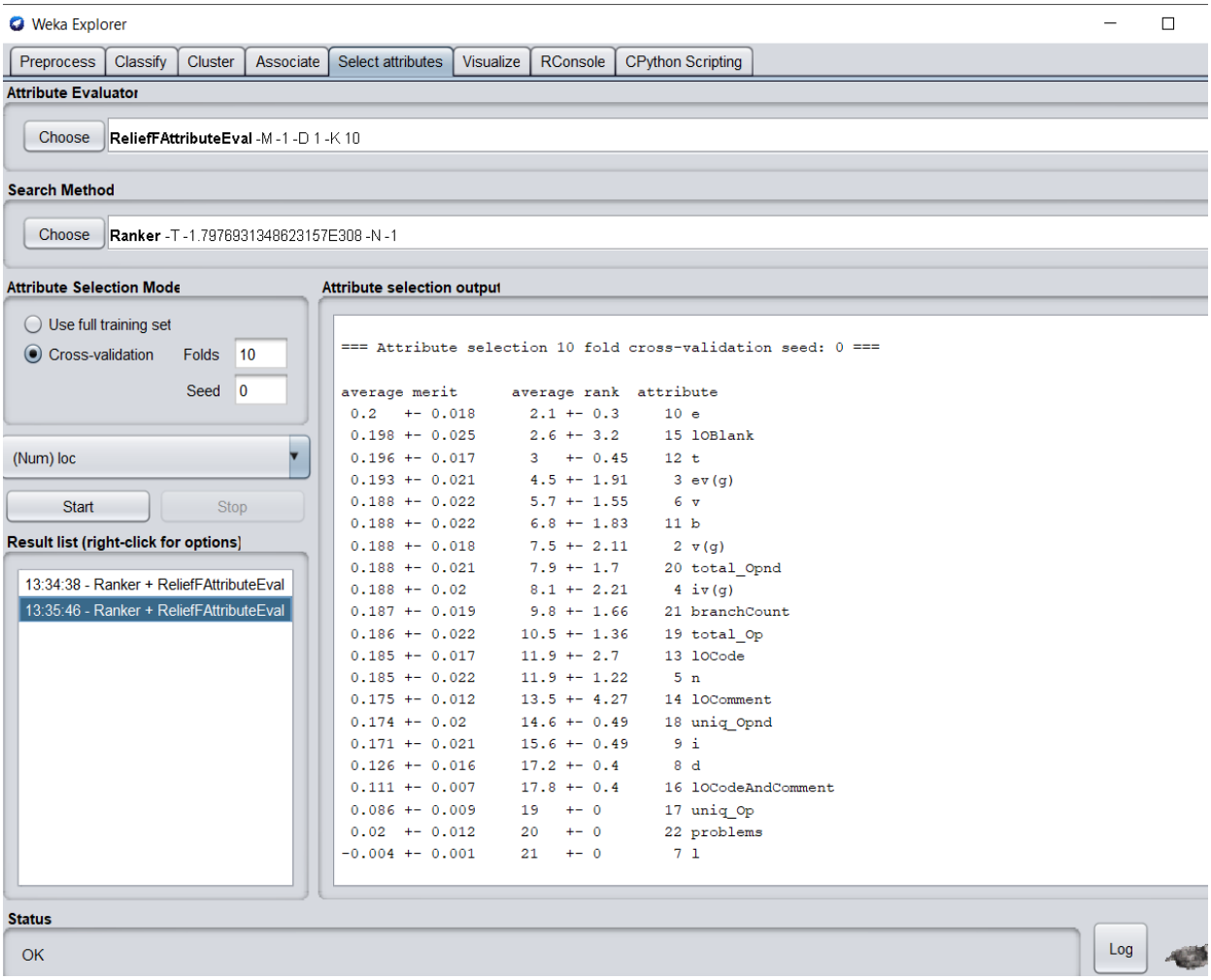


b) Relief Attribute feature evaluation:

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

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.



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).

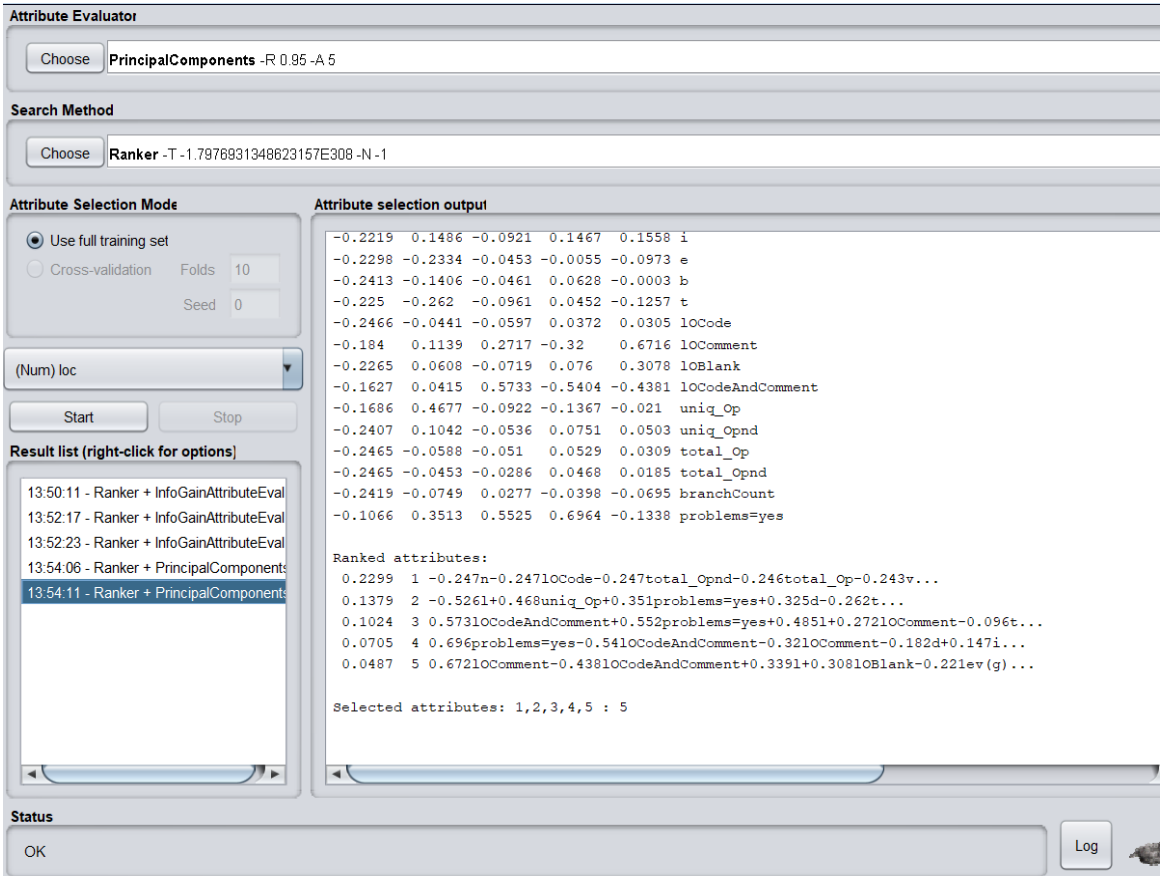
Weka supports feature selection via information gain using the InfoGainAttributeEval Attribute Evaluator. Like the correlation technique above, the Ranker Search Method must be used.

```
=== 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.



Learning from experiment:- We have successfully learned about WEKA and Correlation based feature evaluation, Relief Attribute feature evaluation, Information gain feature evaluation and Principle Component. We successfully have used the select attributes feature of WEKA for validating a dataset using 10-cross validation.

Empirical Software Engineering LAB – A1 G2
EXPERIMENT 5

- ASHISH KUMAR
- 2K18/SE/041

Experiment Objective:- 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 hypothesos for the followings 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.

Introduction:- 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.

Result:-

For John,
Size = 3000 LOC
Effort = 200 person-hours Defect = 6
Productivity = size/effort = 3000/200 = 15 LOC/person-hours
Quality = defect/size = 6 / 3000 = 0.02 defect/size

For Mike,
Size = 2000 LOC
Effort = 100 person-hours Defect = 5
Productivity = size/effort = 2000/100 = 20 LOC/person-hours
Quality = defect/size = 5 / 2000 = 0.025 defect/sizes

Mike implemented Car_Loan_Module having 4000 uncommented SLOC and took 200 person-hours of effort with the final module having 5 defects.
John implemented House_Loan_Module having 5000 uncommented SLOC and took 300 person-hours of effort with the final module having 6 defects.

(a) The quality of the code can be expressed in terms of defect density i.e. number of defects per lines of code.
Mike's code's defect density = 5/4000 = 0.00125 defects/SLOC
John's defect density = 6/5000 = 0.00120 defects/SLOC
The higher the defect density, the lower is the quality of the code. S0, John's quality is better than Mike's. Hence the null hypothesis is correct.

(b) Productivity = Size/Effort
Mike's productivity = 4000/200 = 20 SLOC/person-hours
John's Productivity = 5000/300 =16.667 SLOC/person-hours
John's productivity is less than Mike's
Hence the null hypothesis is incorrect.

(c) John introduced 6 defects while Mike introduced 5. Clearly, John introduced more defects than Mike. Hence the null hypothesis is correct.

(d) John's effort is 300 person-hours, while Mike's effort is 200 person-hours. Clearly, John's effort is more than Mike's. Hence the null hypothesis is correct.

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

Empirical Software Engineering LAB – A1 G2
EXPERIMENT 6

- ASHISH KUMAR
- 2K18/SE/041

Experiment Objective:- 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

Introduction:- A statistical hypothesis is an assumption made by the researcher about the data of the population collected for any experiment. It is not mandatory for this assumption to be true every time. Hypothesis testing, in a way, is a formal process of validating the hypothesis made by the researcher.
In order to validate a hypothesis, it will consider the entire population into account. However, this is not possible practically. Thus, to validate a hypothesis, it will use random samples from a population. On the basis of the result from testing over the sample data, it either selects or rejects the hypothesis. Statistical Hypothesis Testing can be categorized into two types as below:

- **Null Hypothesis** – Hypothesis testing is carried out in order to test the validity of a claim or assumption that is made about the larger population. This claim that involves attributes to the trial is known as the Null Hypothesis. The null hypothesis testing is denoted by H_0.
- **Alternative Hypothesis** – An alternative hypothesis would be considered valid if the null hypothesis is fallacious. The evidence that is present in the trial is basically the data and the statistical computations that accompany it. The alternative hypothesis testing is denoted by H_1or H_a.

Procedure:- Statisticians use hypothesis testing to formally check whether the hypothesis is accepted or rejected. Hypothesis testing is conducted in the following manner:
1. **State the Hypotheses** – Stating the null and alternative hypotheses. 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.

2. **Formulate an Analysis Plan** – The formulation of an analysis plan is a crucial step in this stage. 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.

3. **Analyze Sample Data** – Calculation and interpretation of the test statistic, as described 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 statistic 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.

4. **Interpret Results** – Application of the decision rule described in the analysis plan. Hypothesis testing ultimately uses a p-value to weigh the strength of the evidence or in other words what the data are about the population. The p-value ranges between 0 and 1. It can be interpreted in the following way:

- A small p-value (typically ≤ 0.05) indicates strong evidence against the null hypothesis, so you reject it.
- A large p-value (> 0.05) indicates weak evidence against the null hypothesis, so you fail to reject it.
- A p-value very close to the cutoff (0.05) is considered to be marginal and could go either way.

Decision Errors in R

The two types of error that can occur from the hypothesis testing:

- Type I Error – Type I error occurs when the researcher rejects a null hypothesis when it is true. The term significance level is used to express the probability of Type I error while testing the hypothesis. The significance level is represented by the symbol α *(alpha).*
- Type II Error – Accepting a false null hypothesis H0 is referred to as the Type II error. The term power of the test is used to express the probability of Type II error while testing hypothesis. The power of the test is represented by the symbol β *(beta).*

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.)

Result:-

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

$$\text{Null hypothesis: } \mu_1 - \mu_2 = 0$$

$$\text{Alternative hypothesis: } \mu_1 - \mu_2 \neq 0$$

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.

2. **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.

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

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

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

$$SE = \text{sqrt}(12.33) = 3.51$$

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

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

$$DF = (3.33 + 9)^2 / \{ [(3.33)^2 / (29)] + [(9)^2 / (24)] \} = 152.03 / (0.382 + 3.375) = 152.03/3.757 = 40.47$$

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

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 from experiment:- This experiment gives insights into the use of hypothetical testing on real-life examples. In this experiment we learnt how to validate whether the proposed hypothesis is correct or not using statistical hypothesis.

Empirical Software Engineering LAB – A1 G2
EXPERIMENT 7

- ASHISH KUMAR
- 2K18/SE/041

Experiment Objective:- Consider defect dataset and implement following statistical test using SPSS tool.

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

Introduction:-

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

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

t = Student's t-test
m = mean
μ = theoretical value
s = standard deviation
n = variable set size

b) Chi-Square Test: A chi-squared test, also written as χ^2 test, is astatistical 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.

c) Wilcoxin 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 means ranks differ.

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

χ^2 = chi squared
O_i = observed value
E_i = expected value

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

W	= test statistic
N_r	= sample size, excluding pairs where x1 = x2
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 shit, 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.

Result:-

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

Paired Samples Effect Sizes

			Standardizer[a]	Point Estimate	95% Confidence Interval Lower	Upper
Pair 1	cbo - totalMethods	Cohen's d	45.095	-.133	-.158	-.108
		Hedges' correction	45.098	-.133	-.158	-.108

a. The denominator used in estimating the effect sizes.
Cohen's d uses the sample standard deviation of the mean difference.
Hedges' correction uses the sample standard deviation of the mean difference, plus a correction factor.

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		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error 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[a]	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	Percentiles 25th	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

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

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

Test Statistics

	cbo	wmc	dit	rfc
Chi-Square	18016.244[a]	14453.945[b]	79129.146[c]	12848.642[d]
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

Ranks

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

	totalMethods - cbo	dit - wmc	lcom - rfc
Z	-3.292[b]	-66.949[c]	-23.084[b]
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[a]

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

a. Friedman Test

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

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

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

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[a,b]

	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

Learning from experiment:- Through this experiment I learned how to perform hypotheses tests such as parametric test and non parametric test which includes kruskal Wallis, paired t-test, chi-square test, wilcoxin signed rank test. I also explore the SPSS tool (by IBM) to evaluatethese tests on Software defect prediction dataset.