Thesis no: MSCS-2018-01

Pattern Recognition applied to continuous integration systems

Course Code: DV2572 Master Thesis in Computer Science

Shivakanthreddy Vangala

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

Contact Information:
Author(s):
Shivakanthreddy Vangala
E-mail: shva16@student.bth.se

External advisor:
Rikard Ljungstrand, Software Developer, Ericsson
rikard.ljungstrand@ericsson.com
Per.Karlsson, Software Developer, Ericsson
Per.karlsson@ericsson.com

University advisor:
Michael Unterkalmsteiner, Professor in software engineering department
Department of software engineering

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Context: This thesis focuses on regression testing in the continuous integration environment which is integration testing that ensures that changes made in the new development code to the software product do not introduce new faults to the software product. Continuous integration is software development practice which integrates all development, testing, and deployment activities. In continuous integration, regression testing is done by manually selecting and prioritizing testcases from a larger set of testcases. The main challenge faced using manual testcases selection and prioritization is in some cases where needed testcases are ignored in subset of selected testcases because testers didn't include them manually while designing hourly cycle regression test suite for particular feature development in product. So, Ericsson, the company in which environment this thesis is conducted, aims at improving their testcase selection and prioritization in regression testing using pattern recognition.

Objective s: This thesis study suggests prediction models using pattern recognition algorithms for predicting future testcases failures using historical data. This helps to improve the present quality of continuous integration environment by selecting appropriate subset of testcases from larger set of testcases for regression testing. There exist several candidate pattern recognition algorithms that are promising for predicting testcase failures. Based on the characteristics of the data collected at Ericsson, suitable pattern recognition algorithms are selected and predictive models are built. Finally, two predictive models are evaluated and the best performing model is integrated into the continuous integration system.

Methods: Experiment research method is chosen for this research because discovery of cause and effect relationships between dependent and independent variables can be used for the evaluation of the predictive model. The experiment is conducted in RStudio, which facilitates to train the predictive models using continuous integration historical data. The predictive ability of the algorithms is evaluated using prediction accuracy evaluation metrics.

Results: After implementing two predictive models (neural networks & k-nearest means) using continuous integration data, neural networks achieved a prediction accuracy of 75.3%, k-nearest neighbor gave result 67.75%.

Conclusions: This research investigated the feasibility of an adaptive and self-learning test machinery by pattern recognition in continuous integration environment to improve testcase selection and prioritization in regression testing. Neural networks have proved effective capability of predicting failure testcase by 75.3% over the k-nearest neighbors. Predictive model can only make continuous integration efficient only if it has 100% prediction capability, the prediction capability of the 75.3% will not make continuous integration system more efficient than present static testcase selection and prioritization as it has deficiency of lacking prediction 25%. So, this research can only conclude that neural networks at present has 75.3% prediction capability but in future when data availability is more, this may reach to 100% predictive capability. The present Ericsson continuous integration system needs to improve its data storage for historical data at present it can only store 30 days of historical data. The predictive models require large data to give good prediction. To support continuous integration at present Ericsson is using jenkins automation server, there are other automation servers like Team city, Travis CI, Go CD, Circle CI which can store data more than 30 days using them will mitigate the problem of data storage.

Keywords: continuous integration, pattern recognition, artificial neural networks, k-nearest neighbors

Acknowledgements

I would like to thank Ericsson Gothenburg for offering this thesis and for huge support during thesis. Special Thanks to the MINI-LINK team especially Rikard Ljungstrand, Per Karlsson for their guidance and industrial support during thesis. Special thanks for Dafina Ivanovska, Line manager in Ericsson for her support during thesis.

Additionally, I would like to thank my university supervisor Michael Unterkalmsteiner for his feedback on my work and useful discussions of the research. At last, I want thank my parents for their support without them, I would have not completed my masters thesis.

List of figures

List of tables

Contents

1.Introduction

Continuous Integration is a software development process where developers commit their code intermittently leading to multiple integrations in a day [3]. Work performed by each developer on specific software feature is known as integration. Every integration is verified by an automated build testing which is combined with testcases to identify the integration errors expeditiously [1]. The prime intention of practicing continuous integration is to find errors in the early development process and give quick feedback for the developers on their development tasks.

Continuous Integration uses a regression test suite for testing tasks which helps at enhancing or refactoring existing development code base [2]. A regression test suite incorporates many testcases. The size and number of testcases depends on the development product [4], so large set of testcases will implement regression testing against integration. This takes more time and operational cost will increment and ongoing development of the product needs to be adjourned for the results of the regression testing. To avoid these drawbacks, testcase selection and prioritization approaches are introduced into the regression testing [15], Test case selection and prioritization deals with selecting a subset of testcases from large set of testcases with respect to feature software development in the regression test suite.

This research is conducted in collaboration with Ericsson in their continuous integration environment. In Ericsson, Continuous integration environment is used for the software development process of the MINI-LINK(pt4351) product. Regression testing is performed in three stages in product development with test case selection and prioritization and without testcase selection and prioritization. Regression testing with testcase selection and prioritization is known as hourly cycle test suite whereas regression testing without testcase selection and prioritization is known as daily cycle test suite. Hourly cycle test suite is the subset of the daily cycle test suite.

In hourly cycle test suite, testcase selection and prioritization is done with respect to latest integration development in every two hours. Daily cycle test suite without any testcase selection and prioritization do testing for final integration of the day. If there is no failure testcase in hourly cycle regression testing, integration will be merged into the product. After the final integration of the day, daily cycle regression testing is performed on all integrations of the product.

While performing daily regression testing on final integration, surprisingly Ericsson observed new testcases are failed in daily cycle regression testing, those testcases are supposed to be selected by manual testcase selection and prioritization for hourly cycle regression testing so that errors will be known early in hourly cycle regression testing itself but manual static testcase selection and prioritization didn’t selected them in hourly cycle regression testing. So, Ericsson came to know that the present manual testcase selection and prioritization is not sufficient for finding appropriate subset of testcases for hourly cycle regression testing.

So now Ericsson wants to change its static manual testcase selection and prioritization into dynamic predictive testcase selection and prioritization in regression testing. To accomplish this pattern recognition algorithms are used to add prediction for finding appropriate subset of testcases for hourly cycle regression testing. This will help Ericsson MINI-LINK product to have better testcase selection and prioritization in regression testing. There are many predictive algorithms but in this research only pattern recognition algorithms are selected because pattern recognition will find regularities in the data that helps to predict the future failed testcase using data regularities from historical testcases results data so those testcases can be selected and prioritized in hourly cycle regression testing.

Pattern recognition is one of the disciplines in machine learning that spotlights on the acknowledgements of patterns and uniformity in data. Pattern recognition is widely used in the areas of statistics, engineering, artificial intelligence among others [26]. Pattern recognition is classified on basis of the learning practice of training data. If the training data is labeled with output instances than the learning practice is called as supervised learning. If the training data is not labeled with output instances than it is called as unsupervised learning [33].

In pattern recognition they are many algorithms like Discriminant analysis, Support vector machine, Artificial neural network, Decision trees, Naive Bayes, k-nearest neighbor after extensive studying, examining and analyzing previous research papers[24][33][36][41] for all algorithms in implementation, data overfitting problems and evaluation of the algorithms only artificial neural network and k-nearest neighbor algorithms are selected as these algorithms are already implemented successfully in other pattern recognition non-linear classification problems. Implementing all algorithms increments the operational cost of the research, so experiment is only conducted on selected algorithms.

Experiment research method is performed to evaluate the predictive models according to the selected pattern recognition algorithms. evaluation of the models is done with machine learning evaluation metrics to know the prediction competence of the predictive models. If any predictive model achieves 100% predictive capability than that model is perfect fit for the continuous integration environment and it will improve testcase selection and prioritization in regression testing.

1.1 Aim:

The aim of thesis is to investigate the feasibility of an adaptive and self-learning test machinery by pattern recognition algorithms to predict future failure testcases for improving testcase selection and prioritization in continuous integration environment.

1.2 Objectives:

1. Design a predictive model based on pattern recognition algorithms for CI machinery.

2. The pattern recognition CI machinery should predict failure testcase based on the historical data. This failure testcases will help to improve test selection and prioritization.
3.Develop a way to evaluate the prediction competence of the model in finding failure testcase.

1.3 Research Questions

Research Question 1:

How can Pattern recognition be implemented in continuous integration environment?

Motivation:

Motivation behind this research question is knowing how can self-learning test machinery model will be built based on pattern recognition algorithms in continuous integration environment for testcase selection and prioritization, at first, Research needs to find which data can be used for training of the algorithm for prediction. Finding this research question gives answer to know whether available data is suitable or not suitable for implementation of pattern recognition algorithms in continuous integration for testcase selection and prioritization in regression testing.

Research Question 2:

Which pattern recognition model has the best prediction competence for test case selection and prioritization in regression testing for continuous integration?

Motivation:

After implementation of the pattern recognition models in the continuous integration, so motivation in this research question is knowing the prediction competence of both pattern recognition models on continuous integration environment. This can be known by evaluating both predictive models by pattern recognition evaluation metrics.

2.Background

2.1 Continuous integration

Continuous integration system is treated as one of the leading concepts in automated software industry which assists software development with testing environment [1]. Continuous integration adopts excess testing cycles for software development work station to continuously run regression tests in the background to help developers to know errors in their development code regularly [2]. Many software enterprises engaging their software product development with continuous integration as it prevents the integration problems and it provides expeditious feedback to developers on errors in their development code when they are negligently introduced amid development stage [2].

In continuous integration system, testing is done with the help of combination of automated unit tests [7]. Combination of automated unit tests are performed with practices of test-driven development [7] which is test-case selection and prioritization. Combination of automated unit tests will run all unit tests in the development local environment and verify development codes before integrating to the main stream product.

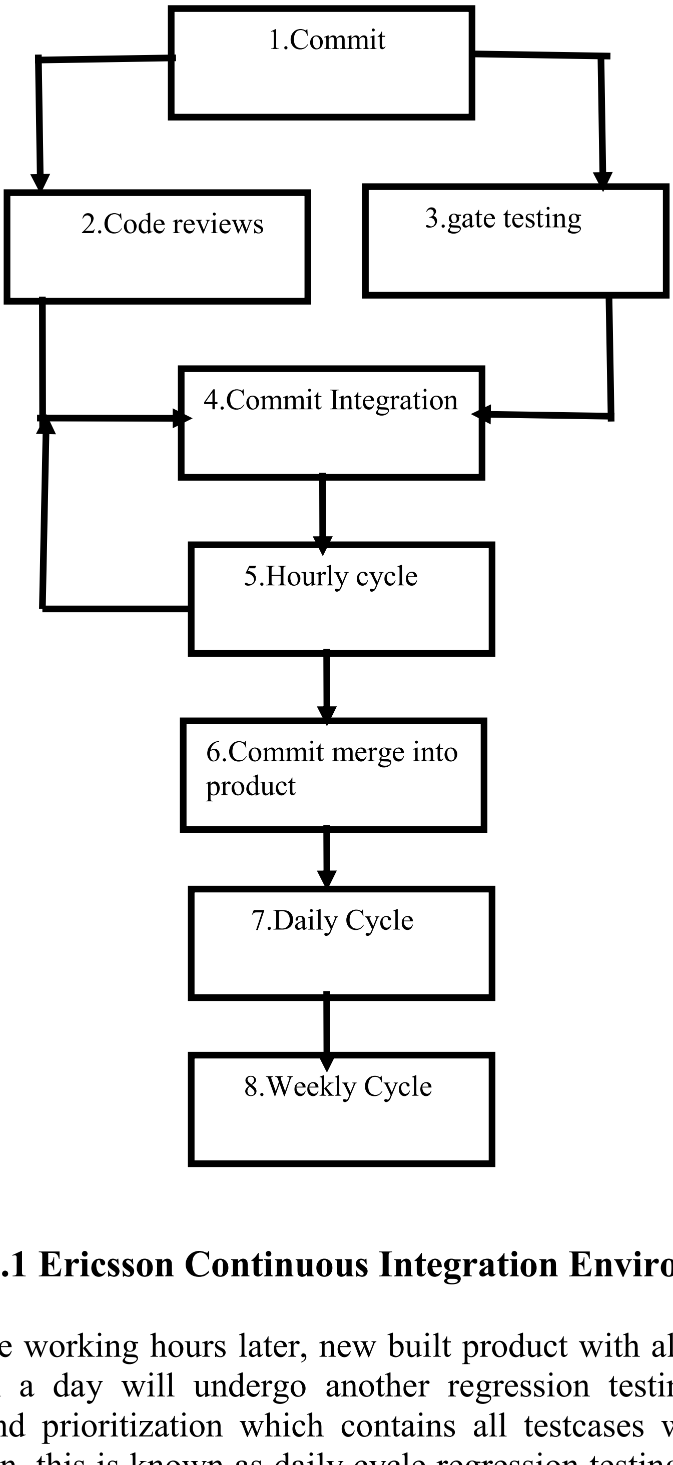
Software enterprises using continuous integration not only aid with automated unit tests. continuous integration also helps to check the quality of the product by adopting a built server to enforce repeated process of the handling quality control mechanisms. In development stage small fragments of code practiced intermittently so quality problems can be resolved effortlessly [9].

In this Research, Experiment is performed to the continuous integration environment in Ericsson for MINI-LINK(pt4351) product. Continuous Integration environment in Ericsson consists of three stages they are Developing stage, building stage, and testing stage [7].

In the Developing stage, (from Figure 2.1.1) Developers develop the software code related to the requirements of the product. Once code is developed, it will be pushed as commits (from Figure 2.1.1 stage 1) in Gerrit. Gerrit is a web based code review system which provides repository management for the legacy product code and It also facilitates code reviews in-between developers. In the Gerrit other developers will review the code (from Figure 2.1.1 stage 2). if everyone accepts the developed code by checking its functionality than it will be merged into the product. Concurrently unit testing will also perform for the developed code which is known as gate testing (from Figure 2.1.1 stage 3) for checking syntax and logical errors. The reason for doing this review management and unit testing is to avoid the code breakdown after merging code into product.

In the Building stage, Merged code will get integrated with previous integration (from Figure 2.1.1 stage 4) here previous integration is last developed code integration into product. Merged code gets integrated with previous integration and then new built is formed.

Testing stage will be performed after new built is created. Hourly cycle regression testing (from Figure 2.1.1 stage 5) will be performed with testcase selection and prioritization, where certain number of testcases are selected from the large set of testcases and made hourly cycle regression test suite according to the feature development code. The result of the hourly cycle regression testing will be known to developers after two or three hours, if there is any failed testcase in the hourly cycle regression testing than development code will be reverted to the code review again (from Figure 2.1.1 stage 5) otherwise it will be continued to be part of new built (from Figure 2.1.1 stage 6)



2.1.1 Ericsson Continuous Integration Environment

After twelve working hours later, new built product with all development code commits in a day will undergo another regression testing without testcase selection and prioritization which contains all testcases without selection or prioritization, this is known as daily cycle regression testing (from Figure 2.1.1 stage 7). The result will be known only after another twelve hours. This is because it must undergo lot of regression testing with more testcases than hourly cycle.

There is also another regression testing that is weekly cycle regression testing (from Figure 2.1.1 stage 8) which is performed on Saturday every week for all committed codes in a week which is new built. In this research we are not considering weekly testing as research is more focused on day cycle regression testing and hourly cycle regression testing.

2.2 Pattern recognition

Pattern recognition is one of the most important disciplines in machine learning that spotlights on the acknowledgements of patterns and regularities in data [25]. It is an assignment of the labelled output value to the instance input value. Pattern recognition is widely used in the areas of statistics, engineering, artificial intelligence among others [27].

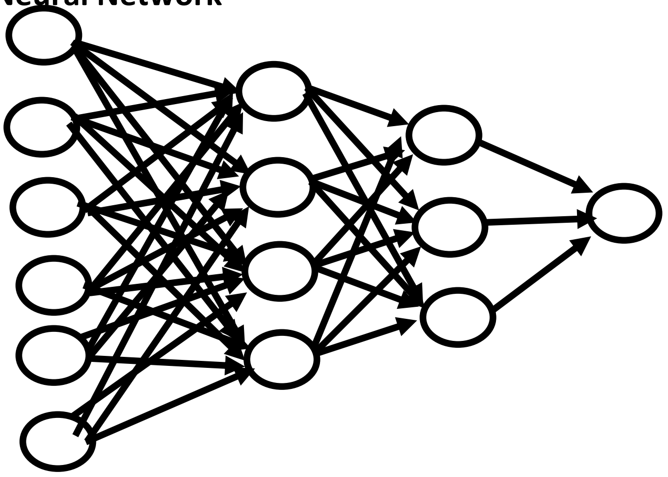
Pattern recognition is categorized on basis of the learning procedure of training data [28]. If the training data is labeled with output instances than the learning procedure is called as supervised learning. If the training data is not labeled with output instances than it is called as unsupervised learning [28].

In this research as output data is available so supervised pattern recognition is used to create a predictive model. Pattern recognition is mainly used for solving classification and regression problems. The main goal of classification is predicting class in-between the classes [31], whereas in regression the goal is predicting discrete and continuous value [31], In this research predictive model is built to predict whether the testcase will fail or not, This perfectly suits for the binary classification, so only classification techniques are studied and implemented in this research. Pattern classification is the organization of the patterns into groups of patterns having shared same set of properties [32]. Pattern classification has three different approaches to solve classification those are statistical pattern recognition approach, Neural network approach, Structural or syntactic approach.

In statistical approach it is based on the fundamental statistical model of pattern and pattern classes [33]. In neural network approach, classifier is represented as a network of cells modeling neurons of the human brain [33]. In structural or syntactic approach, pattern classes are represented by means of formal structures as grammars, automata, and strings. In this research, predictive classifier model needs to be made for continuous integration so only statistical pattern recognition approach and neural network approach is selected as syntactic pattern recognition approach is used only to create formal structures where serial recurrent input will give discrete and continuous output by making patterns. Syntactic pattern recognition perfectly suits for unsupervised learning but in some cases, it is also used for supervised learning. Due to time constraint syntactic pattern recognition is ignored in this research.

For classification problems statistical pattern recognition has algorithms like Discriminate analysis, Decision trees, Support vector machine, K-nearest neighbor, Naïve Bayes among others. From statistical pattern recognition only, k-nearest neighbor is chosen as previous studies [24] indicate that it can handle over fitting issues effectively, Overfitting is where predictive model learns only specific pattern and noise from the training data leaving the rest of the data [24]. From Neural Networks Artificial neural networks with backpropagation algorithm is selected as back propagation algorithm facilitates with error correction which will give high performance than perceptron, Hopfield Network, Radial Basis Function Network [32]. Using Artificial neural network and k-nearest neighbor two predictive models are constructed.

Artificial Neural Network



2.2.1 Artificial Neural Network

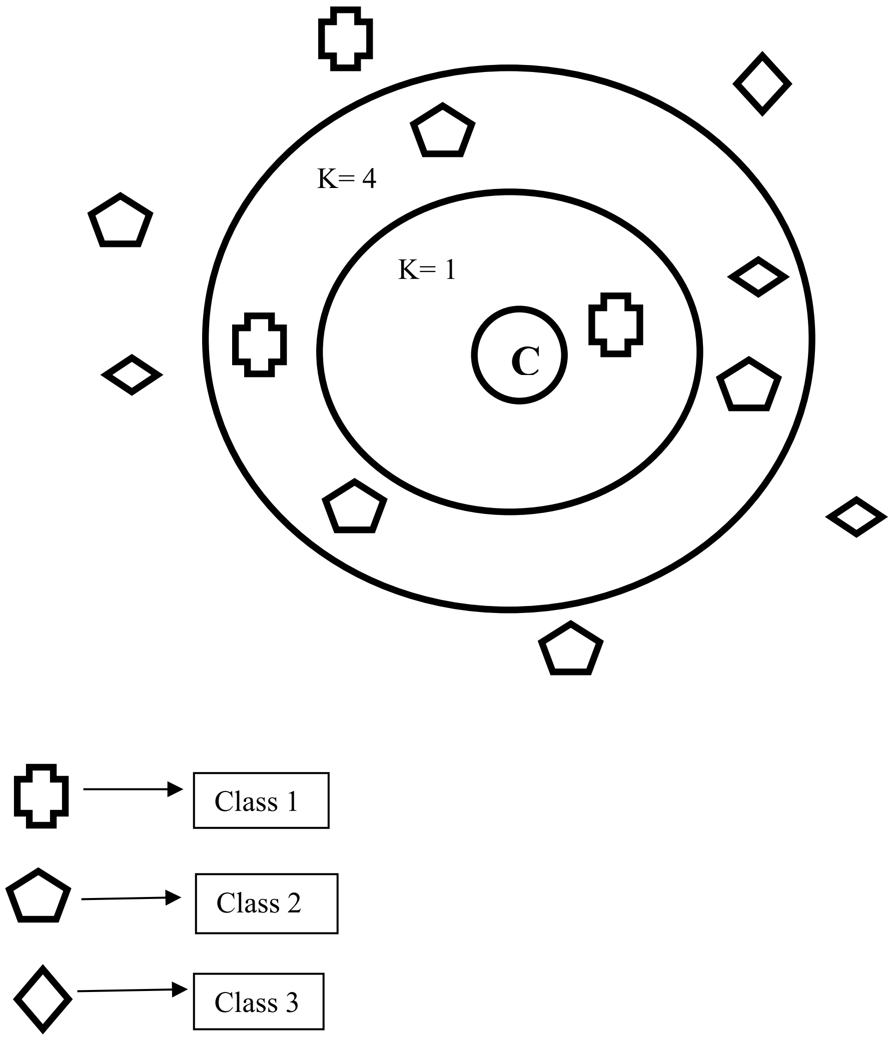
The brain's biological neural network inspires artificial Neural Network. The neuron is the core computational unit of the brain and in oversimplified terms it is capable of processing input signals from other neurons, and deciding on to produce an output based on these signals [36]. Artificial Neural Networks are designed to replicate like the biological neural networks. It has three types of layers input layer, output layer and hidden layer. In the above Diagram 2.2.1 first layer is input layer, second and third layer is hidden layers, final layer is output layer.

The first layer which takes input known as input layer, layers after input layer are called hidden layers, the final hidden layer is called output layer which gives output value [28]. Each layer is connected with simple processing elements or building blocks called neurons [29], where each neuron interacts using weighted connections. weight is used to calculate the signal traveling through the connection, finally output layer will make final alternations to the input and outputs the prediction using activation function [27].

The mapping function of a summed weighted input to the output of the neuron is carried by an activation function [27]. Activation function consists threshold at which neuron is activated and generates output signal [27]. The prediction accuracy of the network is usually calculated using error function or cost function. The difference between network output and expected output is known as error function or cost function [29]. In this research for training data back propagation is used. In backpropagation, error is backpropagated through the network, one layer at a time so that weights are updated according to the amount that they contributed to the error.

K-Nearest Neighbor
K-means nearest neighbor is a simple non-parametric instance-based supervised learning algorithm used for classification and regression [24]. This is also known as lazy learning because without any generalization or summarization, K-means algorithm learns training data and finds association of other training samples that are closet to a test sample and then it designate label to the test sample based on the preponderance of a certain class in the k closet neighborhood (refer Figure 2.2.2) [24]. KNN make predictions by calculating the similarity between an input sample and each training instance. The calculation is done using distance measure. Distance measure can be done by hamming distance, Manhattan distance, Makowski distance, Euclidean distance [24]. The value of the K can be found by algorithm tuning where different values need to implemented and best result will be selected as k value.

Below is the example for the k-nearest neighbor non-linear classification.



2.2.2 K-nearest Neighbor

In the above Figure 2.2.2 this is example for working of k-nearest neighbor in the figure where test sample (center circle) desired to be classified either class 1 or class 2 or class 3. If k= 1 test sample will be assigned to the class 1 as class 1 symbol is only present in the distance circle. If k= 4 test instance will be assigned to the class 2 because class 2 are more in number in the distance circle.

3.Related work

Research works on Testcase selection and prioritization in regression testing is mainly focused on improving present manual static testcase selection and prioritization than automating whole test suite using machine learning techniques

Research work made on Coverage based regression test case selection [54] and Directed test suite augmentation [56] in continuous integration mainly gave basic knowledge about the system environment and regression testing. Which has many similarities of Ericsson environment. In both the papers hybrid approach is compassed in the regression testing that is combining the selection of testcases based on historical code results. Which is very similar to our research but these are case study papers. These papers didn't give any indices of predictive model based on the pattern recognition techniques but this case study papers really helped in finding required data for data set creation from raw historical Ericsson data as they explained more about the files changing attributes through code changes.

Research paper on agile regression testing [55] presented two new techniques those are Weighted sprint testcases prioritization technique and Cluster based release test cases selection. Where Weighted sprint testcases prioritization technique is mainly depends on the importance of the requirements represented in the user stories of an agile development [55]. In cluster based release technique where it clusters the user stories according to the modules covered by each user story [55]. This paper is mainly focused on the agile development in the regression testing. This paper helps to know the different features and stakeholder's requirements.

Research paper on Failure history data-based testcase prioritization [57] for effective regression provided information about the usage of failed testcase history for prioritization and selection but this is implemented in tomcat server. Present Ericsson system uses Jenkins as an automation server which is mainly used for continuous delivery and deployment. Implementation of test case selection and prioritization in tomcat server helped to know server facilitation requirements for better data storage. Research works on testcase selection and prioritization using manual black-box system testing [64] where manual testing is performed in a black-box manner and rank testcases to prioritize, Prioritization of requirements for test(PORT) [64] where testcases are given rank and prioritized using the requirements of the test.

Research work made on the on continuous change impact Analysis process (CCIP) [5], where it considers of complexity of dependencies between code artifacts of various software development process to improve the effectiveness of automated tests. This research gave lot of information regarding data collection. They are also no possible hints of pattern recognition application in continuous integration. Books describing various pattern recognition models give introduction to pattern recognition algorithms and its application but gave no success in finding appropriate application related to automation of regression testing in continuous integration environment [41] [42] [43] [44] [45].

Other solved prediction problems using pattern recognition is studied and gained knowledge on application of pattern recognition. selection of algorithms is inspired from pattern recognition success in prediction of handwritten recognition and character recognition [28] [38]. After Continuous study of literature on topics related to continuous integration, pattern recognition, testcase selection and prioritization, Neural networks and K-nearest neighbor before and after experiment there are research works performed on the improving the present manual testcase selection and prioritization but as far as in this research study till now there is no identification of application of predictive algorithms in continuous integration but applications of pattern recognition algorithms for different research problems[38] and prediction of testcase failure without predictive algorithms[53][54][55][57][58][59][60][61] gave good reference and guidance while doing research. This research papers helped in looking more approaches while data collection, training data to the algorithm and designing the algorithm.

4. Research Methodology

To achieve answers for the research questions in this research, the research method used to perform this research is experiment. Experiment is one of the prevailing research method in the computer science. It is a systematic and scientific approach to research, which involves manipulation of independent variables and controls and measures any change in dependent variable [46]. Experiment method is selected because in this thesis research is more focused about analyzing the data and implementing a predictive model using historical data. To find best model in-between two predictive models, evaluation is must. The evaluation and impact of the predictive model can only be acknowledged if experiment is performed. Research cannot be conducted with case study or survey because with these implementation and evaluation of predictive model is not possible. In experiment dependent variable is performance of the predictive algorithm and independent variables is Dataset and algorithms.

4.1 Software Platform

In this research, Predictive model is conducted and evaluated in the RStudio Integrated development environment(IDE). RStudio is one of the platforms used for the statistical programming and applied machine learning. RStudio facilitates the implementation of different predictive algorithms with the help of third party libraries known as packages [68].

In the RStudio, R programming is used for the implementation of the predictive algorithms. R programming is used to analyze data, plot data, or build statistical model for the data. In this RStudio, statistical model for prediction of failed testcase is created and accuracy is also calculated [68].

4.2 Data collection

In this research, Experiment is performed with the historical data of daily test suite results for the continuous integration environment in Ericsson MINI-LINK product. Ericsson gave access to search and browse historical data of the MINI-LINK product. At present in the Ericsson database mini-link product historical data is stored only for the last 30 days, after that data will be get deleted and new data will be stored. The present Ericsson system uses Jenkins automation server for the continuous integration, Jenkins only stores 30 days data so Ericsson is having 30 days data. After reading literature, many pattern recognition books [42] [43] marked that data size has significant impact on the prediction capability of the model so, in order to have more data for experiment Data collection is done for two months that is 60 days, 60 days data is also not enough for creating efficient predictive model but data collection is only planned for 2 moths so only 60 days data is collected and used for the implementation of predictive model.

S.no	Data
1.	Number of changed files
2.	Files path
3.	Insertions made in files
4.	Deletions made in files
5.	Time and date details
6.	Testcases Results
7.	Ericsson Internal ID's
8.	Product client Information
9.	Internet protocols addresses

4.2.1 Historical data in Ericsson database

The historical data contains (see Table 4.2.1) number of changed files which gives information about the number of files changed due to commits made my developers. Files path gives information about the path to the changed files, Insertions made in files this gives information about the insertions made in the changed files, Deletions made in files this gives information about the deletions made in the changed files. Time and a date details give information about the product development date and time. Ericsson internal ids are developer's identity numbers(signums), Product client information is the Ericsson client's information for the MINI-LINK product, Intern protocols addresses gives networks information of Gothenburg, Budapest, New Delhi Ericsson offices.

4.3 Data preprocessing

Data preprocessing is one of the essential step in data mining to eliminate noisy data from the experiment. These is performed to avoid over fitting problem which trains predictive model only to learn specific pattern and noise from the training data leaving the rest of the data.

Data purifying or data cleaning is the way towards distinguishing and remedying inappropriate data from the dataset, It involves supplanting, altering, or erasing the unrequired data [66]. In the acquired historical data, It contains data related to product client's information, Ericsson internal Id, Internet protocol locations, product feature information and it updates. This data information is removed in our data acquisition as they will not show any impact on the result of the testcase and Ericsson also gave instructions not to use internal product client information as it would breach the trust of business ethics.

In the present historical data, MINI-LINK(pt4351) product has 108 testcases. Results of some testcases data is irrelevant as they don't show any impact on the failure of testcase. These testcases data is excluded from the training as it can create false prediction and affect performance of the model.

The following testcases data is excluded for training data.

- The testcases which are skipped in the regression testing is not considered for data training because these are skipped as they didn't fit to the feature requirements of the integration. In this research it is not considered because research goal is to find failed testcase, if skipped testcases is considered it will only affect the prediction accuracy and not deliver any appropriate prediction of failed testcase so this testcases are excluded. In product MINI-LINK there are 38 skipped testcases and their data is excluded from training.
- Testcases which are failed due to the hardware errors is not considered because industry doesn't have proper data regarding factors causing hardware errors so this data is not considered in this training data. There are 17 testcases which failed due to hardware errors, there are not considered in training data.
- While analyzing testcases results, there are some testcases which are always passed with every integration. This kind of testcases are not considered as they don't have any impact on failure of the testcase. There are 45 testcases which are passed always for every integration, so this testcases results are also not considered.

Finally, 8 testcases are left after exclusion for doing experiment.

After Data Processing, Remaining data available in historical data is our data set in this experiment. (see Table 4.2.1) S.no [1][2][3][4][6] are considered as data elements for the data set. These are also selected as features for training model because these are attributes affecting the result of the testcases made by the developer for product development. Input features are number of files, file path, insertions, and deletions. Output features are results of the testcases.

4.4 Experiment design

Each instance consists of testcases results for the developers changes on files through commits in one day. Each day eight testcases results will be obtained so eight data points are collected for one day like that for 60 days 480 data points will be obtained with the results of the testcases in which 354 are passed testcases data points and 126 failed testcases are collected for experiment.

Training and testing of model is done for each testcase separately. So that prediction capability of each testcase can be known and inclusion all testcases can cause overfitting problem where testcases results will be more than input features. This overfitted model will have impact on prediction and it will affect the performance of the prediction model. So, while training and testing only one testcase is considered.

Leave one out cross validation technique is used for training and testing of the data, which is one of the cross-validation method. Cross validation is a model validation method which evaluate predictive models by partitioning the original dataset into a training set to the train the model and a testing set to evaluate it [65]. It works by splitting the entire dataset into train leaving out one instance for test. That one instance is called fold. This is repeated so that each fold of the dataset is given a chance to be held back test set. This helps to increase the performance of the model and over fitting problems can also be mitigated.

Now cross validation method is used to split the dataset for training and testing. 59 instances are selected for training the algorithm and one instance will be left to test the algorithm. This will continue up to 60 times where each instance will get a chance to be the test instance. So predictive model will get 60 predictions for every testcase and average of all 60 predictions will be given as prediction capability of the predictive model for particular testcase.

The primary aim in this experiment is to evaluate the performance of the both algorithms in terms of prediction of the failed testcase. In predictive algorithms there are many evaluation methods like table summary, Box and whisper plots, prediction accuracy among others. which helps to evaluate the performance of the algorithm. All evaluation methods are applicable for K-nearest algorithm but for artificial neural network only prediction accuracy is available, So In this research, Evaluation of each model is done by the non-linear classification prediction accuracy metrics which is sum of predictions divided by the total number of predictions and multiplied by 100 to get into percentage.

$$\text{Prediction accuracy} = \frac{\text{Sum of all predictions*100}}{\text{Number of predictions}}$$

4.5 Predictive model’s description

In the RStudio, Artificial neural network is implemented by importing the neural net package. neural network package is used to train neural networks using with or without weights backpropagation. This package allows to flexible settings through custom-choice of error and activation function [67].

K-nearest neighbor is implemented using the caret package. Caret package contains functions to streamline the model training process for the complex regression and classification problems. Caret package also helpful in implementation of other algorithms like Discriminant analysis, Decision trees, Support vector machine, Naïve bayes [67].

Testing and training of the model is done using leave one out cross validation technique. In the input instances, File path is considered as one of the features in the experiment but file path is combination of many documents, which cannot be trained to predictive model. As predictive algorithms can only be trained with the numerical value. Data reduction techniques need to perform to get numerical value. To do this, Document term matrix is performed. Document term matrix is a mathematical matrix that describes the numerical frequency of the terms that occur in a collection of documents. To implement document term matrix package (tm) is used. This is text mining package which is used to convert the combination of documents into numerical value.

For the evaluation of predictive models MLmetrics package used. This is evaluation metrics package commonly used in supervised machine learning. It implements metrics for classification, regression, time series and information retrieval problems.

4.5.1 Artificial Neural Network predictive model

```
Step 1. for(i in 1:60)

Step 2. {

Step 3.  train <- raw[-(folds[[i]]),]

Step 4. test <- raw[(folds[[i]]),]

Step 5. fit<-  nnet(T1~No.of.files+filepath+Insertions+Deletions,
data=train,size=4,decay=0.0001)

Step 6. predictions <- predict(fit, test)

Step 7. acc [i] <- Accuracy(predictions, test$T1)

Step 8.  }

Step 9. avg_acc[1] <- mean(acc)
```

4.5.1.1 R script for implementation of Artificial Neural Network

Above code (see Figure 4.5.1.1) is implementation of artificial neural network in r script. First training and testing is done using cross validation method (see Figure 4.5.1.1 step 1, step 3, step 4). Neural network learning is stored in variable fit (see Figure 4.5.1.1 step 5). Now predictions are done for the fit and the test using prediction metrics (see Figure 4.5.1.1 step 6). Accuracy of the predictions is known by acc[i] (see Figure 4.5.1.1 step 7). This process is also same for the all other testcases. finally using evaluation metrics mean(acc) (see Figure 4.5.1.1 step 9) predictive capability of the predictive model is calculated.

4.5.2 K-nearest neighbor predictive model

```
Step 1. for(i in 1:60)

Step 2. {

Step 3. train <- raw[-(folds[[i]]),]

Step 4. test <- raw[(folds[[i]]),]

Step 5.  fit <- knn3(T1~No.of.files+filepath+Insertions+Deletions,
data=train,k=5)

Step 6.  predictions <- predict(fit, test)

Step 7.  acc[i] <- Accuracy(predictions, test$T1) }

Step 8.avg_acc[1] <- mean(acc)
```

4.5.2.1 R script for implementation of K-nearest neighbor

Above code (see Figure 4.5.2.1) is implementation of k nearest neighbor in r script . First training and testing is done using cross validation method (see Figure 4.5.2.1 step 1, step 3, step 4). K-nearest neighbor learning is stored in variable fit(see Figure 4.5.2.1 step 5). Now predictions are done for the fit and the test using prediction metrics (see Figure 4.5.2.1 step 6). Accuracy of the predictions is known by acc[i] (see Figure 4.5.2.1 step 7). This process is also same for the all other testcases. finally using evaluation metrics mean(acc) (see Figure 4.5.2.1 step 8) predictive capability of the predictive model is calculated.

4.6 Threats to internal and external validity

Internal validity and threats

Experiment is implemented to discover cause and effect relationships but research cannot terminate that changes in the independent variable caused the observed changes in the dependent variable. There is a need to prove internal validity to show strong evidence of the causality [46]. Internal validity is establishment of accuracy in selecting dependent variables and independent variables [46]. In this experiment dependent variable is performance of the algorithm and independent variable is dataset and algorithms. Dataset consists of no of files, file path, insertions, deletions, and results of the testcases. To mitigate the threat to internal validity selection of file changes and testcases is considered carefully and only consistent failed testcases and their file changes are considered for the experiment. This testcases are selected only after continuously analysis of 60 days data and functional testers of Ericsson also verified that these are consistent testcase failures. To ensure internal validity, during data pre-processing all threats related to the internal validity are alleviated and no bias is shown while selection process. The equivalent dataset is used for both predictive models so internal validity threat due to the dataset is alleviated.

External validity and threats

External validity refers to the degree to which the results of an empirical investigation can be generalized to and across individuals, settings, and times [46]. Data is acquired from the Ericsson historical data only for MINI-LINK product so if generalization of results to other products is not endorsed as different product will have different requirements and limitations. But if the other product development cycle is practiced in a continuous integration environment with Jenkins as automation server than generalization may be feasible to some settings.

Construct validity

Construct validity alludes to the degree to which inferences can truly be produced from the measurements [69]. Construct validity can be ensured by using best evaluation metrics in experiment. The evaluation of predictive models is done to know their competence in prediction capability. Metrics used for evaluation of prediction capability is prediction accuracy, which is most prevailing evaluation metrics for prediction of non-linear classification model in R scripts.

Statistical Conclusion validity

Conclusion validity is the degree to which conclusions we reach about the relationships in our data are reasonable and not misunderstood [69]. To ensure statistical conclusion validity statistical tests need to be performed. Statistical test 2 Sample T-test is performed in this experiment to validate the results of the experiment.

5.Results

5.1 Artificial Neural networks predictive model

After Implementation of experiment in RStudio, Results are found for the artificial neural network predictive model.
Below are the predictions obtained for Artificial Neural Network for each testcase.

Testcase	Prediction accuracy
Testcase 1	83%
Testcase 2	79%
Testcase 3	67%
Testcase 4	74%
Testcase 5	81%
Testcase 6	68%
Testcase 7	72%
Testcase 8	79%

5.1.1 Predictions obtained for artificial neural network

In the system, Results are found for each testcase. Now using prediction accuracy evaluation metrics. Performance of the model is calculated that is sum of predictions observed divided by the total number of predictions made multiplied by 100 to get into percentage.

$$\text{Prediction accuracy} = \frac{\text{Sum of all predictions*100}}{\text{Number of predictions}}$$
$$= \frac{(83+79+67+74+81+68+72+79)\ *100}{8}$$
$$= \ 75.3\%$$

The prediction capability of the neural network predictive model is 75.3%

5.2 K-nearest neighbor predictive model

After Neural Network is implemented, Now K-nearest Neighbor is implemented. Below is the prediction accuracy for every testcase.

Testcases	Prediction accuracy
Testcase 1	74%
Testcase 2	68%
Testcase 3	59%
Testcase 4	69%
Testcase 5	71%
Testcase 6	65%
Testcase 7	69%
Testcase 8	67%

5.2.1 Predictions obtained for K-nearest neighbor

In the system, Results are found for every testcase. Now using prediction accuracy evaluation metrics. Performance of the model will be calculated that is sum of predictions observed divided by the total number of predictions made multiplied by 100 to get into percentage.

$$\text{Prediction accuracy} = \frac{\text{Sum of all predictions}*100}{\text{Number of predictions}}$$

$$= \frac{(74+68+59+69+71+65+69+67)*100}{8}$$

$$= 67.75\%$$

The prediction accuracy for the K-nearest neighbor model is 67.75%.

5.3 Statistical Tests

Statistical test is a test which verifies the hypothesis concerns on the distribution of one or more variables to accept or reject [70]. In this experiment to conduct statistical hypothesis testing, 2-Sample T-test is selected. 2-Sample T-test is a common scientific process used in statistical and social science disciplines [71]. It performs comparison in-between two sample populations to find difference in them.

The 2 Sample t-test is done in four steps.

1. **Define the hypothesis** -Alternative and Null Hypothesis are defined as below
    - Null Hypothesis(H0): Null Hypothesis is broadly defining that there is no observed change and effect relationship in experiment [71]. It is denoted as (H0). It cooperates to find evidence against hypothesis test. In this experiment, Null hypothesis is “There is no significant statistical performance difference in between both predictive models regarding prediction competence”
    - Alternative Hypothesis(H1): Alternative Hypothesis is broadly defining that there is an observed change and effect relationship in the experiment [71]. It is denoted as (H1) In this experiment Alternative hypothesis is “There is a significant statistical performance difference in between both predictive models regarding prediction competence”
2. **Choosing Level of significance** – In Statistical Hypothesis testing, Null hypothesis is selected as default true statement for every hypothesis testing. Null hypothesis can be rejected only when it occurs Type 1 error. Type 1 error occur only when level of significance is below the p-value. If level of significance is selected as 0.05 that means there is a probability that there is a 5% occurring of Type 1 error and rejecting Null hypothesis [71]. This result explains that there is 95% level of confidence over the hypothesis testing. If level of significance is selected as 0.01 that means there is a 1% occurring type 1 error and rejecting null hypothesis, this result translates that there is 99% level of confidence over the hypothesis testing [71]. In this experiment level of significance is selected as 0.05 to get 95% level of confidence on hypothesis testing
3. **Calculating data-** Calculating values of mean, standard deviation, test static, degree of freedom, P-value is done in RStudio. Results obtained in Rstudio are as follows, mean values for Artificial predictive model results is 75.375

for K-nearest neighbor predictive model results is 67.75. Standard deviation results are for Artificial neural network predictive model results is 6.0223 and for K-nearest neighbor is 4.432. T-statistic result is obtained as 2.876. Degree of freedom results are obtained as 14. P- value results are obtained as 0.012.

4. **Interpreting results** - After getting P-value now p-value is analyzed with level of significance. P-value is less than that of level of significance (p-value <0.05). This results states that null hypothesis is rejected and alternative hypothesis is selected that means that there is statistical significant performance difference in-between both predictive models in predictive competence.

5.4 Challenges encountered amid experiment

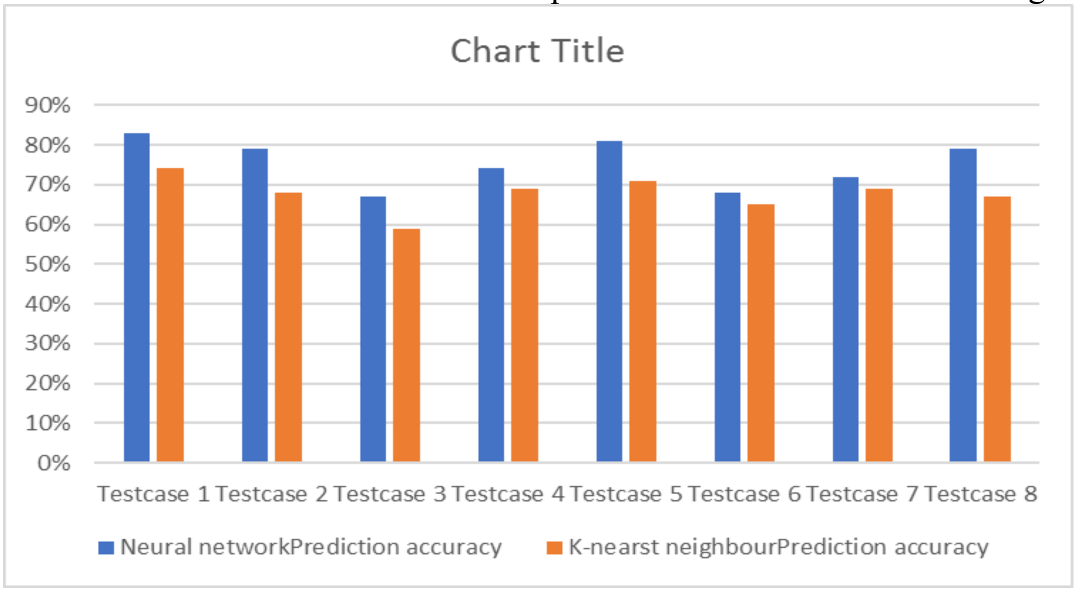
The performance is attained even there are many constraints challenged amid experiment

1. Major constraint encountered amid experiment is data collection. Ericsson doesn't arrange data more than 30 days. limited data have influence on the performance accuracy as sufficient required data will not be trained so prediction can happen with lower accuracy.
2. Contemporary Continuous integration environment in the Ericsson didn't accumulate any commit data information for the hourly cycle. This has major impact on the prediction accuracy. The data on which this experiment implemented is daily cycle data that is final integration of the day that contains all day commits. So, it is not possible to catch which commit has made failure for the testcase.it is challenging to predict correct future failed testcase without appropriate failure testcase commit data.
3. Literature on the Pattern recognition applications on software development process is limited. So, algorithms and models training is inspired from the other applications.

If these constraints are not present in the Ericsson environment than performance of the predictive model would have been more than now.

6. Analysis and Discussion

In the Experiment, the results of the prediction accuracy of both models are obtained. Artificial Neural Network has prediction accuracy of 75.3% and k-nearest neighbor has 67.75%. By observing the accuracy of the prediction results in this research and confirming the statistical significance of performance difference in-between both predictive models. This research state that neural networks have better level of prediction than k-nearest neighbor.



6.1.1 Comparison of results of both predictive models

In the above (see Chart 6.1.1) Artificial Neural network is performing better in prediction of every testcase. While implementing the artificial neural network and its data prediction tells that in future it can handle large amount of data as well as it can produce good prediction results, when there is appropriate data
.
Answers for research questions attained after doing experiment

1.How can pattern recognition be implemented in continuous integration environment?
A. Pattern recognition can be implemented in continuous integration environment for improving its testcase selection and prioritization in regression testing. Present testcase selection and prioritization can be made dynamic using pattern recognition by building predictive model.

For building predictive model data must be collected which contains attributes affecting testcase results. Predictive model will be trained with Ericsson historical data containing attributes affecting testcase results like number of changed files, insertions made in the files, deletions made in the files. Now when new commit is merged in product, the commit will make changes to the files in the continuous integration system. Based on the file changing attributes, trained predictive model predicts the testcases which might fail according to the new file changes.

2.Which pattern recognition model has the best prediction competence for test case selection and prioritization in regression testing for continuous integration?

A. Artificial Neural network has the prediction accuracy of 75.3 % more than the k-nearest neighbor which got 67.75%. If data is available for more number of days than present 30 days data and if hourly cycle data is also present these prediction accuracy results may increase as many patterns can be found as different pattern recognition books [42] [43] indicate that more data helps to get more prediction accuracy.

After seeing results and Neural network had prediction capability of performing 75.3% and k-nearest neighbor has prediction capability of performing 67.75%. Neural networks have proved effective capability of predicting failure testcase by 75.3% over the k-nearest neighbor. Predictive model can only make continuous integration efficient only if it has 100% prediction capability as present static testcase selection and prioritization is able to select and prioritize most of the testcases for hourly cycle it is only failing to select and prioritize one or two new testcases in hourly cycle. If predictive model dynamic testcase selection and prioritization is integrated in the continuous integration system with 75.3% prediction capability it may also not able to select and prioritize new testcases like the static testcase selection and prioritization than the prediction capability of the 75.3% will not make continuous integration system more efficient system as it has deficiency of lacking prediction 25%. This research can only conclude that neural networks have at present has 75.3% prediction capability but in future when data availability is more. Predictive capability may reach to 100% than this model can be integrated in continuous integration.

7.Conclusion and Future work

This Research Investigated the feasibility of an adaptive and self-learning test machinery by pattern recognition in continuous integration environment for improving testcase selection and prioritization in regression testing. Artificial Neural networks has demonstrated effective capability of predicting failure testcase by 75.3% over k-nearest neighbor predictive model.

Predictive model can only make continuous integration efficient only when it has 100% prediction capability, the prediction capability of the 75.3% will not make Continuous integration system more efficient than present static testcase selection and prioritization as it has deficiency of lacking prediction 25%. So, this research can only conclude that Artificial neural networks at present has 75.3% prediction capability but in future when data availability is more, this may reach to 100%.

Acquiring the appropriate data is major constraint faced during research. The present Ericsson continuous integration system needs to improve its data storage for historical data at present it can only store 30 days of historical data. The predictive models require large data to give good prediction.
The possible measures can be implemented to store more data

1.To support continuous integration at present Ericsson is using Jenkins automation server, there are other automation servers like Team city, Travis CI, Go CD, Circle CI which can store data more than 30 days using them will mitigate the problem of data storage

2. Gerrit is used for code repository management for legacy codes but Gerrit have capacity to store only edited development codes. Error prone codes are not stored so other code review tools like collaborator, codacy, kiuwan has capacity to store error-prone development codes. Using them can solve the problem of data acquisition for commits.

Future work:

By doing this research and after scrutinizing results, this research propose that artificial neural networks have best prediction capability for finding failure testcase in the continuous integration environment to improve testcase selection and prioritization in regression testing.

In future this research can be progressed using multi-class classification where skipped testcase failures and hardware failures data can be included. This will also improve testcase selection and prioritization in the continuous integration system.

There is wide research scope in the automation of continuous integration using predictive concepts, which is yet to be done in this field. Continuous integration is the wide area and it can be more utilized by integrating pattern recognition and machine learning algorithms during testing, development, and deployment. Implementation of predictive models in continuous integration will increase the product efficiency.

Future work can extend the pattern recognition algorithms implementation in gate testing. This helps to get error free development code without any usage of testcases.

8.References

[1] Seth, Nikita, and Rishi Khare. "ACI (Automated Continuous Integration) Using Jenkins: Key for Successful Embedded Software Development." In *Recent Advances in Engineering & Computational Sciences (RAECS), 2015 2nd International Conference on*, 1–6. IEEE, 2015.
[2] Brandtner, Martin, Emanuel Giger, and Harald Gall. "Supporting Continuous Integration by Mashing-up Software Quality Information." In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on*, 184–193. IEEE, 2014.
[3] Zampetti, Fiorella, Simone Scalabrino, Rocco Oliveto, Gerardo Canfora, and Massimiliano Di Penta. "How Open Source Projects Use Static Code Analysis Tools in Continuous Integration Pipelines," 334–44. IEEE, 2017.
[4] Mossige, Morten, Arnaud Gotlieb, and Hein Meling. "Test Generation for Robotized Paint Systems Using Constraint Programming in a Continuous Integration Environment," 489–90.
[5] Dösinger, Stefan, Richard Mordinyi, and Stefan Biffl. "Communicating Continuous Integration Servers for Increasing Effectiveness of Automated Testing." In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, 374–377. ACM, 2012.
[6] Qiu, Dong, Bixin Li, Shunhui Ji, and Hareton Leung. "Regression Testing of Web Service: A Systematic Mapping Study." *ACM Computing Surveys* 47, no. 2 (August 25, 2014)
[7] Hilton, Michael, Timothy Tunnell, Kai Huang, Darko Marinov, and Danny Dig. "Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects," 426–37. ACM Press, 2016.
[8] Vassallo, Carmine, Fiorella Zampetti, Daniele Romano, Moritz Beller, Annibale Panichella, Massimiliano Di Penta, and Andy Zaidman. "Continuous Delivery Practices in a Large Financial Organization," 519–28. IEEE.
[9] Stolberg, Sean. "Enabling Agile Testing through Continuous Integration," 369–74. IEEE, 2009.
[10] Saff, David, and Michael D. Ernst. "An Experimental Evaluation of Continuous Testing during Development." In *ACM SIGSOFT Software Engineering Notes*, 29:76–85. ACM, 2004.
[11] Vasilescu, Bogdan, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. "Quality and Productivity Outcomes Relating to Continuous Integration in GitHub." In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 805–816. ACM, 2015.
[12] Vincenzi, Auri M. R., Tiago Bachiega, Daniel G. de Oliveira, Simone R. S. de Souza, and José C. Maldonado. "The Complementary Aspect of Automatically and Manually Generated Test Case Sets," 23–30. ACM Press, 2016.
[13] R, Beena, and Sarala S. "Code Coverage Based Test Case Selection and Prioritization." *International Journal of Software Engineering & Applications* 4, no. 6 (November 30, 2013): 39–49.

[14] Elbaum, Sebastian, Alexey G. Malishevsky, and Gregg Rothermel. *Prioritizing Test Cases for Regression Testing*. Vol. 25. 5. ACM, 2000.
[15] Pravin. “EFFECTIVE TEST CASE SELECTION AND PRIORITIZATION IN REGRESSION TESTING.” *Journal of Computer Science* 9, no. 5 (May 1, 2013): 654–59.
[16] Arafeen, Md Junaid, and Hyunsook Do. “Test Case Prioritization Using Requirements-Based Clustering.” In *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*, 312–321. IEEE, 2013.
[17] Suppriya, M., and A. K. Ilavarasi. “Test Case Selection and Prioritization Using Multiple Criteria.” *International Journal* 5, no. 1 (2015).
[18] Musa, Samaila, A. Sultan, Abdul-Ghani A. Md, and Salmi Baharom. “A Regression Test Case Selection and Prioritization for Object-Oriented Programs Using Dependency Graph and Genetic Algorithm.” *Research Inventy: International Journal of Engineering and Science* 4, no. 7 (2014): 54–64.
[19] Pellegrini, Alessandro, Pierangelo Di Sanzo, and Dimiter R. Avresky. “A Machine Learning-Based Framework for Building Application Failure Prediction Models,” 1072–81. IEEE, 2015.
[20] Liang Hu, Xi-Long Che, and Si-Qing Zheng. “Online System for Grid Resource Monitoring and Machine Learning-Based Prediction.” *IEEE Transactions on Parallel and Distributed Systems* 23, no. 1 (January 2012): 134–45.
[21] Benchettara, Nesserine, Rushed Kanawati, and Céline Rouveirol. “Supervised Machine Learning Applied to Link Prediction in Bipartite Social Networks,” 326–30. IEEE, 2010.
[22] Papadopoulou, Nikela, Georgios Goumas, and Nectarios Koziris. “A Machine-Learning Approach for Communication Prediction of Large-Scale Applications,” 120–23. IEEE, 2015.
[23] Asri, Hiba, Hajar Mousannif, Hassan Al Moatassime, and Thomas Noel. “Using Machine Learning Algorithms for Breast Cancer Risk Prediction and Diagnosis.” *Procedia Computer Science* 83 (2016): 1064–69.
[24] Gao, Yunlong, Jin-yan Pan, and Feng Gao. “Improved Boosting Algorithm through Weighted K-Nearest Neighbors Classifier.” In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference On*, 6:36–40. IEEE, 2010.
[25] Makhoul, John. “Pattern Recognition Properties of Neural Networks.” In *Neural Networks for Signal Processing [1991]., Proceedings of the 1991 IEEE Workshop*, 173–187. IEEE, 1991.
[26] Briand, Lionel C., Victor R. Basili, and William M. Thomas. “A Pattern Recognition Approach for Software Engineering Data Analysis.” *IEEE Transactions on Software Engineering* 18, no. 11 (1992): 931–942.
[27] Jabba, Daladier, Miguel Rodriguez, Geovanni Berdugo, Maria Calle, Miguel A. Jimeno, and Eduardo E. Zurek. “A Pattern Recognition Procedure for the Identification of Digital Numbers in Electrical Meters Using Neural Networks.” In *Industrial Electronics and Applications (ISIEA), 2012 IEEE Symposium on*, 315–320. IEEE, 2012.

[28] Ali, Md Shahazan, and Md Nazrul Islam Mondal. “Character Recogntion System: Performance Comparison of Neural Networks and Genetic Algorithm.” In *Computer and Information Engineering (ICCIE), 2015 1st International Conference on*, 91–94. IEEE, 2015.
[29] Ciregan, Dan, Ueli Meier, and Jürgen Schmidhuber. “Multi-Column Deep Neural Networks for Image Classification.” In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 3642–3649. IEEE, 2012.
[30] Rigoll, G. “Mutual Information Neural Networks for Dynamic Pattern Recognition Tasks.” In *Industrial Electronics, 1996. ISIE’96., Proceedings of the IEEE International Symposium on*, 1:80–85. IEEE, 1996.
[31] Connell, Scott D., and N. K. Jain. “Writer Adaptation of Online Handwriting Models.” In *Document Analysis and Recognition, 1999. ICDAR’99. Proceedings of the Fifth International Conference on*, 434–437. IEEE, 1999.
[32] Haykin, Simon S., and Simon S. Haykin. *Neural Networks and Learning Machines*. 3rd ed. New York: Prentice Hall, 2009.
[33] Katagiri, Shigeru, Biing-Hwang Juang, and Chin-Hui Lee. “Pattern Recognition Using a Family of Design Algorithms Based upon the Generalized Probabilistic Descent Method.” *Proceedings of the IEEE* 86, no. 11 (1998): 2345–2373.
[34] Nagy, George. “Classification Algorithms in Pattern Recognition.” *IEEE Transactions on Audio and Electroacoustics* 16, no. 2 (1968): 203–212.
[35] Briand, Lionel C., Victor R. Basili, and William M. Thomas. “A Pattern Recognition Approach for Software Engineering Data Analysis.” *IEEE Transactions on Software Engineering* 18, no. 11 (1992): 931–942.
[36] Jain, Anil K., Robert P. W. Duin, and Jianchang Mao. “Statistical Pattern Recognition: A Review.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, no. 1 (2000): 4–37.
[37] Stepanyuk, Sergiy. “Neural Network Information Technologies of Pattern Recognition.” In *Perspective Technologies and Methods in MEMS Design (MEMSTECH), 2010 Proceedings of VIth International Conference on*, 210–216. IEEE, 2010.
[38] Nandini, Y, S. Bhargav, and G. Ravi Shastri. “Handwritten Pattern Recognition Using Neural Networks.” *International Journal of Scientific and Research Publications* “ no. 3 (2013): 1.
[39] Nagy, George. “Classification Algorithms in Pattern Recognition.” *IEEE Transactions on Audio and Electroacoustics* 16, no. 2 (1968): 203–212.
[40] Kim, Yongwook Bryce, Joohyun Seo, and Una-May OReilly. “Large-Scale Methodological Comparison of Acute Hypotensive Episode Forecasting Using MIMIC2 Physiological Waveforms,” 319–24. IEEE, 2014.
[41] Fukunaga, Keinosuke. *Introduction to Statistical Pattern Recognition*. 2. ed., [Reprint]. Computer Science and Scientific Computing. San Diego: Academic Press, 2009.
[42] Devroye, Luc, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. 3. print. Applications of Mathematics 31. New York, NY: Springer, 2008.

[43] Webb, Andrew R. *Statistical Pattern Recognition*. 2. ed., Reprint. Chichester: Wiley, 2004.
[44] Fu, K. C. *Sequential Methods in Pattern Recognition and Machine Learning*. Vol. 52. Academic press, 1968.
[45] Bishop, Christopher M. *Pattern Recognition and Machine Learning*. Corrected at 8. printing 2009. Information Science and Statistics. New York, NY: Springer, 2009.
[46] Boettger, Ryan K., and Chris Lam. "An Overview of Experimental and Quasi-Experimental Research in Technical Communication Journals (1992–2011)." *IEEE Transactions on Professional Communication* 56, no. 4 (December 2013): 272–93.
[47] Pellegrini, Alessandro, Pierangelo Di Sanzo, and Dimiter R. Avresky. "A Machine Learning-Based Framework for Building Application Failure Prediction Models," 1072–81. IEEE, 2015.
[48] Liang Hu, Xi-Long Che, and Si-Qing Zheng. "Online System for Grid Resource Monitoring and Machine Learning-Based Prediction." *IEEE Transactions on Parallel and Distributed Systems* 23, no. 1 (January 2012): 134–45.
[49] Benchettara, Nesserine, Rushed Kanawati, and Céline Rouveirol. "Supervised Machine Learning Applied to Link Prediction in Bipartite Social Networks," 326–30. IEEE, 2010.
[50] Papadopoulou, Nikela, Georgios Goumas, and Nectarios Koziris. "A Machine-Learning Approach for Communication Prediction of Large-Scale Applications," 120–23. IEEE, 2015.
[51] Asri, Hiba, Hajar Mousannif, Hassan Al Moatassime, and Thomas Noel. "Using Machine Learning Algorithms for Breast Cancer Risk Prediction and Diagnosis." *Procedia Computer Science* 83 (2016): 1064–69.
[52] Ali, AI-Haj, and M. H. Kabir. "Wavelets Pre-Processing of Artificial Neural Networks Classifiers." *IEEE Transactions on Consumer Electronics* 53, no. 2 (2008)
[53] Aleem, Saiqa, Luiz Fernando Capretz, and Faheem Ahmed. "Benchmarking Machine Learning Techniques for Software Defect Detection." *International Journal of Software Engineering & Applications* 6, no. 3 (May 31, 2015): 11–23.
[54] Gondra, Iker. "Applying Machine Learning to Software Fault-Proneness Prediction." *Journal of Systems and Software* 81, no. 2 (February 2008): 186–95.
[55] Di Nardo, Daniel, Nadia Alshahwan, Lionel Briand, and Yvan Labiche. "Coverage-Based Regression Test Case Selection, Minimization and Prioritization: A Case Study on an Industrial System: COVERAGE-BASED REGRESSION TESTING: AN INDUSTRIAL CASE STUDY." *Software Testing, Verification and Reliability* 25, no. 4 (June 2015): 371–96.
[56] Kandil, Passant, Sherin Moussa, and Nagwa Badr. "Cluster-Based Test Cases Prioritization and Selection Technique for Agile Regression Testing: Cluster-Based Technique for Agile Regression Testing." *Journal of Software: Evolution and Process* 29, no. 6 (June 2017): e1794.
[57] Xu, Zhihong, Yunho Kim, Moonzoo Kim, Myra B. Cohen, and Gregg Rothermel. "Directed Test Suite Augmentation: An Empirical Investigation:

DIRECTED TEST SUITE AUGMENTATION: AN EMPIRICAL INVESTIGATION.” *Software Testing, Verification and Reliability* 25, no. 2 (March 2015): 77–114.
[58] Kim, Jeongho, Hohyeon Jeong, and Eunseok Lee. “Failure History Data-Based Test Case Prioritization for Effective Regression Test,” 1409–15. ACM Press, 2017.
[59] Elbaum, Sebastian, Gregg Rothermel, and John Penix. “Techniques for Improving Regression Testing in Continuous Integration Development Environments,” 235–45. ACM Press, 2014.
[60] Hemmati, Hadi, Zhihan Fang, Mika V. Mäntylä, and Bram Adams. “Prioritizing Manual Test Cases in Rapid Release Environments: Prioritizing Manual Test Cases in Rapid Release Environments.” *Software Testing, Verification and Reliability* 27, no. 6 (September 2017): e1609.
[61] Gonzalez-Sanchez, Alberto, Éric Piel, Rui Abreu, Hans-Gerhard Gross, and Arjan J. C. van Gemund. “Prioritizing Tests for Software Fault Diagnosis.” *Software: Practice and Experience*, 2011, n/a-n/a.
[62] Yoo, S., and M. Harman. “Regression Testing Minimization, Selection and Prioritization: A Survey.” *Software Testing, Verification and Reliability*, 2010, n/a-n/a.
[63] Kundu, Debasish, Monalisa Sarma, Debasis Samanta, and Rajib Mall. “System Testing for Object-Oriented Systems with Test Case Prioritization.” *Software Testing, Verification and Reliability* 19, no. 4 (December 2009): 297–333.
[64] Srikanth, Hema, Sean Banerjee, Laurie Williams, and Jason Osborne. “Towards the Prioritization of System Test Cases: TOWARDS THE PRIORITIZATION OF SYSTEM TEST CASES.” *Software Testing, Verification and Reliability* 24, no. 4 (June 2014): 320–37.
[65] Brenning, Alexander. “Spatial Cross-Validation and Bootstrap for the Assessment of Prediction Rules in Remote Sensing: The R Package Sperrorest.” In *Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International*, 5372–5375. IEEE, 2012.
[66]Zhang, Xiangxiang, Zhijun Fang, Zhenghao Xi, and Xiaoshuang Liu. “Large Scale Face Data Purification Based on Correlation Function and Multi-Phase Grouping.” Edited by Yansong Wang. *MATEC Web of Conferences* 128 (2017): 02021.
[67] Venables, W. N., Brian D. Ripley, and W. N. Venables. *Modern Applied Statistics with S*. 4th ed. Statistics and Computing. New York: Springer, 2002.
[68] Shinde, Priyanka P., Kavita S. Oza, and R. K. Kamat. “Big Data Predictive Analysis: Using R Analytical Tool.” In *I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017 International Conference On*, 839–842. IEEE, 2017.
[69] Feldt, Robert, and Ana Magazinius. “Validity Threats in Empirical Software Engineering Research-An Initial Survey.” In *SEKE*, 374–379, 2010.
[70] Simion, Emil. “The Relevance of Statistical Tests in Cryptography.” *IEEE Security & Privacy* 13, no. 1 (2015).

[71] Jiaxi, Li. “The Application and Research of T-Test in Medicine,” 321–23. IEEE, 2010.

Appendix

Artificial Neural Network.R

```
library(neuralnet)
library(MLmetrics)
library(tm)
raw <- read.csv("C:/Users/eshvang/Desktop/data.csv", stringsAsFactors =
TRUE, header = TRUE, sep = ";")
corpus <- Vcorpus(Vectorsource(Filepath))
tdm <- Document TermMatrix(corpus,list(stopwords=TRUE,
removePunctuation = TRUE, asPlain= TRUE, stemming=TRUE,
removenumbers =TRUE)
filepath <- as.matrix(tdm)
filepath <-cbind(filepath, c(0,1))
colnames(filepath)[ ncol(filepath)] <- 'y'
filepath <- as.dataframe(filepath)
filepath$y <-as.factor(filepath$y)
size  <- (sample(nrow(raw), floor(nrow(raw) * 0.7)))
folds <- createFolds(raw$T1, k = 60)
acc <- NULL
avg_acc <- NULL
for(i in 1:60)
  {

         train <- raw[-(folds[[i]]),]
        test <- raw[(folds[[i]]),]
        fit<-
        nnet(T1~No.of.files+filepath+Insertions+Deletions,data=train,size=4,deca
        y=00001)
        predictions <- predict(fit, test)
        acc[i] <- Accuracy(predictions, test$T1)
```

```
}
avg_acc[1] <- mean(acc)

for(i in 1:60)
{

    train <- raw[-(folds[[i]]),]
    test <- raw[(folds[[i]]),]
    fit <-
    nnet(T2~No.of.files+filepath+Insertions+Deletions,data=train,size=4,dec
    ay=0.0001)
    predictions <- predict(fit, test)
    acc[i] <- Accuracy(predictions, test$T2)

}
avg_acc[2] <- mean(acc)

for(i in 1:60)
{

    train <- raw[-(folds[[i]]),]
    test <- raw[(folds[[i]]),]
    fit <-
    nnet(T3~No.of.files+filepath+Insertions+Deletions,data=train,size=4,dec
    ay=0.0001)
    predictions <- predict(fit, test)
    acc[i] <- Accuracy(predictions, test$T3)

  }
  avg_acc[3] <- mean(acc)

  for(i in 1:60)
  {
```

```
  train <- raw[-(folds[[i]]),]
  test <- raw[(folds[[i]]),]
   fit<-
  nnet(T4~No.of.files+filepath+Insertions+Deletions,data=train,size=4,deca
  y=0.01)
  predictions <- predict(fit, test)
  acc[i] <- Accuracy(predictions, test$T4)

  }
  avg_acc[4] <- mean(acc)

 for(i in 1:60)
 {
  train <- raw[-(folds[[i]]),]
  test <- raw[(folds[[i]]),]
  fit<-
  nnet(T5~No.of.files+filepath+Insertions+Deletions,data=train,size=4,deca
  y=0.0001)
  predictions <- predict(fit, test)
  acc[i] <- Accuracy(predictions, test$T5)

 }
 avg_acc[5] <- mean(acc)

for(i in 1:60)
{
   train <- raw[-(folds[[i]]),]
   test <- raw[(folds[[i]]),]
    fit<-
  nnet(T6~No.of.files+filepath+Insertions+Deletions,data=train,size=4,deca
  y=0.0001)
  predictions <- predict(fit, test)
  acc[i] <- Accuracy(predictions, test$T6)

 }
```

```
  avg_acc[6] <- mean(acc)

  for(i in 1:60)
  {

      train <- raw[-(folds[[i]]),]
      test <- raw[(folds[[i]]),]
      fit<-
     nnet(T7~No.of.files+filepath+Insertions+Deletions,data=train,size=4,deca
     y=0.0001)
     predictions <- predict(fit, test)
     acc[i] <- Accuracy(predictions, test$T7)

  }
 avg_acc[7] <- mean(acc)

 for(i in 1:60)
 {
     train <- raw[-(folds[[i]]),]
     test <- raw[(folds[[i]]),]
       fit<-nnet(T8~No.of.files+filepath+Insertions+Deletions,
     data=train,size=4,decay=0.0001)
     predictions <- predict(fit, test)
     acc[i] <- Accuracy(predictions, test$T8)

 }
 avg_acc[8] <- mean(acc)
```

K-nearest neighbour.R

```
library(caret)
library(MLmetrics)
library(tm)
raw <- read.csv("C:/Users/eshvang/Desktop/iris.csv", stringsAsFactors =
TRUE, header = TRUE, sep = ";")
corpus <- Vcorpus(Vectorsource(Filepath))
tdm <- Document TermMatrix(corpus,list(stopwords=TRUE,
removePunctuation = TRUE, asPlain= TRUE, stemming=TRUE,
removenumbers =TRUE)
filepath <- as.matrix(tdm)
filepath <-cbind(filepath, c(0,1))
colnames(filepath)[ ncol(filepath)] <- 'y'
filepath <- as.dataframe(filepath)
filepath$y <-as.factor(filepath$y)
folds <- createFolds(raw$T1, k = 60)
acc <- NULL
avg_acc <- NULL
for(i in 1:60)
{
    train <- raw[-(folds[[i]]),]
    test <- raw[(folds[[i]]),]
    fit <- knn3(T1~No.of.files+filepath+Insertions+Deletions, data=train,k=5)
    predictions <- predict(fit, test)
    acc[i] <- Accuracy(predictions, test$T1)

 }
 avg_acc[1] <- mean(acc)
 for(i in 1:60)
 {
    train <- raw[-(folds[[i]]),]
    test <- raw[(folds[[i]]),]
```

```
        fit <- knn3(T2~No.of.files+filepath+Insertions+Deletions, data=train,k=5)
        predictions <- predict(fit, test)
        acc[i] <- Accuracy(predictions, test$T2)

    }
    avg_acc[2] <- mean(acc)
    for(i in 1:60)
    {
        train <- raw[-(folds[[i]]),]
        test <- raw[(folds[[i]]),]
        fit <- knn3(T3~No.of.files+filepath+Insertions+Deletions, data=train,k=5)
        predictions <- predict(fit, test)
        acc[i] <- Accuracy(predictions, test$T3)

    }
    avg_acc[3] <- mean(acc)
    for(i in 1:60)
   {
        train <- raw[-(folds[[i]]),]
        test <- raw[(folds[[i]]),]
        fit <- knn3(T4~No.of.files+filepath+Insertions+Deletions, data=train,k=5)
        predictions <- predict(fit, test)
        acc[i] <- Accuracy(predictions, test$T4)

   }
avg_acc[4] <- mean(acc)
  for(i in 1:60)
   {
        train <- raw[-(folds[[i]]),]
        test <- raw[(folds[[i]]),]
        fit <- knn3(T5~No.of.files+filepath+Insertions+Deletions, data=train,k=5)
        predictions <- predict(fit, test)
        acc[i] <- Accuracy(predictions, test$T5)
```

```
 }
avg_acc[5] <- mean(acc)
for(i in 1:60)
{
     train <- raw[-(folds[[i]]),]
     test <- raw[(folds[[i]]),]
     fit <- knn3(T6~No.of.files+filepath+Insertions+Deletions, data=train,k=5)
     predictions <- predict(fit, test)
     acc[i] <- Accuracy(predictions, test$T6)

 }
avg_acc[6] <- mean(acc)
for(i in 1:60)
{
     train <- raw[-(folds[[i]]),]
     test <- raw[(folds[[i]]),]
     fit <- knn3(T7~No.of.files+filepath+Insertions+Deletions, data=train,k=5)
     predictions <- predict(fit, test)
     acc[i] <- Accuracy(predictions, test$T7)

  }
avg_acc[7] <- mean(acc)
for(i in 1:60)
{
     train <- raw[-(folds[[i]]),]
     test <- raw[(folds[[i]]),]
     fit <- knn3(T8~No.of.files+filepath+Insertions+Deletions, data=train,k=5)
     predictions <- predict(fit, test)
     acc[i] <- Accuracy(predictions, test$T8)

}
  avg_acc[8] <- mean(acc)
```