ANALYZING MULTI-OBJECTIVE SOFTWARE TEST EFFORT ESTIMATION TECHNIQUES

OSMAN BERKCAN DERYA

AUGUST 2023

ÇANKAYA UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

M.Sc. Thesis in

COMPUTER ENGINEERING

ANALYZING MULTI-OBJECTIVE SOFTWARE TEST EFFORT ESTIMATION TECHNIQUES

OSMAN BERKCAN DERYA

AUGUST 2023

ABSTRACT

ANALYZING MULTI-OBJECTIVE SOFTWARE TEST EFFORT ESTIMATION TECHNIQUES

DERYA, OSMAN BERKCAN
M.Sc. in Computer Engineering

Supervisor: Assoc. Prof. Dr. Tansel DÖKEROĞLU
August 2023, 75 pages

Software testing effort estimation is an estimate of the approximate time and resources required by an engineer during the testing phase of a software project. Effort estimation of software test process is one of the most significant stages in the software development process to determine the test effort of the software project. Estimating the effort closest to the real effort is of great importance for both the company providing this service and the customers, especially the software testers. Because wrong software test effort estimations cause projects not to be completed or spread over a wide period of time. Therefore, different methods have been developed in the literature for software test effort estimation. In this thesis, machine learning methods with some feature selection method was used for estimating software test effort.

Estimation of software testing effort is found by running algorithms in the WEKA data mining tool. Algorithms were applied to 3 data sets (CocomoNasa, CocomoNasa-2, Cocomo-81) taken from PROMISE (Predictor Models in Software Engineering) data warehouse with 10-fold cross validation technique. After new models have been created, correlation coefficient was used for performance criterion.Besides MAE (Mean Absolute Error) and RAE (Relative AbsoluteError) were used for error rates.

Keywords: Machine learning, Weka, Attribute selection, Software test effort estimation

ÖZET

ÇOK YÖNLÜ YAZILIM TEST EFORU TAHMİNLEME TEKNİKLERİNİN ANALİZİ

DERYA, OSMAN BERKCAN
Bilgisayar Mühendisliği Yüksek Lisans

Danışman: Doç. Dr.Tansel DÖKEROĞLU
Ağustos 2023, 75 sayfa

Yazılım test efor tahmini, bir mühendisin yazılım projesinin test aşamasında ihtiyaç duyduğu yaklaşık süre ve kaynakların tahminidir. Yazılım test efor tahmini, yazılım projesinin test eforunu belirlemek için yazılım geliştirme sürecindeki en önemli aşamalardan birisidir. Gerçek efora en yakın efor tahminini yapmak yazılım test sorumluları başta olmak üzere hem bu hizmeti veren firma hem de hem de müşteriler için çok önemlidir. Çünkü yanlış yapılan yazılım test efor tahminleri projelerin tamamlanamamasına ya da geniş bir zaman dilimine yayılmasına neden olmaktadır.Bu yüzden yazılım test efor tahmini için literatürde farklı yöntemler geliştirilmiştir. Bu tez çalışmasında, yazılım test projelerinin eforu, Makine Öğrenmesi (MÖ) algoritmaları kullanılarak ve farklı methodlarla öznitelik seçimi yapılarak tahmin edilmeye çalışılmıştır.

Yazılım test eforunun tahmini, WEKA (Waikato Environment for Knowledge Analaysis – Bilgi Analizi için Waikato Ortamı) veri madenciliği aracında bulunan algoritmaların çalıştırlması sonucu bulunmuştur. Algoritmalar 10 kat çapraz doğrulama tekniği ile PROMISE (Yazılım Mühendisliğinde Tahmin Modelleri) veri deposundan alınan 3 adet veri setine (CocomoNasa, CocomoNasa-2, Cocomo-81) uygulanmıştır. Performans ölçütü olarak korelasyon katsayısı, Ortalama Mutlak Hata ve Bağıl Mutlak Hata, baz alınarak sonuçlar değerlendirilmiştir.

Anahtar Kelimeler: Makine öğrenimi, Yazılım Test Efor Tahmini,Weka,Öznitelik Seçimi

ACKNOWLEDGEMENT

I would like to thank Associate Professor Tansel DOKEROGLU for guiding me in completing this thesis.

I would like to thank my dear wife, Doğay DERYA, who supported me throughout my graduate education and never stopped believing in me in every aspect of life.

Also, I would like to thank you to my family who have always supported me throughout my life. I am grateful to all of you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

WEKA	:Waikato Environment for Knowledge Analysis
PROMISE	:Predictor Models in Software Engineering
MAE	:Mean Absolute Error
RAE	:Relative Absolute Error
ML	:Machine Learning
CFS	:Correlation Based Feature Selection
PSO	:Particle Swarm Optimization
GS	:Genetic Search

CHAPTER I
INTRODUCTION

With the developments in information technologies, effort estimation has gained a great importance for both software developers and customers. Software effort estimation is the process of estimating all kinds of resources that necessary to develop a software engineering project. Although software effort estimation is simple in concept, in reality it is difficult and complex. Therefore, many software projects could not be completed on time or the project costs were much higher than the estimated amount. More than half (60%) of substantial projects surpassed their planned budgets. It has been noted that some projects were never finish due to a 15% cost overrun. [1] In budget works conducted at different stages of software projects, sales amount and cost analyzes cannot be calculated realistically due to cost estimation difficulties. These difficulties may arise from the unique characteristics of the project, as well as the lack of information out of control, subjective interpretations in the evaluation of the information, direct and indirect cost separation errors that occur in cost analysis studies, and the inability to accurately estimate the project risks.

Similar to this process, software test effort estimation also severely affects the time required to complete a project. With the correct estimation of this period, a more accurate test planning process is entered, a correct job separation and sharing is made, the management of resources is provided more efficiently, and it plays an important role in minimizing changes and inconsistencies in project delivery dates. Although the software testing process was the most neglected of the processes in the software life cycle in the past, the importance given to this process is increasing thanks to the experiences today. In the planning phase of the test process, one of the most important issues is the test effort estimation phase for resource planning. The test process to be carried out after a correct test effort estimation process will ensure that the software products that will emerge will contain less errors.

The important thing is to define the right test processes and to ensure continuity throughout the project in a controlled manner [2]. Because software testing fulfills the

requirements in software, companies form teams among employees for software testing activities. In the realm of software testing, various elements such as emulators, manual testing, test documentation of test, and independent testing teams all play a role in shaping the final product. [3] Additionally, test managers overseeing testing efforts need to carefully plan their resources, including time estimation to incorporate testing into the software development process. However, the main challenge in achieving project goals often stems from inadequate estimation, insufficient data, and project personnel limitations. It's evident that estimating the testing effort in large projects is a complex task influenced by both internal and external factors. Relying solely on experiential estimates can be misleading. Despite the existence of estimation techniques in the literature for more precise and efficient software testing effort management, these methods remain underexplored, with a scarcity of research in this area. Consequently, there is a need for alternative approaches to accurately assess the efforts required in the software testing process.

In this research, which is the subject of this study, the effort in the software testing process was calculated using some machine learning algorithms. Afterwards, some feature selection pre-processes were made and hybrid methods were applied. The effect of these processes on the forecasting has been observed.

1.1 LITERATURE REVIEW

In this section, software testing and artificial learning algorithms are used together. An extensive literature review on research areas is included in the literature. The studies carried out in the software testing world have been used to identify open problems. So, it has been very useful in shaping this study.

Kafle [4], in 2014, examined 5 different companies and on test effort estimation. It was used 150 articles. It's been observed that organizations often depend on the insights and judgments of experts when making estimations regarding test effort. It has been observed that errors in estimating test effort are closely associated with errors in estimating the overall project effort. Nonetheless, this research does not put forth a particular method for estimating test effort; instead, it underscores the importance of conducting more extensive research in this field.

Hourani [5] mentioned that artificial intelligence sheds light on the future of software testing. In addition, as the use of artificial intelligence in software testing

increases. He argued that more consistent results would be obtained, tests could be performed more automatically, and software development efficiency would increase.

Sharma [6] studied the estimation of effort for software testing with the Neuro Fuzzy Inference System (NFIS) method in 2017.It was observed that a hybrid approach, which combines fuzzy logic and artificial neural networks (Hybrid method), yields superior outcomes.

Cotroneo [7] has conducted a study on adaptively combining test cases as tests progress, by teaching past test experiences to machine learning algorithms. He experienced an increase in error detection efficiency as he used the methodology he developed online during the tests.

Briand [8] developed a methodology that uses various machine learning models to reconfigure test cases for improvement. They achieved very positive results using the methodology they proposed in a case study involving black box testing.

Durelli [9] focused on a mapping study investigating how machine learning algorithms are used to improve software testing activities. In their research, it was stated that machine learning algorithms are used as a basis in areas such as automatic test scenario creation and improvement, test focus evaluation, test activities effort estimation. With this study, it is aimed to inform researchers about how software testing area and machine learning algorithms intersect and their current status in the literature.

1.2 THE AIM OF THE THESIS STUDY

The aims of this thesis can be summarized as follows:

-Performing performance analyzes of different ML algorithms on each data set and interpreting the results in detail,

-Cost estimation of software projects using ML algorithms on different datasets that are frequently used in the literature obtained from the PROMISE data store.

-Determining which features in the data sets are used together or which features are important and which features are unimportant for the selected algorithm, and which algorithms have higher success rates as a result of this.

1.3 THE ORGANIZATION OF THE THESIS STUDY

The thesis work consists of five main chapter.

Chapter 2, General Sections, consists of 6 sub-headings that are, description of software testing, examples of software testing importance, software testing types, software testing activities, effort estimation with software testing, machine learning concept.

Chapter 3, Applications Section, consists of 5 sub-headings that are machine learning algorithms, evaluation criteria, dataset, application platform, and findings. The sub-heading of the findings is divided into 2 parts in itself. In the first part, software test effort estimation was performed by applying the ML algorithm found in the WEKA program to the datasets of CocomoNasa, CocomoNasa-2, Cocomo-81.[10] Algorithms applied to the data sets were tested with 10-fold cross-validation technique. Correlation coefficient, error rates MAE, RAE were used as evaluation criteria. In the second part, feature selection performed on each data set in the WEKA program using Random Search, Genetic Search, Particle Swarm Optimization and Ranker. After the feature selection applied to the datasets, some attributes remove from the datasets. The number of selected features varies according to the applied method and datasets. In this way, the effects of feature selection methods on software test effort estimation were examined and the performances of the algorithms were calculated and compared.

Chapter 4, Discussion and Conclusion, the performance results of the ML algorithms were evaluated and compared.

CHAPTER II
RESARCH AND STUDY

2.1 SOFTWARE TEST

The main factor that determines the quality of software testing always lies in the definition of the software. Basically, the ultimate goal is perceived as making the software error-free at the end of the process. However, the main purpose of the test should be to understand whether the software successfully has all the functions of its creation purpose. [11] The purpose of the tests is to increase confidence in the software and its tasks. Sometimes these purposes can be confused. The purpose of the software testing process is to add value to the quality of the software as a product. The value added by testing shows itself as software quality and system reliability. Increasing reliability is achieved by finding and eliminating errors. Therefore, no software tester should test to show that the software is working. It should always accept as a prerequisite that there are errors in the software and strive to find as many errors as possible. It should also be acknowledged that no software is error-free.

If we need to make the software test definition again with all this infrastructure, "Software testing is the process of running the software to find bugs." [12].

There are different definitions of software testing in the literature.

- It is checking whether a certain number of test cases selected from an infinite set comply with a certain behavior. [13]
- Software testing; It is a process, or series of processes, done to ensure that the software does what it was designed to do and not what it was not expected to do. [14]
- Testing is the engineering concurrent lifecycle process that uses test software and makes necessary changes in order to improve and measure the quality of the tested software. [15]

The importance of software testing is increasing day by day in software projects. While software developers used to test their own codes, today many software developers adopt the independent test team model.

2.2 SOME EXAMPLES FOR IMPORTANCE OF SOFTWARE TESTING

In this section, the importance of software testing will be mentioned with some more concrete examples. In projects where software testing is not done enough, serious economic and physical problems may occur. Some examples that have been experienced before are below:

In 1983, due to a software error in the Soviet early warning system, III. World War had almost broken out. The Russians said the system launched five ballistic missiles from the United States. It turned out that the error was caused by a bug in the software that prevented satellite triggers from being detected as missiles by collecting the sun's rays reflected from the clouds. [16]

According to company plan, Denver Airport was scheduled to start service on 10 October 1993.Estimation software cost for Automatically luggage system was 186 million dollars at the beginning. However, due to many errors in this software, airport could not start to serve at the planned date. When airport started to serve on 28 February 1995, total calculated loss was 340 million dollars because of the delay. [17]

Sometimes, software errors cause dramatic results. An example of this result occurred during Gulf War on 1991.One of the American Patriot missiles missed its target and hit American soldiers barracks. Some of American soldiers died due to missed target. Based on the study findings, the Patriot system's operational duration exceeding 100 hours and a time discrepancy of 0.34 seconds resulted in a missile deviation of approximately 600 meters.

2.3 TYPES OF SOFTWARE TESTİNG

Software testing can be divided into 2 category as functional testing and non-functional testing.

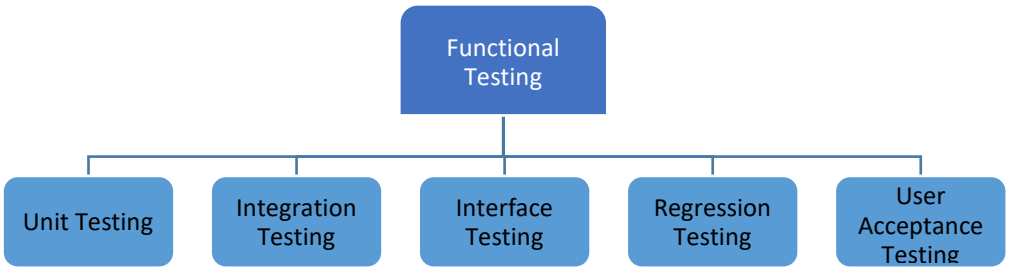

Figure 2.1: Type of Functional Testing

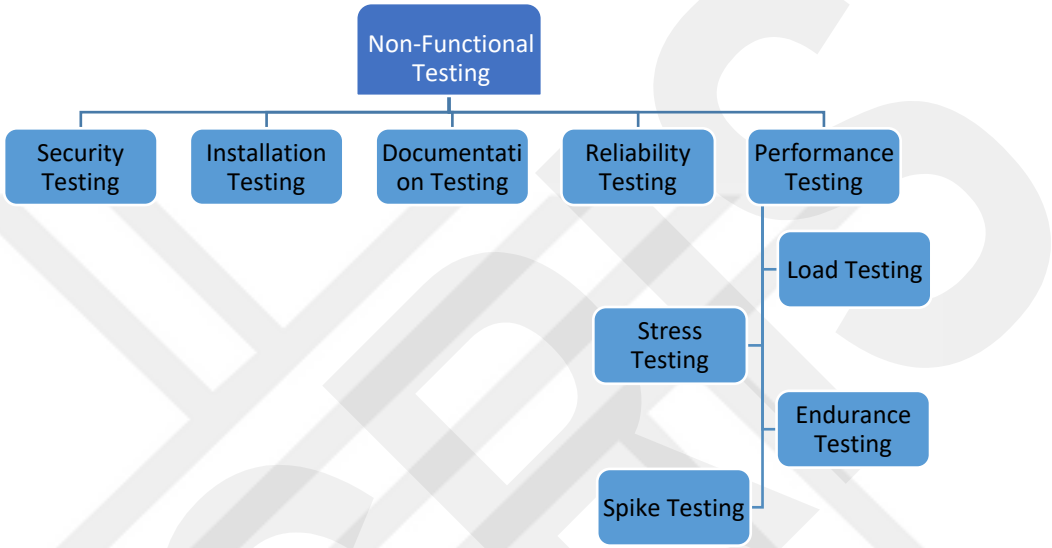

Figure 2.2: Type of Non-Functional Testing

2.3.1 FUNCTIONAL TESTING

Functional tests of a system encompass assessments of the system's ability to execute the required functions. These functions are determined by the functional requirements, which are typically outlined in various work products such as user requirements, epics, user stories, use cases, or functional specifications, reflecting the business needs.

Functions requirements define what the system should do. Functional testing should be done at all test levels (e.g. unit level functional tests needs), but the focus is different at each level. [18] Functional testing is not related to the source code of the application as it is used under Black Box testing. The focus when performing this test is always the user-friendliness of the main functions of the application. It is checked whether the requirements are met or not. It largely overlaps with user acceptance tests. Often the same test sets can be applied to both. System-level functional tests are used to control system behaviors that meet certain requirement specifications. All functional

requirements for the system must be fulfilled by the system. Functional tests are closed-box tests by nature. All functions should be tested. It should focus on the following goals:

- All allowed inputs must be accepted by the software. Unauthorized entries should be rejected.
- All possible system outputs should be examined.
- All system states and state transitions should be implemented and studied.
- It should be applied for all functions.

2.3.2 NON-FUNCTIONAL TESTING

Non-functional tests of a system evaluate aspects of systems and software, such as usability, performance, or security. Non-functional tests aim to evaluate the effectiveness of the system in fulfilling its intended purpose. It is important to note that non-functional testing should be conducted across all test levels, with regularity and an early initiation. Delayed identification of non-functional errors can pose significant risks to the project's overall success.

The aim is to determine whether the system is ready or not. The quality characteristics of the components or the system are tested. It is as important as functional testing in the quality and correct operation of the software. For example, how many users can use the system at the same time, is the system secure enough, and the system is tested to answer such questions.

Non-functional tests can be applied at all test levels. Non-functional testing usually considers the external behavior of the software

2.4 ACTIVITIES OF SOFTWARE TESTING

Test activities consist of such a process. The tests are run in accordance with the steps in this process: [19]

1. Test Planning: This is the step where the actions to be taken during the test are planned. Test objectives, strategy, environment and schedule are determined at this step.
2. Test Monitoring and Control: Test surveillance uses the test surveillance metrics defined in the test plan. In addition, in this step, the planned and actual progress are constantly compared.
3. Test Analysis: In the test analysis step, it is determined what is being tested.

4. Test Design: This step contains the preparation of the test environment, writing the test procedures and test cases.
5. Test Implementation: In this step, if necessary, test software is created or completed. Test procedures are created from test scenarios.
6. Test Execution: In this step, the tests are run manually or automatically. The actual test results are compared with the expected results.
7. Test Completion: Test completion activities are an important completion step in a project, such as the release of a software, the completion of a test project. This step also checks whether the error reports are turned off or not.

2.5 EFFORT ESTIMATION WITH SOFTWARE TESTING

Software testing effort is an estimate of the time and resources required to complete software tests. Test effort depends on the scope and complexity of the test project, the number of tests, the time and resources required to create and execute test cases, and other factors. [20]

The following steps are usually followed to calculate test effort:

1. The scope and complexity of the test project is determined. This includes the functionality of the software to be tested, use cases, technical requirements and other relevant factors.
2. Test scenarios and test cases are created. Test cases are a detailed design that defines how tests will be run and how their results will be evaluated. The creation of test cases plays an important role in determining the test effort.
3. The resources required for the execution of the test cases are determined. This includes the hardware, software and human resources required to run the tests.
4. Estimated timetable for execution of test scenarios and evaluation of results. This includes the time required to run the tests, the time required to debug and report the tests, and other factors.
5. Test effort is calculated by combining all these factors. Test effort is usually expressed in man hours and can vary depending on the number and complexity of test cases.

Accurate calculation of software test effort can make test planning and resource management more efficient and help test projects complete successfully.

2.6 MACHINE LEARNING ALGORITHMS

Machine learning is a branch of artificial intelligence used for computer systems to analyze data, recognize patterns, and learn from experience. This field uses algorithms and statistical models to enable computers to learn from data without human intervention. [21]

Machine learning aims to develop a system that learns to make inferences based on data, rather than a set of programmed instructions to perform a specific task. Data-driven learning enables a model to learn from experience and predict future data. The key elements of machine learning are:

• *Data*: For machine learning, processing and analysis requires large amounts of data. These data are used as training datasets and often contain human-labeled examples.

• *Algorithms*: Machine learning algorithms analyze data to discover patterns and relationships. These algorithms perform statistical analyzes on datasets, train the model, and are used to predict results.

• *Model Training*: Machine learning models are trained on data. During the training phase, a dataset is fed to the model and the model adjusts itself using algorithms to capture patterns in the data. The model learns statistical parameters to understand the relationship between input data and outputs.

• *Prediction and Results*: After the training process, the model can analyze new data and make predictions. For example, an image recognition model can analyze a new image to predict what objects it contains. These estimates can receive feedback for greater accuracy and can be used to improve the performance of the model.

Machine Learning operations can be examined under 3 main sections: supervised learning, unsupervised learning, and reinforcement learning.

1. Unsupervised Learning: Unsupervised learning uses unlabeled datasets. These datasets do not have output labels or targets. The algorithm analyzes data based on structures and patterns in the data and performs operations such as grouping, size reduction, or discovering hidden structure. K-means clustering, hierarchical clustering and dimension reduction methods (PCA, t-SNE) are examples of unsupervised learning algorithms.

2. Reinforcement Learning: Under this category, a model learns through experiences by interacting with an environment. The model is reinforced with a reward-punishment system to identify correct actions. Reinforcement learning

algorithms are often used in areas such as game theory, robotics and automatic control. Example algorithms include Q-Learning, Deep Q-Networks (DQN), and Actor-Critic.

3. Supervised Learning: Supervised learning works on labeled datasets. In these data sets, there is a relationship between the input samples and the target outputs. The algorithm tries to learn this relationship to catch patterns in the data and make output predictions for new input samples. [22] Supervised learning consist of many algorithms. But most know ones are: (SVM), decision trees, neural networks, and linear regression.

CHAPTER III
IMPLEMANTATION

3.1 MACHINE LEARNING ALGORITHMS

3.1.1 Random Forest

The Random Forest Algorithm is one of the collective learning algorithms. During the training, result clusters are constructed in accordance with statistical models with multiple decision trees and many variables. Basically, the result is decided by the average of these result clusters or their separation as clusters.

This supervised artificial learning algorithm examines historical data and tries to create trends with a predictive understanding. It uses decision trees as a classifier. The Random Forest Algorithm generates random decision trees. Randomness can be expressed in two ways:

1) Random selection of samples selected at the time of bagging.

2) Random selection of selected attributes for each decision tree.

The power of each decision tree classifier and their correlation with each other are the main indicators of error percentages in the results of the random forest classification algorithm. Random Forest Algorithm works effectively on large datasets. It can process thousands of data without the need for input changes, give estimates of important variables, and produce a constant generalization error as forest growth progresses.

It has an efficient method for estimating missing data. Even if large data rates are declining, there are ways to compensate for class error in datasets against unbalanced class populations. The Random Forest Algorithm has pioneered parallel applications using multi-threaded, multi-core and parallel architectures.

3.1.2 Linear Regression

Linear regression is a statistical modeling method used to explain the linear relationship of a dependent variable (outcome variable) with one or more independent

variables. Linear regression estimates the dependent variable using the first order function of the independent variables.

It is founded on the concept of identifying the optimal line or curve that characterizes the association between variables. This approach is frequently employed to forecast the value of the dependent variable by leveraging the values of the independent variables.

Linear regression models are based on the assumption that the relationship between dependent and independent variables is linear; this means that a change in the value of the independent variable is associated with a constant change in the value of the dependent variable. This relationship is typically represented by a straight line, so linear regression is called "linear" regression.

3.1.3 Bagging

Bagging (Bootstrap Aggregating) is an ensemble learning technique that is widely used in the field of machine learning. Bagging provides a stronger and more stable model by combining multiple learning models.

The Bagging algorithm creates different subsets using bootstrap, which is the sampling method. Bootstrap creates new datasets by recursively sampling from the dataset. Each subset is the same size and contains randomly sampled data from the original dataset. On each subset, individual models are trained using the basic learning algorithm (usually decision trees). These models capture different aspects of the dataset in different ways and make different errors.

After training is complete, each sub model independently predicts new inputs. In classification problems, class estimation is made with the voting method (majority vote), while in regression problems, the estimation is made by averaging the outputs of the sub-models.

One of the main advantages of bagging is that it reduces variance. By combining different sub models, the errors made by each one is reduced and a more stable model is obtained. It can also work effectively on large datasets thanks to its parallel computing capability.

Bagging is more effective especially when used with high variance learning algorithms such as decision trees. Random Forest, a popular implementation of Bagging, combines many decision trees to create a more powerful classification or regression model.

3.1.4 SMOreg

SMOreg is a regression algorithm that stands for Support Vector Machines for Regression. Regression analysis is a statistical technique that examines the relationship of a dependent variable with independent variables. SMOreg uses the Support Vector Machine method to perform this analysis.

Support Vector Machines is a machine learning algorithm that used in regression and classification analysis. SVM generates a hypothesis function to classify data points in a space. It draws a decision boundary (hyper plane) between classes and uses support vectors that best decompose this boundary.

SMOreg is SVM adapted for regression analysis. The aim is to create a new model in which the data points fit best with a line (regression line). This takes advantage of SVM to solve regression problems. It generally performs well on noisy and complex datasets. By using the advantages of SVM, models with high accuracy and generalizability can be created in regression problems.

3.1.5 Multilayer Perceptron

Multilayer Perceptron (MLP) is one of the most basic and widely used types of artificial neural networks. MLP is a feed forward neural network with many hidden layers. Its name means that it is multi-layered and each layer contains more than one perceptron.

MLP takes data to the input layer and passes it sequentially through one or more hidden layers. Each hidden layer contains a series of artificial nerve cells (perceptrons) or neurons. Each neuron weights the inputs, combines them, and passes them through an activation function. After the outputs are transmitted to the last layer, they are passed through a final activation function and create the final outputs.

The main purpose of MLP is to model complex relationships between data and make predictions. Weights are adjusted throughout the training process to reduce errors on the data and approach expected outputs. The backpropagation algorithm is used to calculate weight updates. This process is accomplished by calculating backwards the amount of error between actual outputs and expected outputs and optimizing the weights.

Advantages of MLP include modeling of complex functions through multiple layers, flexibility, ability to learn, ability to generalize, and adaptability to different

types of datasets. MLP is widely used to solve classification, regression, pattern recognition and many other machine learning problems.

However, MLP also has some disadvantages. The training process can take time, especially in large data sets and complex structures. It may also encounter overfitting problems and may require trial and error to determine the optimum model structure.

3.1.6 KStar

KStar is a classification algorithm based on the k-NN (k-Nearest Neighbors) algorithm. KStar is an improved version of the k-NN algorithm and performs better especially when working with categorical data.

KStar is an instance-based classification algorithm. It uses a measure of similarity or distance between data points to classify. KStar makes its classification decision based on the k nearest neighbors around a new data point.The KStar algorithm, unlike the k-NN algorithm, can be applied not only to numerical values, but also to categorical data. In order to process categorical data effectively, KStar uses weighted voting to determine the similarity measure between data points.

When classifying, KStar considers the classes of its neighbors to determine the class of a new data point. KStar categorizes the data point using either majority vote or weighted vote. By using weights proportional to the class of neighbors, a classification is made in which closer neighbors have more influence.

KStar stands out as a classification algorithm suitable for categorical data and can perform well, especially in classification problems. However, in situations such as large datasets or datasets with many features, the computational cost may increase.

3.1.7 Random Tree

Random Tree is an algorithm or a concept used in the fields of machine learning and data mining. It usually refers to an approach based on decision trees.

Random Trees is a classification or regression algorithm used to model patterns and relationships in datasets. This method represents the dataset with a tree structure and makes classification or prediction with successive decisions.

Random Trees basically consist of two components: randomness and tree structure. Randomness allows the algorithm to randomly sample during the training

phase. For example, training is performed for each tree by randomly selecting subset samples from the dataset. This allows the model to diversify and generalize better.

The tree structure is represented by decision nodes and branches based on the properties of the dataset. Each node takes the decision to split the data using a feature and a threshold value. These splitting decisions can be interpreted as rules representing the patterns and classes contained in the data set.

Random Trees are a widely used method for classification problems. It can also be applied for regression analysis. This algorithm can handle complex relationships in data and produce easily applicable and interpretable results.

Random Trees use techniques such as random sampling and feature selection to reduce model overfitting and increase generalization. It can also present information such as feature importance ranking, so it can be used to identify important features in the dataset. Random tree models have been extensively developed in the field of ML in recent years. [23]

3.1.8 M5P

M5P is used to perform regression analysis on datasets. [24] Regression analysis is a statistical technique that examines the relationship of a dependent variable with independent variables. The M5P algorithm uses the decision tree structure to analyze data and make predictions.

M5P solves regression problems using the structure of decision trees. The tree structure represents the regression functions of the features and target variable in the dataset. Each node makes regression estimates by dividing the data by the value or threshold value of a particular feature.

The M5P algorithm takes some measures to reduce overfitting while constructing the tree's structure. For example, it imposes restrictions to control the tree's size and prunes branches as needed. This makes the model more generalizable and better fit to new data.

The M5P algorithm is considered a flexible and effective method for solving regression problems. It performs particularly well when working with properties with numeric values. At the same time, the models obtained thanks to the decision tree structure are easy to interpret.

To summarize, M5P is a machine learning algorithm that uses decision tree structure to perform regression analysis. It is used to make regression predictions on datasets and attempts to model patterns and relationships in data using tree structure.

3.1.9 IBK

IBk represents the implementation of the k-NN (k-Nearest Neighbors) algorithm in the Weka library. The k-NN algorithm is an instance-based learning algorithm used in classification and regression problems. This algorithm uses the k nearest neighbors around a data point to determine its class or value.

This algorithm makes classification or regression predictions using a measure of similarity between data points. To classify the data point, it considers the class of its k nearest neighbor and uses the majority vote method.

The IBk algorithm can be customized with various parameters that Weka provides. For example, the value of k, the number of neighbors, can be determined by the user. It also provides options for choosing different measures of similarity or distance.

Weka has an easy to use interface and offers many different machine learning algorithms. IBk is an option that implements the k-NN algorithm among these algorithms. Weka supports classification, regression, clustering, feature selection, and many other machine learning tasks.

In conclusion, Weka's IBk algorithm represents the implementation of the k-NN algorithm in the Weka library. It is used to classify data points or make regression estimates and uses information from the nearest neighbors around it.

3.2 EVALUATION CRITERIA

In this thesis, correlation coefficient, MAE and RAE were used as evaluation criteria.

3.2.1 Correlation Coefficient

The correlation coefficient indicates the strength and direction of the link between two different variables. Correlation indicates the relationship between the variables and the correlation coefficient indicates the state of the relationship. The correlation coefficient can take a number between −1 and 1 depending on the condition of the relationship

The negative value of the number indicates that there is an inverse relationship between the variables. That is, as one variable increases, the other decreases. The positive value of the number indicates that there is a linear relationship between the variables, that is, as one variable increases, the other also increases. If the number value indicating the correlation coefficient is zero, it means that there is no relationship between the two variables. It is understood that the correlation between the variables increases as the correlation coefficient approaches 1 and decreases as it approaches 0.

There is a range for -1 to 1 and the explanations for the value ranges it receives are as follow.

Range of Correlation Coefficient Values	Level of Correlation	Range of Correlation Coefficient Values	Level of Correlation
0.80 to 1.00	Very Strong Positive	-1.00 to -0.80	Very Strong Negative
0.60 to 0.79	Strong Positive	-0.79 to -0.60	Strong Negative
0.40 to 0.59	Moderate Positive	-0.59 to -0.40	Moderate Negative
0.20 to 0.39	Weak Positive	-0.39 to -0.20	Weak Negative
0.00 to 0.19	Very Weak Positive	-0.19 to -0.01	Very Weak Negative

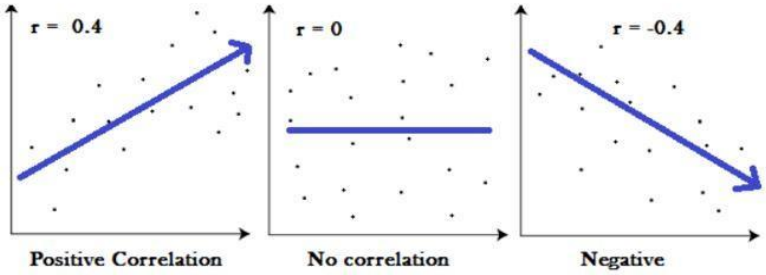

Figure 3.1: Correlation Coefficient

3.2.2 Mean Absolute Error

MAE (Mean Absolute Error) is the error rate that gives the average of the difference between the actual values and the predicted value.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |x_i - x| \qquad (3.1)$$

Where:

n = the number of errors,

Σ = summation symbol,

|xi – x| = the absolute errors.

3.2.2.1 Importance of Mean Absolute Error

Mean Absolute Error (MAE) is used to evaluate the accuracy of forecasts. Some of its most important features are that it is easy to understand, interpretable and reliable. It is a very important performance statistic for regression models because it is a tool based on these features. Among the many reasons, these are the most significance.

- All individual differences are given same weight on the average. This makes it easy to compare the performance of several models or variations of the same model.
- The MAE interpretation is a basic and obvious statistic that represents the average size of forecast errors. It is simple for non-technical stakeholders to understand.
- Resistance to outliers. MAE is not as affected by extreme results as other metrics such as Mean Squared Error (MSE). This makes it a suitable measure for datasets with outliers or extreme values.

3.2.3 Relative Absolute Error

Relative absolute error gives a sum of the difference between the predicted values and the actual values, dividing it by the sum of the difference between the true value and the mean of the true value.

$$RAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{\sum_{i=1}^{n}|y_i - \bar{y}|} \quad (3.2)$$

where:

n: represents the number of observations

yi: represents the realized value

ŷi: represents the predicted value

ȳ: represents the average of the realized values

3.3 PLATFORM

Weka has a modular design and can perform operations on data sets and data mining with its features. Weka stands for Waikato Environment for Knowledge Analysis. Weka software comes with a unique ".arff" extension support. However, within the Weka software, there are tools for converting CSV files to ARFF format.

The application part of this thesis research was carried out on the Weka platform. Version 3.8.6 is used as the Weka platform.

There are five different interfaces in Weka as Figure 3.2. these areas are separated according to the study areas. These workspaces are Explorer, experimenter, knowledge Flow, WorkBench, and Simple CLI, as seen below. This thesis work was done in the Explorer menu. On the main screen, the Explorer button is pressed and the Explorer menu opens.

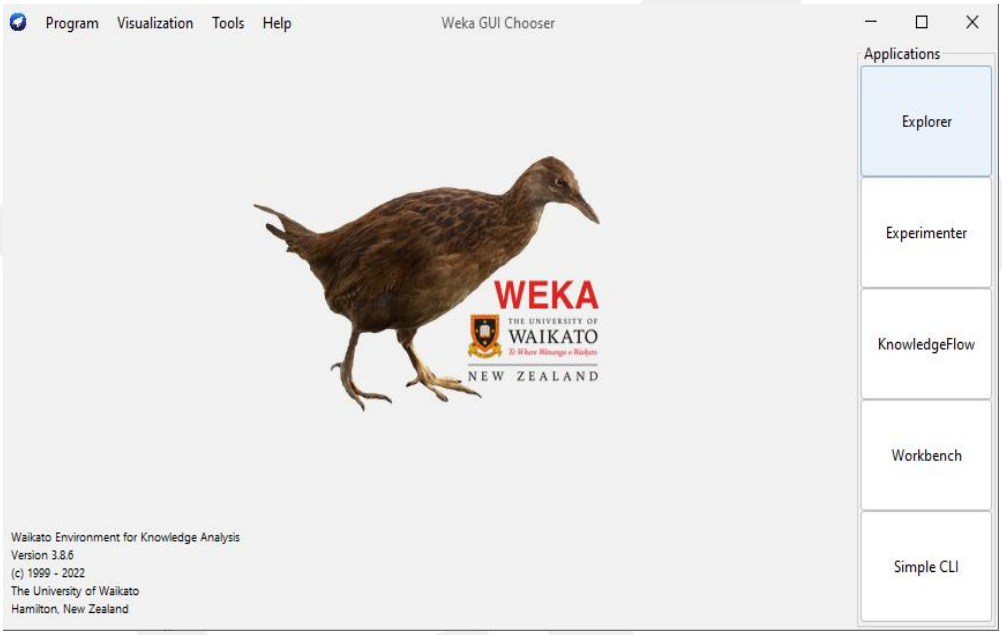

Figure 3.2: Home Page of WEKA

In the window that opens, click the Open File button as Figure 3.3.

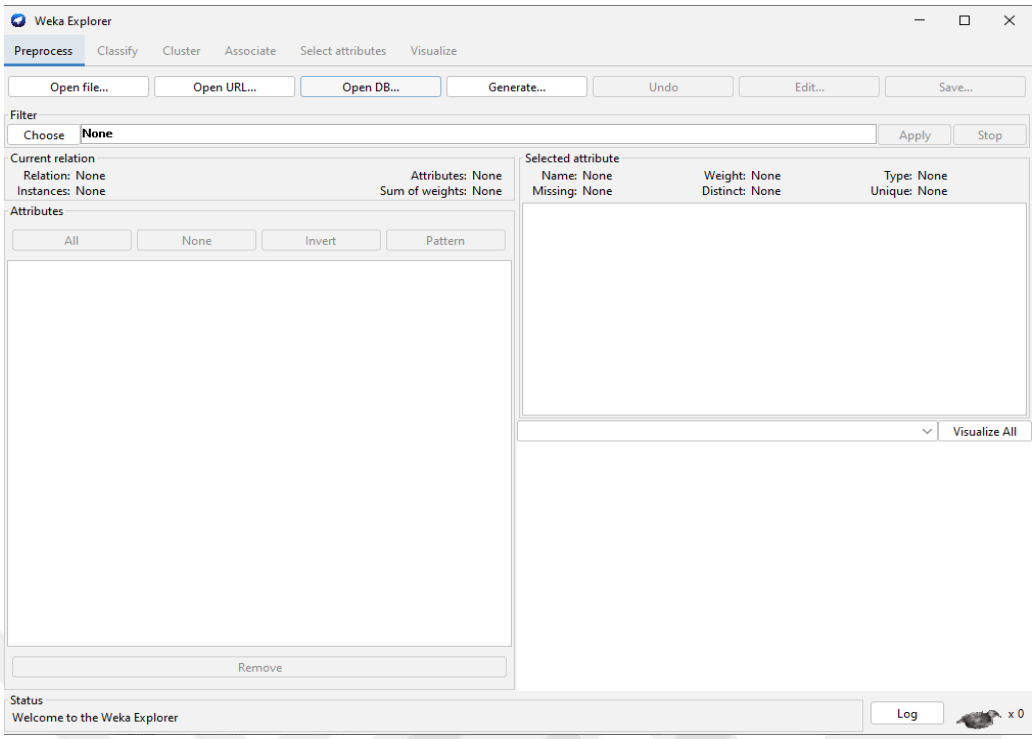

Figure 3.3: Explorer Page of WEKA

The dataset to be studied is selected from the local computer as Figure 3.4.

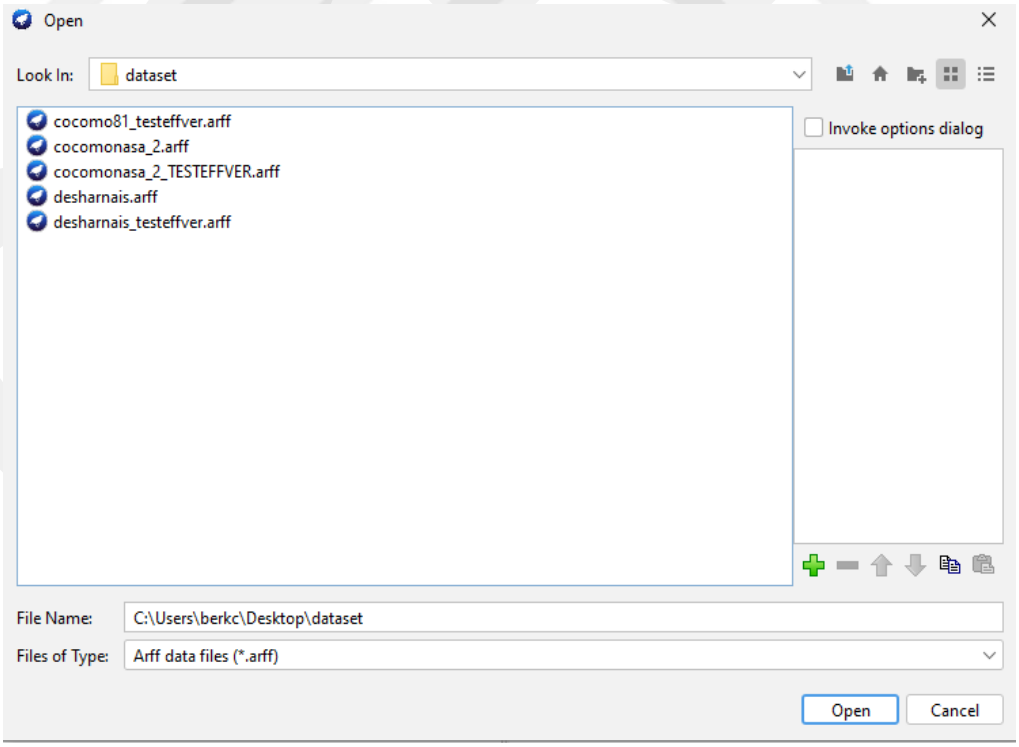

Figure 3.4: Dataset Selection on WEKA

In order to run the ML algorithms on the dataset, the relevant algorithm is selected from the window in Figure 3.5 and the Start button is clicked. The algorithm selected on the data set is run with default settings.

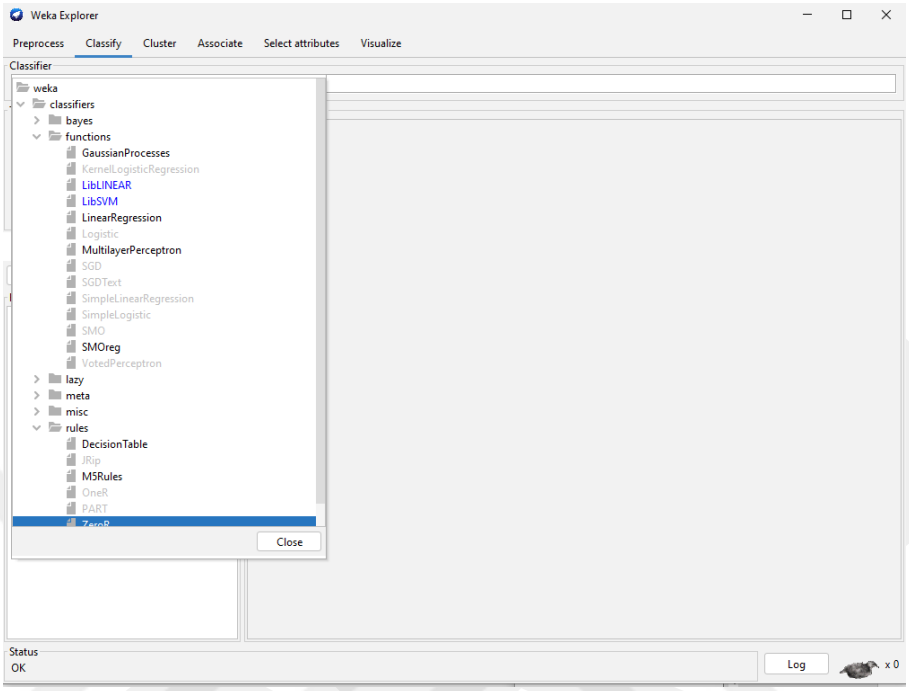

Figure 3.5: Algorithm Selection on WEKA

After start button clicked, the result is seen as Figure 3.6.

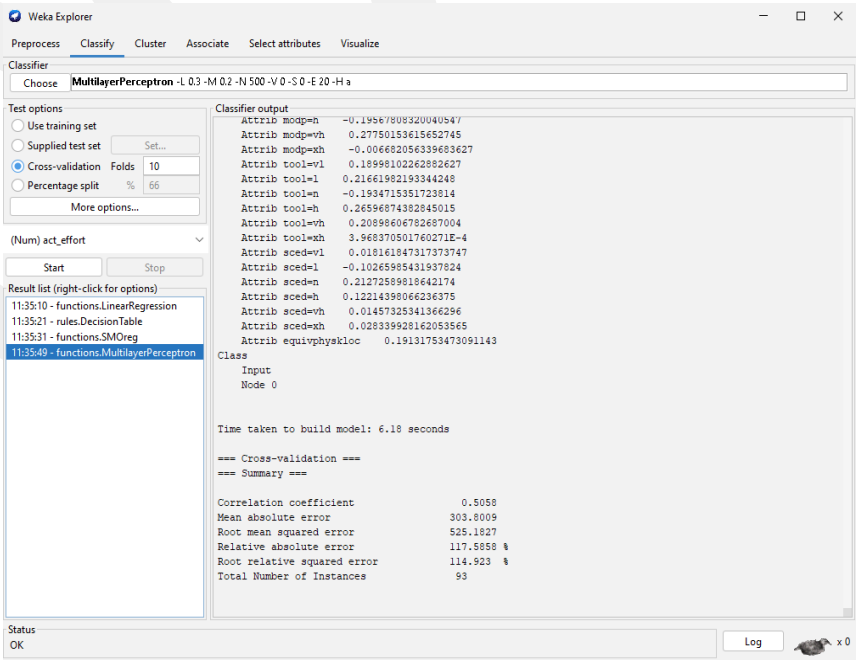

Figure 3.6: Result Screen of Classify

It is the Select attributes menu that enables the selection of attributes on data sets using the WEKA program. When the Select attributes menu is clicked, the Select attributes window in Figure 3.7 opens. Attribute selection method and search method are selected from the Select attributes window.

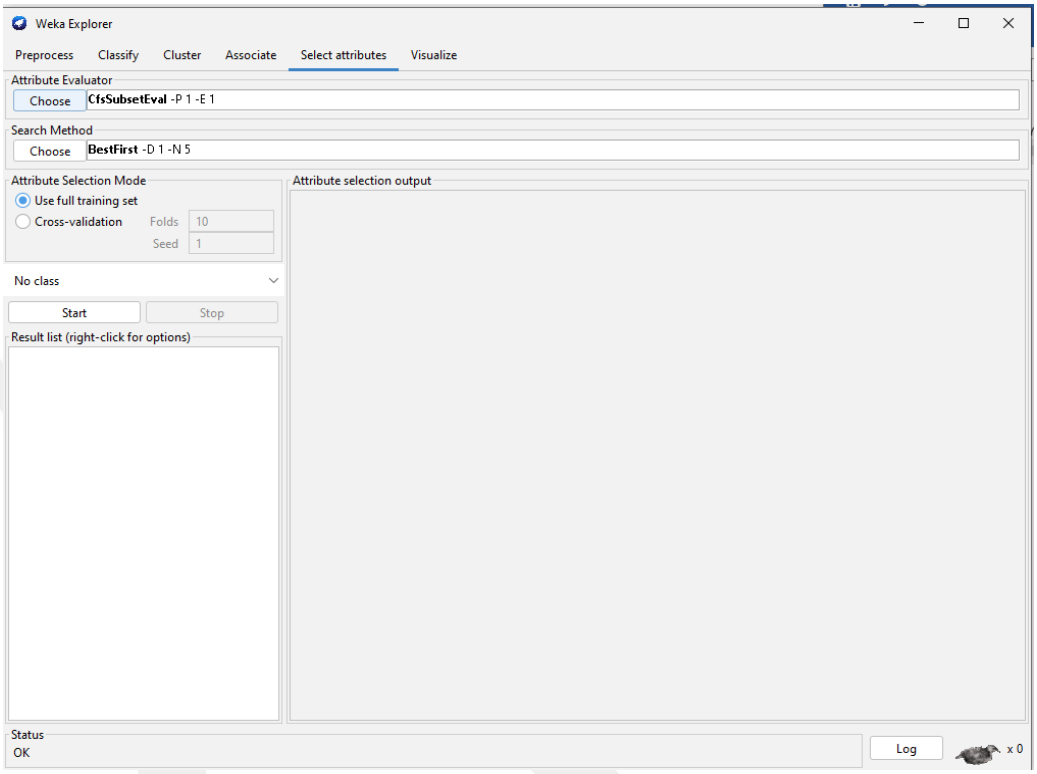

Figure 3.7: Attribute Selection on WEKA

Throughout this study, studies on hybrid methods were carried out by using different attribute evaluation selection and different search method selections. For example, in Figure 3.8, CfsSubsetEval is used as attribute evaluator and PSO search is used as Search Method. As a result, 3 attributes were not selected. The model was created with the remaining 21 attributes instead of 24 attributes.

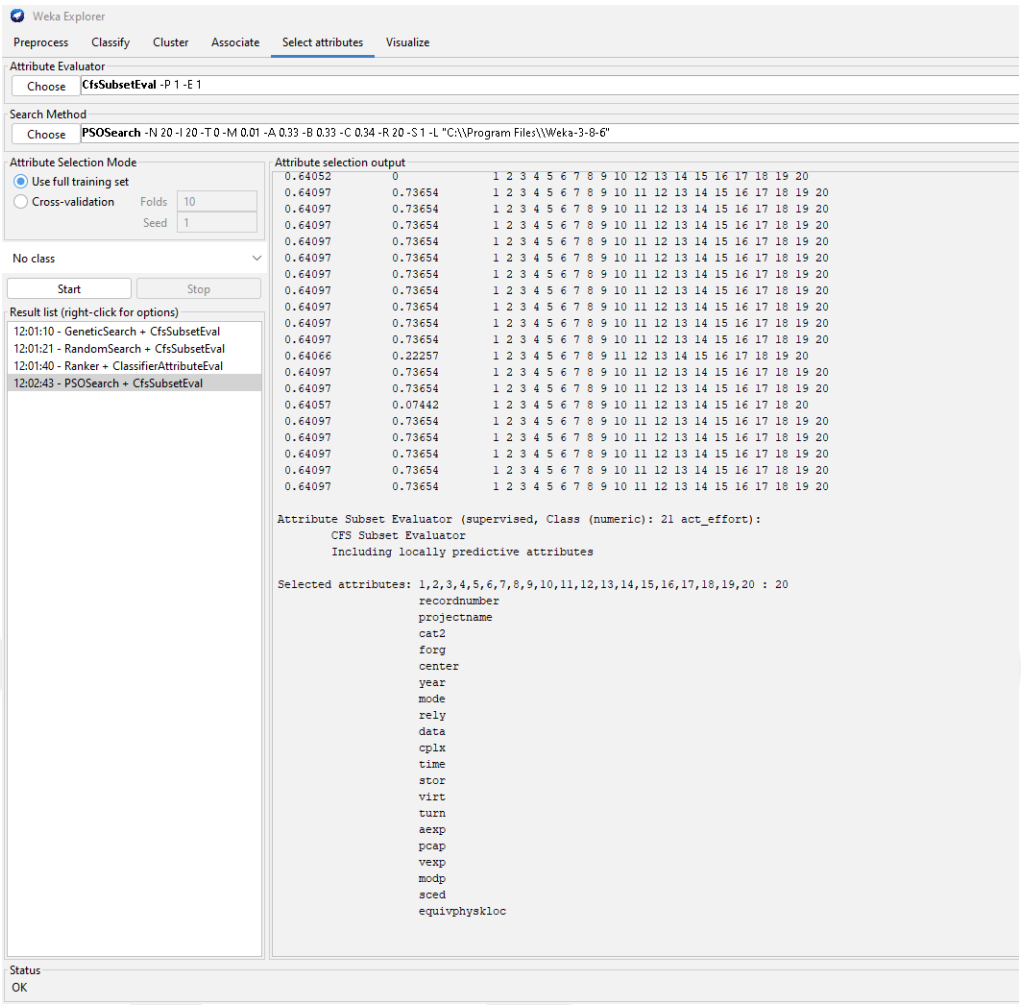

Figure 3.8: Result of Attribute Selection on WEKA

3.4 METHODS

3.4.1 Evaluator Methods

3.4.1.1 Correlation Based Feature Selection (CFS)

Correlation-based Feature Selection is an algorithm for feature selection and can be implemented with software tools such as Weka. CFS uses a correlation-based metric to evaluate the effects of features on classification performance. Its operation can be summarized in the following steps:

1. Dataset preparation: In the first step, Properties (variables) and target variable (class) must be determined.
2. Calculation of correlation between features: First, the correlation of each feature with the class effect is calculated. Then, the relationships between the features are also evaluated. The correlation of traits with each other is measured by considering the class effect.

3. Calculation of feature scores: Each feature is assigned a score, which is correlated with the feature's classification performance.
4. Ranking and selection of features: After the scores are calculated, the features are ranked. CFS encourages selection of the highest rated properties. This enables the selection of more important and relevant features from an informatic point of view.
5. Filtering out irrelevant features: CFS filters out features that have low impact on classification performance or unnecessary.

Using these steps, CFS evaluates the correlation between features and tries to select important features. Feature selection helps build more efficient and performant machine learning models with less dimensional datasets.

3.4.1.2 Classifier Attribute Evaluator

Classifier Attribute Evaluation is a feature selection algorithm and evaluates each feature based on the performance of the classifier model. A classifier model is used to determine the effect of features on classification, and this model evaluates features in order of importance.

In this method, a metric is used for the performance of the classifier model. For example, metrics such as accuracy, precision, precision, or sensitivity can be used. The contribution of each feature to the performance of the model is expressed as the feature evaluation score or degree of importance.

Classifier Attribute Evaluation offers the benefits of feature selection with the classifier model. This method filters out unnecessary or noisy features in the dataset while highlighting the more informative or more important features. The basis of the algorithm is to create a good subset of features with features that are highly correlated with the output class. [25] As a result, a less dimensional subset of features is created and the performance of the model can be improved.

3.4.1.3 Correlation Attribute Evaluator

Correlation Attribute Evaluation is a method that evaluates the correlation of features with the target variable or classification effect when selecting features. Correlation Attribute Evaluation is a feature selection algorithm and evaluates each feature based on its correlation with the target variable. Correlation is a statistical

measure that measures the relationship between the characteristics and the target variable.

In this method, the correlation of each feature with the target variable is calculated. Positive correlation indicates that the feature behaves similarly to the target variable, while negative correlation indicates that the feature is inversely related to the target variable. If the correlation is nearly zero, it indicates that the relationship of the feature with the target variable is weak.

Correlation Attribute Evaluation evaluates features based on their correlation scores or severity ratings. Features with high correlation scores are considered important features that provide more information in classification and increase the performance of the model.

3.4.1.4 Relief Attribute Evaluator

Relief Attribute Evaluation is a feature selection algorithm and uses the Relief algorithm to determine the effect of features on classification. This method determines the importance of each feature by measuring the differences between classes.

The relief algorithm measures the discriminating power of features between classes. For this purpose, similarity and difference scores between close neighbors for each sample are calculated. For example, if an instance has close neighbors to two different classes, that instance may have a property that discriminates between those classes.

Relief Attribute Evaluation evaluates features using the Relief algorithm. Discrimination scores are calculated for each feature and the features are ranked according to their importance. Features with a high discriminative power score are considered more effective and informative in classification.

3.4.2 Search Methods

Four search methods were used in this thesis study, details of methods are given in following section.

3.4.2.1 Random Search

Random Search is a method for hyperparameter tuning in machine learning models. Hyperparameters are user-specified adjustable parameters that affect the performance of a machine learning algorithm. Random Search does random trials to

find the best combination of hyperparameters. This method, after defining parameter ranges and possible values, trains and evaluates a model by randomly selecting values from these ranges. This process is repeated for a certain amount of time or number of attempts. As a result, the hyperparameter combination that provides the best performance is selected and the model is retrained with these values.

However, the downside of Random Search is that it doesn't use a specific optimization strategy like other optimizer algorithms. Therefore, this method may sometimes not guarantee the best performance. However, it can be an effective option in situations where computational resources are limited or there is no obvious relationship between hyperparameters.

3.4.2.2 Particle Swarm Optimization (PSO) Search

PSO is a natural meta-heuristic optimization method. This method mimics a group of particles in nature working together to discover a target. PSO aims to find the best solution by moving many solution candidates in the potential solution area. One of the widely accepted fundamental benefits of metaheuristic algorithms is that they provide mechanisms to solve large or intractable problems in reasonable execution times while the exact algorithms fail to succeed due to time limitations. [26]

Weka's PSO algorithm is often used for hyperparameter tuning in machine learning models. In machine learning models, the correct setting of hyperparameters significantly affects the performance and generalization ability of the model. The PSO algorithm updates the motions and positions of particles to optimize a target function. Each particle tracks the best available position (best solution) and the best position of all particles (global best solution). This teamwork allows the particles to perform a search in the solution space.

3.4.2.3 Genetic Search

Genetic Search in Weka can be used to optimize for machine learning models such as feature selection or hyperparameter tuning. Genetic Search is a population-based search method based on genetic algorithm principles. Below is a general explanation of how Genetic Search works:

1. Population generation: In the first step, an initial population is randomly generated. Each individual expresses a solution represented as genes. Genes represent a particular combination of traits or hyperparameters.

2. Calculation of fitness value: Each individual is evaluated using a fitness function. The fitness function is used to measure how well an individual is performing.
3. Selection: Individuals with high fitness are transferred to the next generation using the selection operator.
4. Crossover: Crossover operation is performed between selected individuals. Crossover creates new individuals by combining the genetic material of two individuals. This increases genetic diversity and potentially helps discover better solutions.
5. Mutation: Mutation process can be applied in newly created individuals. Mutation enables the discovery of new solutions by making random changes in the genetic material of individuals. This helps to explore a potentially wider search area.
6. Iteration: Starting from the second step, the fitness value is calculated and new generations are created. The genetic algorithm process continues by using selection, crossover and mutation operators. This process continues until the iteration count or stopping criterion is met.

As a result, the Genetic Search algorithm in Weka makes feature selection using the principles of genetic algorithms. This method aims to make the dataset less dimensional and to find better performing feature combinations.

3.4.2.4 Ranker

"Ranker" is a term used as a feature selection or ranking method. Ranker evaluates features in the dataset and ranks them in order of importance.

A ranker rates each feature with a point or importance value. These scores may be based on the classification performance of the features or their relationship to a target variable. Higher rated features are considered more important or more informative.

Ranker methods are used for feature selection in machine learning models. Feature selection is used to filter out redundant or noisy features in the dataset or to create a less dimensional feature subset. The ranker performs this selection by determining their importance in classification or prediction. Ranker methods are a widely used tool in feature selection and ranking problems. Ranking the features in

order of importance provides better understanding and can give the model a better generalizability.

3.5 DATASET

In this section, the datasets used in the research will be discussed in detail. Three different datasets were used for this study the data of real software projects are kept in the data sets obtained from the data warehouse. There are linked and independent attributes in datasets, each of which contains a different number of project data. These attributes are used to perform the test effort cost estimation. If an attribute gives the true cost value, the linked attribute; If it gives cost-related values, it is called an independent attribute. The independent attributes in the data sets determine the value of the linked attribute.

The datasets used are publicly available datasets. The names of the datasets used are CocomoNASA/Software cost estimation, COCOMONASA 2 / Software cost estimation, Cocomo-81/Software cost estimation.

3.5.1 CocomoNasa / Software Cost Estimation:

The dataset consists of 60 NASA projects. These projects took place in different locations between 1980 and 1990. There are sixty instances and seventeen attributes. The Name of attributes are analyst's capability, programmer's capability, application experience, modern programing practices, use of software tools, virtual machine experience, language experience, schedule constraint, main memory constraint, data base size, time constraint for CPU, turnaround time, machine volatility, process complexity, required software reliability, line of code measure, actual effort in person/months.

Table 3.1: Attribute Selection for CocomoNasa Dataset

Attribute Name	Description
acap	analyst's capability
pcap	programmer's capability
aexp	application experience
modp	modern programing practices
tool	use of software tool
vexp	virtual machine experience
lexp	language experience
sced	schedule constraint
stor	main memory constraint
data	data base size
time	time constraint for CPU
turn	turnaround time
virt	machine volatility
rely	required software reliability
cplx	process complexity
loc	line of code measure
act_effort	actual effort

3.5.2 CocomoNasa2 / Software Cost Estimation:

The dataset consists of 93 NASA projects. These projects took place in different locations between 1971 and 1987.There are 93 instances and 24 attributes. 15 of them are standard COCOMO-I discrete attributes and 7 of them are describing the project, one of them is lines of code measure, and last one is actual effort in person months. Name of attributes are analysts capability, programmers capability, application experience, modern programing practices, use of software tools, virtual machine experience, language experience, schedule constraint, main memory constraint, data base size, time constraint for CPU, turnaround time, machine volatility, process complexity, required software reliability, line of code measure, actual effort in person/months, record number, project name, category of application, flight or ground system, which NASA center, year of development, development mode.

Table 3.2: Attributes Description for CocomoNasa-2 Dataset

Attribute Name	Description
record number	real unique id
project name	project name
cat2	category of application (Avionics, application ground, avionics monitoring….)
forg	flight or ground system
center	which NASA center
year	year of development
mode	development mode (embedded, organic, semidetached)
acap	analyst's capability
pcap	programmer's capability
aexp	application experience
modp	modern programing practices
tool	use of software tool
vexp	virtual machine experience
lexp	language experience
sced	schedule constraint
stor	main memory constraint
data	data base size
time	time constraint for CPU
turn	turnaround time
virt	machine volatility
rely	required software reliability
cplx	process complexity
loc	line of code measure
act_effort	actual effort

3.5.3 Cocomo-81 / Software Cost Estimation:

There are 63 instances and 17 attributes. Name of attributes are analyst's capability, programmer's capability, application experience, modern programing practices, use of software tools, virtual machine experience, language experience, schedule constraint, main memory constraint, data base size, time constraint for CPU, turnaround time, machine volatility, process complexity, required software reliability, line of code measure, actual effort in person/months.

Table 3.3: Attributes Description for CocomoNasa-81 Dataset

Attribute Name	Description
acap	analyst's capability
pcap	programmer's capability
aexp	application experience
modp	modern programing practices
tool	use of software tool
vexp	virtual machine experience
lexp	language experience
sced	schedule constraint
stor	main memory constraint
data	data base size
time	time constraint for CPU
turn	turnaround time
virt	machine volatility
rely	required software reliability
cplx	process complexity
loc	line of code measure
act_effort	actual effort

3.5.4 Summary for Datasets

The summary dataset table is below as Table 3.4.

Table 3.4: Summary for Datasets

Data Set	Instance	Attribute Number	Measurement Unit	Effort
CocomoNasa	60	17	Loc	Man/ Month
CocomoNasa-2	93	24	Loc	Man/ Month
Cocomo-81	63	17	Loc	Man/ Month

3.5.5 The Numeric Values of the Effort Multipliers

The largest and smallest value ranges that the attributes in the data sets can take were examined.

Table 3.5: The Numeric Values of the Effort Multipliers

Attributes	Very Low	Low	Normal	High	Very High	Extra High	Productivity
acap	1.46	1.19	1	0.86	0.71		2.06
pcap	1.42.	1.17	1	0.86	0.7		1.67
aexp	1.29	1.13	1	0.91	0.82		1.57
modp	1.24.	1.1	1	0.91	0.82		1.34
tool	1.24	1.1	1	0.91	0.83		1.49
vexp	1.21	1.1	1	0.9			1.34
lexp	1.14	1.07	1	0.95			1.2
sced	1.23	1.08	1	1.04	1.1		e
stor			1	1.06	1.21	1.56	-1.21
data		0.94	1	1.08	1.16		-1.23
time			1	1.11	1.3	1.66	-1.3
turn		0.87	1	1.07	1.15		-1.32
virt		0.87	1	1.15	1.3		-1.49
rely	0.75	0.88	1	1.15	1.4		-1.87
cplx	0.7	0.85	1	1.15	1.3	1.65	-2.36

3.6 FINDINGS

It is frequently mentioned in the literature that a certain percentage of software effort data can be used for test effort estimation [27]. Due to this situation, 40% of the software effort data was taken. [28] According to the results, the new data formed was accepted as a software test effort.

In this research, model trainings were carried out on three data sets by using different machine learning algorithms. These algorithms are, respectively, linear regression, random forest, bagging, multilayer perceptron, SMOreg, M5P, IBk, KStar and random tree. Datasets are randomly divided into training and test data using 10-fold cross validation technique. The created models evaluated according to correlation coefficient, error rate MAE and RAE. In WEKA, the default values for the 3.8.6 version of these algorithms are used: The values used in this study from the adjustable parameters for selected algorithms are as follows:

- For Linear Regression, attributeSelectionMethhod parameter is selected as M5 Method.
- For the MLP algorithm, the "hiddenLayers" parameter is selected as "a". This means that the number of hidden layers and the number of neurons are automatically determined based on the data. LearningRate is 0.3 and momentum is 0.2.
- The SMOReg complexity parameter c 1 is selected. FilterType is Nomalize training data, Kernel is PolyKernel, and regOptimzer is RegSMOImproved.
- KNN 1, distanceWeighting No distance weighting is selected in IBk.
- KStar globalBlend 20, missingMode Avarage entropy curves are selected.
- For Bagging, the classifier REPTRee is selected, numExecutions used to set up the ensemble model are 1, and numerations are 10.
- In M5P, 4 is selected as the minimum instance to be accepted for the leaf node.
- In RandomForest, a value of 0, which represents no limit, is processed for maxDepth. numIterations 100, numExecutions 1 used to set up the ensemble model is selected.
- In RandomTree, minNum 1, which represents the total weight of the instances in the leaf, is selected, and for maxDepth, 0, which represents no limit, is selected.

The effort estimation of the software projects was carried out in two parts by using the ML algorithms in the WEKA environment. In the first part; The default

settings of WEKA environment algorithms are preferred. No attribute selection has been implemented for each algorithm used. In the second part, hybrid methods are tried with Select Attributes on Weka and the results are tested for all algorithms and datasets one by one. The following hybrid methods were tried respectively.

- CfsRandomEvaluater + RandomSearch
- CfsRandomEvaluater + PSOSearch
- CfsRandomEvaluater + GeneticSearch
- ClassifierAttributeEvaluation + Ranker
- CorrelationAttributeEvaluation + Ranker
- ReliefAttributeEvaluation + Ranker

As a result, the results obtained in the first part and the second part were compared.

3.6.1 Models with Original Datasets

For software test effort estimation, performance measurements of machine learning algorithms applied to the CocomoNasa CocomoNasa-2 ve Cocomo-81 are given. No attribute selection is made and the original data is used.

3.6.1.1 CocomoNasa Dataset

According to Table 3.6, M5P give the best result. The correlation coefficient is 0.922, MAE 60.3936 and RAE 35.0178 % for M5P.

The worst performance is Random Tree. The correlation coefficient is 0.3915, MAE 137.1558 and RAE 79.5265 % for Random Tree.

Table 3.6: Models Result for CocomoNasa Dataset

CocomoNasa			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	0.7994	98.8186	57.2976
Random Forest	0.7281	93.4642	54.193
Bagging	0.8083	74.0699	42.9476
Multilayer Perceptron	0.8931	71.781	41.6205
SMOreg	0.7178	99.5863	57.7427
IBk	0.5768	118.1707	68.5184
KStar	0.6772	88.1806	51.1294
Random Tree	0.3915	137.1558	79.5265
M5p	0.922	60.3936	35.0178

3.6.1.2 CocomoNasa-2 Dataset

According to Table 3.7, KStar give the best result. The correlation coefficient is 0.7437, MAE is 146.6103 and RAE is 56.7453 % for KStar.

The worst performance is Linear Regression. The correlation coefficient is -0.3101, MAE is 258.3653 and RAE is 100 %.

Table 3.7: Models Result for CocomoNasa-2 Dataset

CocomoNasa-2			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	-0.3101	258.3653	100
Random Forest	0.667	164.1404	339.9489
Bagging	0.4241	195.6914	75.7421
Multilayer Perceptron	0.5058	303.8009	117.5858
SMOreg	0.4176	307.0868	118.8576
IBk	0.6581	198.9049	76.9859
KStar	0.7437	146.6103	56.7453
Random Tree	0.637	201.0081	77.8
M5p	0.7092	147.0593	56.9191

3.6.1.3 Cocomo-81 Dataset

According to Table 3.8, Random Forest give the best result. The correlation coefficient is 0.7529, MAE is 215.6668 and RAE is 59.421 %.

The worst performance is IBK. The correlation coefficient is 0.0768, MAE is 313.021 and RAE is 86.2442 %.

Table 3.8: Models Result for Cocomo-81 Dataset

Cocomo-81			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	0.6102	349.7908	96.3751
Random Forest	0.7529	215.6668	59.421
Bagging	0.4622	268.7705	74.0522
Multilayer Perceptron	0.6739	264.9429	72.9976
SMOreg	0.6625	191.8657	52.8632
IBk	0.0768	313.021	86.2442
KStar	0.5621	210.9438	58.1197
Random Tree	0.6439	226.4391	62.389
M5p	0.6843	206.9436	57.0175

3.6.2 Models with Attribute Selection

3.6.2.1 CocomoNasa Dataset with Attribute Selection

Attribute selections is made before applying the selected ML algorithms to the CocomoNasa datasets.

3.6.2.1.1 CFS + Random Search for CocomoNasa

According to Table 3.9, CfsSubsetEval and Random Search have been applied to the CocomoNasa dataset under the select attributes menu of Weka. In this case, 10 out of 17 attributes was selected. The selected attributes are Rely, Data, Time, Stor, Turn, Lexp, Modp, Tool, Loc, Act. Effort.

According to Table 3-9, Multilayer Perceptron gave the best performance. The correlation coefficient is 0.9245, MAE is 57.5215 and RAE 33.3525 %. The worst performance is IBK. The Correlation Coefficient is 0.5511, MAE is 118.1307 and the RAE is 68.4952.

Compared with the original dataset, Correlation Coefficient of Random Tree increased by 44%. Even if not the best result, it is the model that showed the most improvement. When comparing the all algorithms, the CFS + Random Search model showed improvement in 5 out of 9 algorithms compared to the original model.

Table 3.9: CFS + Random Search for CocomoNasa

CocomoNasa / CFS + Random Search			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	0.7782	105.5529	61.2023
Random Forest	0.7281	93.4642	54.193
Bagging	0.8111	74.6154	74.6154
Multilayer Perceptron	0.9245	57.5215	33.3525
SMOreg	0.7691	90.0964	52.2402
IBk	0.5511	118.1307	68.4952
KStar	0.7818	75.6172	43.8448
Random Tree	0.8331	74.3994	43.1387
M5p	0.903	69.6416	40.38

3.6.2.1.2 CFS + PSO for CocomoNasa

According to Table 3.10, CfsSubsetEval and PSO have been applied under the select attributes menu of Weka. In this case, 12 out of 17 attributes were selected. The selected attributes are Rely, Data, Time, Stor, Turn, Virt, Vexp, Lexp, Modp, Tool, Loc, Act_effort.

According to Table 3.10, M5P gave the best performance. The correlation coefficient is 0.9021, MAE is 70.3909 and RAE is 40.815%. There is no change in the best result comparison with the original model. However, the model created with the Random tree achieved 36% improvement over the original model. However, it did not achieve the best result. The worst performance is IBk. The Correlation Coefficient is 0.5504, MAE is 120.4973 and RAE 69.8675 %.

Table 3.10: CFS + PSO for CocomoNasa

CocomoNasa / CFS + PSO			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	0.7396	135.8567	78.7732
Random Forest	0.8115	80.0985	46.4432
Bagging	0.8103	74.934	43.4487
Multilayer Perceptron	0.8956	64.8591	37.607
SMOreg	0.6779	113.7726	65.9683
IBk	0.5504	120.4973	69.8675
KStar	0.7592	79.0905	45.8587
Random Tree	0.7531	87.2165	50.5704
M5p	0.9021	70.3909	40.8145

3.6.2.1.3 CFS + Genetic Search for CocomoNasa

According to Table 3.11, CfsSubsetEval and GS have been applied under the select attributes menu of Weka. In this case, 10 out of 17 attributes were selected. The selected attributes are Rely, Data, Time, Stor, Virt, Turn, Lexp, Tool, Loc, Act_effort.

According to Table 3.11, M5P gave the best performance. The correlation coefficient is 0.9118, MAE is 67.1147 and RAE is 38.914 %. The worst performance is Random Tree. Correlation Coefficient is 0.3915, MAE is 137.1558 and RAE is 79.5265 %.

Although M5P gave the best results, the success rate decreased when compared to the original dataset. Even if it didn't give the best results, KStar was the algorithm that showed the most improvement when CFS+GA was applied. When compared to the original model, an improvement of 10% was observed.

Table 3.11: CFS + PSO for CocomoNasa

CocomoNasa / CFS + GS			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	0.7648	111.1063	64.4223
Random Forest	0.8193	81.1336	47.0434
Bagging	0.8154	73.2855	42.4928
Multilayer Perceptron	0.8879	68.7779	39.8792
SMOreg	0.7492	96.0769	55.7078
IBk	0.555	118.0807	68.4662
KStar	0.7759	82.247	47.6889
Random Tree	0.3915	137.1558	79.5265
M5p	0.9118	67.1147	38.9148

3.6.2.1.4 Classifier Attribute Evaluation + Ranker for CocomoNasa

According to Table 3.12, Classifier Att. Eval. and Ranker have been applied under the select attributes menu of Weka. In this case, attributes are listed in order of importance. The last three elements in the order of importance are removed from the list. These three elements: rely, vexp, pcap. Therefore, the model was created with the remaining 14 attributes.

According to Table 3.12, M5P gave the best performance. The correlation coefficient is 0.9103, MAE is 62.9752 and RAE 36.5147 %. The worst performance is Random Tree. The Correlation Coefficient is 0.5973, MAE is 115.0607 and the RAE is 66.7151.

Compared to the original dataset, improvement was seen in 8 out of 9 algorithms. Only a slight decrease was seen in M5P. However, it has the best correlation coefficient performance. Random Tree showed an improvement of 20%. However, it has the worst result for this method and dataset.

Table 3.12: Classifier Att. Eval. + Ranker for CocomoNasa

CocomoNasa / Classifier Att.Eval + Ranker			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	0.8849	67.5768	39.1828
Random Forest	0.824	81.691	47.3666
Bagging	0.8132	74.1465	42.9921
Multilayer Perceptron	0.9054	62.768	36.3945
SMOreg	0.8045	74.7173	43.323
IBk	0.6164	101.91	59.0901
KStar	0.7568	78.9804	45.7949
Random Tree	0.5973	115.061	66.7151
M5p	0.9103	62.9752	36.5147

3.6.2.1.5 Correlation Attribute Evaluation + Ranker for CocomoNasa

According to Table 3.13, Correlation Att. Eval. and Ranker have been applied under the select attributes menu of Weka. In this case, attributes are listed in order of importance. The last three elements in the order of importance are removed from the list. These three elements: virt, vexp, acap. Therefore, the model was created with the remaining 14 attributes.

According to Table 3.13, Multilayer Perceptron gave the best performance. The correlation coefficient is 0.9147, MAE is 61.8765 and RAE 35.8776 %. The worst performance is IBK. The Correlation Coefficient is 0.6036, MAE is 108.3573 and the RAE is 62.8284.

Compared to the original dataset, improvement was seen in 8 out of 9 algorithms. Only a slight decrease was seen in M5P. Random Tree showed an improvement of 32%.

Table 3.13: Correlation Att. Eval. + Ranker for CocomoNasa

CocomoNasa / Correlation Att.Eval + Ranker			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	0.8778	89.4276	51.8524
Random Forest	0.7744	86.8572	50.3621
Bagging	0.8094	73.6507	42.7046
Multilayer Perceptron	0.9147	61.8765	35.8776
SMOreg	0.7912	85.6615	49.6687
IBk	0.6036	108.357	62.8284
KStar	0.7674	75.7444	43.9186
Random Tree	0.7113	93.3438	54.1231
M5p	0.8945	69.3779	40.2271

3.6.2.1.6 Relief Attribute Evaluation + Ranker for CocomoNasa

According to Table 3.14, Relief Att. Eval. and Ranker have been applied under the select attributes menu of Weka. In this case, attributes are listed in order of importance. The last three elements in the order of importance are removed from the list. These three elements: vexp, time, lexp. Therefore, the model was created with the remaining 14 attributes.

According to Table 3.14, M5P gave the best performance. The correlation coefficient is 0.9088, MAE is 63.6781 and RAE 36.9222 %. The worst performance is Random Tree. The Correlation Coefficient is 0.576, MAE is 124.3087 and the RAE is 72.0774 %.

Compared to the original dataset, improvement was seen in 7 out of 9 algorithms. Only a slight decrease was seen in M5P and IBK. Random Tree showed an improvement of 18%.

Table 3.14: Relief Att. Eval. + Ranker for CocomoNasa

CocomoNasa / Relief Att.Eval + Ranker			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	0.8558	86.2261	49.9961
Random Forest	0.7995	86.1667	49.9617
Bagging	0.8094	74.3053	43.0841
Multilayer Perceptron	0.8976	70.4519	40.8498
SMOreg	0.8334	73.8929	42.845
IBk	0.5768	117.491	68.1241
KStar	0.7697	79.2382	45.9444
Random Tree	0.576	124.309	72.0774
M5p	0.9088	63.6781	36.9222

3.6.2.2 CocomoNasa-2 Dataset with Attribute Selection

3.6.2.2.1 CFS + Random Search for CocomoNasa-2

According to Table 3.15, CfsSubsetEval and Random Search have been applied under the select attributes menu of Weka. In this case, 21 out of 24 attributes were selected. The selected attributes are Record number, project name, cat2, forg, year, mode, rely, data, cplx, time, stor, virt, turn, acap, aexp, pcap, vexp, modp, sced, equivphyskloc, act_effort.

According to Table 3.15, KStar gave the best performance. The correlation coefficient is 0.7433, MAE is 147.7978 and RAE is 57.205%. The worst performance is Linear Regression. Correlation Coefficient is -0.3101, MAE is 258.3653 and RAE is 100 %.

Compared to the original dataset, the Multilayer Perceptron Correlation Coefficient has increased by 15%. It is the model that shows the most improvement, if not the best result. KStar, which showed the best overall result, showed no improvement. Although it shows a small decrease, it can be interpreted that it almost maintains its success rate.

Table 3.15: CFS + Random Search for CocomoNasa-2

CocomoNasa-2 / CFS + Random Search			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	-0.3101	258.3653	100
Random Forest	0.6376	174.4391	67.5165
Bagging	0.4184	195.3487	75.6095
Multilayer Perceptron	0.6575	221.4978	85.7305
SMOreg	0.4176	307.0868	118.8576
IBk	0.5889	213.1759	82.5095
KStar	0.7433	147.7978	57.205
Random Tree	0.2493	202.0807	78.2151
M5p	0.734	134.5333	52.0709

3.6.2.2.2 CFS + PSO for CocomoNasa-2

According to Table 3.16, CfsSubsetEval and PSO have been applied under the select attributes menu of Weka. In this case, 22 out of 24 attributes were selected. The selected attributes are Record number, project name, cat2, forg, center, year, mode, rely, data, cplx, time, stor, virt, turn, acap, aexp, pcap, vexp, modp, sced, equivphyskloc, act_effort.

According to Table 3.16, KStar gave the best performance. The correlation coefficient is 0.7505, MAE is 146.7144 and RAE is 56.7856%. The worst performance is Linear Regression. Correlation Coefficient is -0.3101, MAE is 258.3653 and RAE is 100 %.

Compared to the original dataset, the Multilayer Perceptron Correlation Coefficient has increased by 16%. It is the model that shows the most improvement, if not the best result. Random Tree shows a huge decrease with 24%.

Table 3.16: CFS + PSO for CocomoNasa-2

CocomoNasa-2 / CFS + PSO			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	-0.3101	258.3653	100
Random Forest	0.6411	172.1151	66.617
Bagging	0.4184	195.3487	75.6095
Multilayer Perceptron	0.6603	240.6701	93.1511
SMOreg	0.3958	267.5702	103.5628
IBk	0.6464	212.0916	82.0898
KStar	0.7505	146.7144	56.7856
Random Tree	0.3947	225.2032	87.1647
M5p	0.734	134.5333	52.0709

3.6.2.2.3 CFS + Genetic Search for CocomoNasa-2

According to Table 3.17, CfsSubsetEval and GS have been applied under the select attributes menu of Weka. In this case, 22 out of 24 attributes were selected. The selected attributes are Record number, projectname, cat2, forg, center, year, mode, rely, data, cplx, time, stor, virt, turn, acap, aexp, pcap, vexp, modp, sced, equivphyskloc, act_effort.

According to Table 3.17, It has been observed that CFS + GS and CFS + PSO gave the same results.

Table 3.17: CFS + GS for CocomoNasa-2

CocomoNasa-2 / CFS + GS			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	-0.3101	258.3653	100
Random Forest	0.6411	172.1151	66.617
Bagging	0.4184	195.3487	75.6095
Multilayer Perceptron	0.6603	240.6701	93.1511
SMOreg	0.3958	267.5702	103.5628
IBk	0.6464	212.0916	82.0898
KStar	0.7505	146.7144	56.7856
Random Tree	0.3947	225.2032	87.1647
M5p	0.734	134.5333	52.0709

3.6.2.2.4 Classifier Attribute Evaluation + Ranker for CocomoNasa-2

According to Table 3.18, Classifier Att. Eval. and Ranker have been applied under the select attributes menu of Weka. In this case, attributes are listed in order of importance. The last three elements in the order of importance are removed from the list. These three elements: Record number, aexp, acap. Therefore, the model was created with the remaining 21 attributes.

According to Table 3.18, KStar gave the best performance. The correlation coefficient is 0.8046, MAE is 132.7143 and RAE 51.3669 %. The worst performance is Linear Regression. The Correlation Coefficient is -0.3101, MAE is 258.3653 and the RAE is 100. However, it didn't change according to original model. Except Linear Regression, worst performance has SMOreg. The Correlation Coefficient is 0.4107, MAE is 294.7582 and the RAE is 114.0858.

Compared to the original dataset, improvement was seen in 5 out of 9 algorithms. Only a slight decrease was seen in Random tree, SMOreg and Bagging. Multilayer Perceptron showed an improvement of 18%.

Table 3.18: Classifier Att. Eval. + Ranker for CocomoNasa-2

CocomoNasa-2 / Classifier Attribute Evaluation			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	-0.3101	258.365	100
Random Forest	0.7085	157.227	60.8545
Bagging	0.4122	195.52	75.6757
Multilayer Perceptron	0.6806	213.479	82.6269
SMOreg	0.4107	294.758	114.086
IBk	0.662	195.298	75.5897
KStar	0.8046	132.714	51.3669
Random Tree	0.5459	204.892	79.3032
M5p	0.7209	136.542	52.8485

3.6.2.2.5 Correlation Attribute Evaluation + Ranker for CocomoNasa-2

According to Table 3.19, Correlation Att. Eval. and Ranker have been applied under the select attributes menu of Weka. In this case, attributes are listed in order of importance. The last three elements in the order of importance are removed from the list. These three elements: Tool, lexp, year. Therefore, the model was created with the remaining 21 attributes.

According to Table 3.19, Random Forest gave the best performance. The correlation coefficient is 0.6994, MAE is 164.1088 and RAE 63.5181 %. The worst performance is Linear Regression. The Correlation Coefficient is -0.3101, MAE is 258.3653 and the RAE is 100. However, it didn't change according to original model. Except Linear Regression, worst performance has SMOreg. The Correlation Coefficient is 0.3777, MAE is 267.3135 and the RAE is 103.4634.

Compared to the original dataset, improvement was seen in 3 out of 9 algorithms. Multilayer Perceptron showed an improvement of 12%. A slight decrease was seen in MP5, KStar, IBK and SMOreg. However Random Tree showed a huge decrease of 25%.

Table 3.19: Correlation Att. Eval. + Ranker for CocomoNasa-2

CocomoNasa-2 / Correlation Attribute Evaluation			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	-0.3101	258.365	100
Random Forest	0.6994	164.109	63.5181
Bagging	0.4247	193.069	74.7272
Multilayer Perceptron	0.6215	245.462	95.0057
SMOreg	0.3777	267.314	103.463
IBk	0.6403	218.712	84.6522
KStar	0.6864	157.635	61.0124
Random Tree	0.3899	257.03	99.4831
M5p	0.6765	159.244	61.6352

3.6.2.2.6 Relief Attribute Evaluation + Ranker for CocomoNasa-2

According to Table 3.20, Relief Att. Eval. and Ranker have been applied under the select attributes menu of Weka. In this case, attributes are listed in order of importance. The last three elements in the order of importance are removed from the list. These three elements: Stor, year, record. Therefore, the model was created with the remaining 21 attributes.

According to Table 3.20, M5P gave the best performance. The correlation coefficient is 0.7508, MAE is 140.4593 and RAE 54.3646 %. The worst performance is Linear Regression. The Correlation Coefficient is -0.3101, MAE is 258.3653 and the RAE is 100. However, it didn't change according to original model. Except Linear Regression, worst performance has Bagging. The Correlation Coefficient is 0.4098, MAE is 194.7614 and the RAE is 75.3822%.

Compared to the original dataset, improvement was seen in 5 out of 9 algorithms. A slight decrease was seen in KStar, IBK and Bagging. Multilayer Perceptron showed a huge improvement with 18%.

Table 3.20: Relief Att. Eval. + Ranker for CocomoNasa-2

CocomoNasa-2 / Relief Attribute Evaluation			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	-0.3101	258.365	100
Random Forest	0.7311	150.459	58.235
Bagging	0.4098	194.761	75.3822
Multilayer Perceptron	0.6805	213.719	82.7196
SMOreg	0.4754	273.58	105.889
IBk	0.6564	199.429	77.1887
KStar	0.7308	144.878	56.075
Random Tree	0.7162	192.323	74.4385
M5p	0.7508	140.459	54.3646

3.6.2.3 Cocomo-81 Dataset with Attribute Selection

3.6.2.3.1 CFS + Random Search for Cocomo-81

According to Table 3.21, CfsSubsetEval and Random Search have been applied under the select attributes menu of Weka. In this case, 11 out of 17 attributes were selected. The selected attributes are relying, data, time, stor, turn, acap, pcap, vexp, modp, loc, actual.

According to Table 3.21, KStar gave the best performance. The correlation coefficient is 0.9097, MAE is 183.1173 and RAE is 50.4529 %. The worst performance is Random Tree. Correlation Coefficient is 0.4613, MAE is 307.7039 and RAE is 84.7792 %.

Compared to the original dataset, the IBK Correlation Coefficient has huge increased from 0.0768 to 0.5258. It is the model that shows the most improvement, if not the best result. Random Tree shows a huge decrease with approximately 18%.

Table 3.21: CFS + Random Search for Cocomo-81

CocomoNasa-81 / CFS + Random Search			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	0.6314	329.8316	90.8759
Random Forest	0.8048	193.219	53.2361
Bagging	0.4769	264.1178	72.7703
Multilayer Perceptron	0.5413	352.0298	96.992
SMOreg	0.661	191.3021	52.708
IBk	0.5258	290.6483	80.08
KStar	0.9097	183.1173	50.4529
Random Tree	0.4613	307.7039	84.7792
M5p	0.6674	218.1156	60.0957

3.6.2.3.2 CFS + PSO for Cocomo-81

According to Table 3.22, CfsSubsetEval and PSO have been applied under the select attributes menu of Weka. In this case, 12 out of 17 attributes were selected. The selected attributes are relying, data, time, stor, turn, acap, pcap, vexp, lexp, modp, loc, actual.

According to Table 3.22, KStar gave the best performance. The correlation coefficient is 0.8597, MAE is 201.859 and RAE is 55.6166 %. The worst performance is Random Tree. Correlation Coefficient is 0.4117, MAE is 304.489 and RAE is 83.8936 %.

Compared to the original dataset, improvement was seen in 5 out of 9 algorithms. A decrease was seen in Random Tree by 23%. IBk showed a huge improvement with 44%.

Table 3.22: CFS + PSO for Cocomo-81

Cocomo-81 / CFS + PSO			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	0.6328	330.793	91.1408
Random Forest	0.7422	202.496	55.792
Bagging	0.4773	264.13	72.7738
Multilayer Perceptron	0.6118	307.926	84.8405
SMOreg	0.6726	186.926	51.5023
IBk	0.5206	303.708	83.6783
KStar	0.8597	201.859	55.6166
Random Tree	0.4117	304.489	83.8936
M5p	0.6773	208.567	57.4648

3.6.2.3.3 CFS + Genetic Search for Cocomo-81

According to Table 3.23, CfsSubsetEval and PSO have been applied under the select attributes menu of Weka. In this case, 12 out of 17 attributes were selected. The selected attributes are relying, data, time, stor, turn, acap, pcap, vexp, lexp, modp, loc, actual.

According to Table 3.23, It has been observed that CFS + GS and CFS + PSO gave the same results.

Table 3.23: CFS + GS for Cocomo-81

Cocomo-81 / CFS + GS			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	0.6328	330.793	91.1408
Random Forest	0.7422	202.496	55.792
Bagging	0.4773	264.13	72.7738
Multilayer Perceptron	0.6118	307.926	84.8405
SMOreg	0.6726	186.926	51.5023
IBk	0.5206	303.708	83.6783
KStar	0.8597	201.859	55.6166
Random Tree	0.4117	304.489	83.8936
M5p	0.6773	208.567	57.4648

3.6.2.3.4 Classifier Attribute Evaluation + Ranker for Cocomo-81

According to Table 3.24, Classifier Att. Eval. and Ranker have been applied under the select attributes menu of Weka. In this case, attributes are listed in order of importance. The last three elements in the order of importance are removed from the

list. These three elements: Pcap, vexp, rely. Therefore, the model was created with the remaining 14 attributes.

According to Table 3.24, Random Forest gave the best performance. The correlation coefficient is 0.764, MAE is 220.5193 and RAE 60.758 %. The worst performance has Bagging. The Correlation Coefficient is 0.4225, MAE is 273.9946 and the RAE is 75.4916.

Compared to the original dataset, improvement was seen in 5 out of 9 algorithms. Only a slight decrease was seen in M5P, Random Tree, SMOreg and Bagging. IBk showed a huge improvement of 44%.

Table 3.24: Classifier Att. Eval. + Ranker for Cocomo-81

Cocomo-81 / Classifier Attribute Evaluation			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	0.6184	334.931	92.281
Random Forest	0.764	220.519	60.758
Bagging	0.4225	273.995	75.4916
Multilayer Perceptron	0.7224	267.257	73.6353
SMOreg	0.6556	193.899	53.4235
IBk	0.5598	265.961	73.278
KStar	0.7409	202.687	55.8449
Random Tree	0.5528	267.414	73.6785
M5p	0.6697	221.118	60.9229

3.6.2.3.5 Correlation Attribute Evaluation + Ranker for Cocomo-81

According to Table 3.25, Correlation Att. Eval. and Ranker have been applied under the select attributes menu of Weka. In this case, attributes are listed in order of importance. The last three elements in the order of importance are removed from the list. These three elements: Tool, aexp, acap. Therefore, the model was created with the remaining 14 attributes.

According to Table 3.25, Random Forest gave the best performance. The correlation coefficient is 0.8379, MAE is 197.6953 and RAE 54.4694 %. The worst performance has Bagging. The Correlation Coefficient is 0.4098, MAE is 276.8935 and the RAE is 76.2903.

Compared to the original dataset, improvement was seen in 6 out of 9 algorithms. Only a slight decrease was seen in M5P, Random Tree, and Bagging. KStar showed a huge improvement of 23%.

Table 3.25: Correlation Att. Eval. + Ranker for Cocomo-81

Cocomo-81 / Correlation Attribute Evaluation			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	0.6388	333.727	91.9491
Random Forest	0.8379	197.695	54.4694
Bagging	0.4098	276.894	76.2903
Multilayer Perceptron	0.6865	279.547	77.0213
SMOreg	0.6791	194.345	53.5462
IBk	0.5508	271.7	74.8594
KStar	0.7967	201.259	55.4513
Random Tree	0.598	259.095	71.3863
M5p	0.679	206.685	56.9462

3.6.2.3.6 Relief Attribute Evaluation + Ranker for Cocomo-81

According to Table 3.26 Relief Att. Eval. and Ranker have been applied under the select attributes menu of Weka. In this case, attributes are listed in order of importance. The last three elements in the order of importance are removed from the list. These three elements: Cplx, sced, pcap. Therefore, the model was created with the remaining 14 attributes.

According to Table 3.26, Random Forest gave the best performance. The correlation coefficient is 0.7811, MAE is 202.192 and RAE 55.7084 %. The worst performance has Random Tree. The Correlation Coefficient is 0.2225, MAE is 349.914 and the RAE is 96.4091 %.

Compared to the original dataset, improvement was seen in 2 out of 9 algorithms that are Linear Algorithm and Random Forest.

Table 3.26: Relief Att. Eval. + Ranker for Cocomo-81

Cocomo-81 / Relief Attribute Evaluation			
Algorithms	**Correlation Coefficient**	**MAE**	**RAE (%)**
Linear Regression	0.6216	329.365	90.7472
Random Forest	0.7811	202.192	55.7084
Bagging	0.4339	271.4	74.7768
Multilayer Perceptron	0.6409	278.156	76.6381
SMOreg	0.6591	184.564	50.8513
IBk	0.0665	326.587	89.982
KStar	0.4731	220.061	60.6317
Random Tree	0.2225	349.914	96.4091
M5p	0.6708	224.265	61.7899

CHAPTER IV
RESULT AND DISCUSSION

In this section, a detailed analysis of the new models created using machine learning algorithms and their results has been made. The analyzes obtained as a result of the examination of the researched studies are presented in detail in the Tables.

Existing studies were compared according to the software test effort estimation method, the data sets they used, whether they made feature selection and evaluation criteria. In these analyzes and comparisons, Correlation Coefficient, MAE and RAE were used as performance evaluation criteria.

The main purpose of this section is to help researchers learn which machine learning method provides promising accuracy estimation in software test effort estimation.

4.1 RESULT AND DISCUSSION FOR COCOMONASA DATASET

The results of the models created with the CocomoNasa data set are given in the Table 4.1. It gave improved results with different algorithms in each applied method. Classification Att.Eval. and Correlation Att.Eval. methods showed improvement in 8 of 9 algorithms. They showed an improvement rate of 89 %. Relief Att.Eval. showed improvement in 7 of 9 algorithms.

The algorithm that showed the most improvement was the Random tree as 44% with CFS + Random Search. However, Multilayer Perceptron gave the best results. Multilayer Perceptron gives the best results with CFS + Random Search. Mp5 gave the closest result to the best result with the original model. No improvement was observed in the models created with M5P.

When all methods are compared, the analysis of the highest values that each algorithm could reach was made. These values were captured 3 times with the CFS +

Random Search and Classifier Att.Eval. + Ranker. With a rate of 33 percent, they had the best rate among other methods.

Table 4.1: Result for CocomoNasa Dataset

ALGORITHMS		Linear Regression	Random Forest	Bagging	Multilayer Perceptron	SMOreg	Ibk	Kstar	Random Tree	M5p
Original Model	Correlation Coefficient	0.7994	0.7281	0.8083	0.8931	0.7178	0.5768	0.6772	0.3915	0.922
	MAE	98.8186	93.4642	74.0699	71.781	99.5863	118.1707	88.1806	137.1558	60.3936
	RAE (%)	57.2976	54.193	42.9476	41.6205	57.7427	68.5184	51.1294	79.5265	35.0178
CFS + RandomSearch	Correlation Coefficient	0.7782	0.7281	0.8111	0.9245	0.7691	0.5511	0.7818	0.8331	0.903
	MAE	105.5529	93.4642	74.6154	57.5215	90.0964	118.1307	75.6172	74.3994	69.6416
	RAE (%)	61.2023	54.193	74.6154	33.3525	52.2402	68.4952	43.8448	43.1387	40.38
CFS + PSO	Correlation Coefficient	0.7396	0.8115	0.8103	0.8956	0.6779	0.5504	0.7592	0.7531	0.9021
	MAE	135.8567	80.0985	74.934	64.8591	113.7726	120.4973	79.0905	87.2165	70.3909
	RAE (%)	78.7732	46.4432	43.4487	37.607	65.9683	69.8675	45.8587	50.5704	40.8145
CFS + GA	Correlation Coefficient	0.7648	0.8193	0.8154	0.8879	0.7492	0.555	0.7759	0.3915	0.9118
	MAE	111.1063	81.1336	73.2855	68.7779	96.0769	118.0807	82.247	137.1558	67.1147
	RAE (%)	64.4223	47.0434	42.4928	39.8792	55.7078	68.4662	47.6889	79.5265	38.9148
ClassifierAttEval + Ranker	Correlation Coefficient	0.8849	0.824	0.8132	0.9054	0.8045	0.6164	0.7568	0.5973	0.9103
	MAE	67.5768	81.691	74.1465	62.768	74.7173	101.91	78.9804	115.0607	62.9752
	RAE (%)	39.1828	47.3666	42.9921	36.3945	43.323	59.0901	45.7949	66.7151	36.5147
Corr. Att.Eval + Ranker	Correlation Coefficient	0.8778	0.7744	0.8094	0.9147	0.7912	0.6036	0.7674	0.7113	0.8945
	MAE	89.4276	86.8572	73.6507	61.8765	85.6615	108.3573	75.7444	93.3438	69.3779
	RAE (%)	51.8524	50.3621	42.7046	35.8776	49.6687	62.8284	43.9186	54.1231	40.2271
Relief. Att.Eval + Ranker	Correlation Coefficient	0.8558	0.7995	0.8094	0.8976	0.8334	0.5768	0.7697	0.576	0.9088
	MAE	86.2261	86.1667	74.3053	70.4519	73.8929	117.4907	79.2382	124.3087	63.6781
	RAE (%)	49.9961	49.9617	43.0841	40.8498	42.845	68.1241	45.9444	72.0774	36.9222

4.2 RESULT AND DISCUSSION FOR COCOMONASA-2 DATASET

The results of the models created with the CocomoNasa-2 data set are given in the Table 4.2. It gave improved results with different algorithms in each applied method. Classification Att.Eval. and Relief Att.Eval. methods showed improvement in 5 of 9 algorithms. They showed an improvement rate of 55 %.

The algorithm that showed the most improvement was the Multilayer Perceptron with 18% with Classifier Att.Eval. + Ranker. However, KStar gave the best results. KStar gives the best results with Classifier Att.Eval. + Ranker. Mp5 gave the closest result to the best result with the 0.7508 of correlation coefficient.

When all methods are compared, the analysis of the highest values that each algorithm could reach was made. These values were captured 4 times with the Relief Att.Eval. + Ranker. With a rate of 44 percent except linear regression. It had the best rate among other methods. It was not included in this analysis as no change was observed in linear regression.

Table 4.2: Result for CocomoNasa-2 Dataset

ALGORITHMS		Linear Regression	Random Forest	Bagging	Multilayer Perceptron	SMOreg	Ibk	Kstar	Random Tree	M5p
Original Model	Correlation Coefficient	-0.3101	0.667	0.4241	0.5058	0.4176	0.6581	0.7437	0.637	0.7092
	MAE	258.3653	164.1404	195.6914	303.8009	307.0868	198.9049	146.6103	201.0081	147.0593
	RAE (%)	100	339.9489	75.7421	117.5858	118.8576	76.9859	56.7453	77.8	56.9191
CFS + RandomSearch	Correlation Coefficient	-0.3101	0.6376	0.4184	0.6575	0.4176	0.5889	0.7433	0.2493	0.734
	MAE	258.3653	174.4391	195.3487	221.4978	307.0868	213.1759	147.7978	202.0807	134.5333
	RAE (%)	100	67.5165	75.6095	85.7305	118.8576	82.5095	57.205	78.2151	52.0709
CFS + PSO	Correlation Coefficient	-0.3101	0.6411	0.4184	0.6603	0.3958	0.6464	0.7505	0.3947	0.734
	MAE	258.3653	172.1151	195.3487	240.6701	267.5702	212.0916	146.7144	225.2032	134.5333
	RAE (%)	100	66.617	75.6095	93.1511	103.5628	82.0898	56.7856	87.1647	52.0709
CFS + GA	Correlation Coefficient	-0.3101	0.6411	0.4184	0.6603	0.3958	0.6464	0.7505	0.3947	0.734
	MAE	258.3653	172.1151	195.3487	240.6701	267.5702	212.0916	146.7144	225.2032	134.5333
	RAE (%)	100	66.617	75.6095	93.1511	103.5628	82.0898	56.7856	87.1647	52.0709
ClassifierAttEval + Ranker	Correlation Coefficient	-0.3101	0.7085	0.4122	0.6806	0.4107	0.662	0.8046	0.5459	0.7209
	MAE	258.3653	157.227	195.5197	213.4793	294.7582	195.2976	132.7143	204.8919	136.5422
	RAE (%)	100	60.8545	75.6757	82.6269	114.0858	75.5897	51.3669	79.3032	52.8485
Corr. Att.Eval + Ranker	Correlation Coefficient	-0.3101	0.6994	0.4247	0.6215	0.3777	0.6403	0.6864	0.3899	0.6765
	MAE	258.3653	164.1088	193.0693	245.4617	267.3135	218.7118	157.6348	257.0298	159.244
	RAE (%)	100	63.5181	74.7272	95.0057	103.4634	84.6522	61.0124	99.4831	61.6352
Relief. Att.Eval + Ranker	Correlation Coefficient	-0.3101	0.7311	0.4098	0.6805	0.4754	0.6564	0.7308	0.7162	0.7508
	MAE	258.3653	150.459	194.7614	213.7188	273.58	199.4288	144.8783	192.3231	140.4593
	RAE (%)	100	58.235	75.3822	82.7196	105.8888	77.1887	56.075	74.4385	54.3646

4.3 RESULT AND DISCUSSION FOR COCOMO-81 DATASET

The results of the models created with the Cocomo-81 data set are given in the Table 4.3. It gave improved results with different algorithms in each applied method. Correlation Att.Eval + Ranker. methods showed improvement in 6 of 9 algorithms. They showed an improvement rate of 66 %.

The algorithm that showed the most improvement was the IBK as 48% with Classifier Att.Eval. + Ranker. However, KStar gave the best results. KStar gives the best results with Cfs + Random Search.

When all methods are compared, the analysis of the highest values that each algorithm could reach was made. These values were captured 3 times with the Correlation Att.Eval. + Ranker with a rate of 33 percent.

Table 4.3: Result for Cocomo-81 Dataset

ALGORITHMS		Linear Regression	Random Forest	Bagging	Multilayer Perceptron	SMOreg	Ibk	Kstar	Random Tree	M5p
Original Model	Correlation Coefficient	0.6102	0.7529	0.4622	0.6739	0.6625	0.0768	0.5621	0.6439	0.6843
	MAE	349.7908	215.6668	268.7705	264.9429	191.8657	313.021	210.9438	226.4391	206.9436
	RAE (%)	96.3751	59.421	74.0522	72.9976	52.8632	86.2442	58.1197	62.389	57.0175
CFS + RandomSearch	Correlation Coefficient	0.6314	0.8048	0.4769	0.5413	0.661	0.5258	0.9097	0.4613	0.6674
	MAE	329.8316	193.219	264.1178	352.0298	191.3021	290.6483	183.1173	307.7039	218.1156
	RAE (%)	90.8759	53.2361	72.7703	96.992	52.708	80.08	50.4529	84.7792	60.0957
CFS + PSO	Correlation Coefficient	0.6328	0.7422	0.4773	0.6118	0.6726	0.5206	0.8597	0.4117	0.6773
	MAE	330.7929	202.4956	264.1304	307.9262	186.926	303.7079	201.859	304.4894	208.5668
	RAE (%)	91.1408	55.792	72.7738	84.8405	51.5023	83.6783	55.6166	83.8936	57.4648
CFS + GA	Correlation Coefficient	0.6328	0.7422	0.4773	0.6118	0.6726	0.5206	0.8597	0.4117	0.6773
	MAE	330.7929	202.4956	264.1304	307.9262	186.926	303.7079	201.859	304.4894	208.5668
	RAE (%)	91.1408	55.792	72.7738	84.8405	51.5023	83.6783	55.6166	83.8936	57.4648
ClassifierAttEval + Ranker	Correlation Coefficient	0.6184	0.764	0.4225	0.7224	0.6556	0.5598	0.7409	0.5528	0.6697
	MAE	334.9313	220.5193	273.9946	267.2573	193.899	265.9606	202.6874	267.414	221.1179
	RAE (%)	92.281	60.758	75.4916	73.6353	53.4235	73.278	55.8449	73.6785	60.9229
Corr. Att.Eval + Ranker	Correlation Coefficient	0.6388	0.8379	0.4098	0.6865	0.6791	0.5508	0.7967	0.598	0.679
	MAE	333.7268	197.6953	276.8935	279.5468	194.3445	271.7003	201.2589	259.0946	206.6847
	RAE (%)	91.9491	54.4694	76.2903	77.0213	53.5462	74.8594	55.4513	71.3863	56.9462
Relief. Att.Eval + Ranker	Correlation Coefficient	0.6216	0.7811	0.4339	0.6409	0.6591	0.0665	0.4731	0.2225	0.6708
	MAE	329.3646	202.1921	271.4004	278.1559	184.5635	326.5873	220.061	349.9143	224.2648
	RAE (%)	90.7472	55.7084	74.7768	76.6381	50.8513	89.982	60.6317	96.4091	61.7899

4.4 CONCLUSION

The software testing process is one of the most important stages of software development projects. The fact that the software is intangible and contains many unknowns both complicates the software testing process and takes time. Incorrect software testing effort and time estimations play a role in the failure of software projects.

Therefore, many software test effort estimation methods have been developed to improve the accuracy of software test effort estimation. One of these estimation methods is Artificial Intelligence methods. In this thesis, six different models have been developed using ML algorithms for the estimation of test effort of software test projects. Each developed model was applied on the datasets. The attributes of the data sets used in software test effort estimation significantly affect the estimation accuracy.

It has been determined that ignoring the feature selection in the estimation process of the software test effort negatively affects the estimation result. Feature selection is one of the commonly used preprocessing techniques of the machine learning community for the removal of irrelevant, noisy, and redundant data while increasing the learning accuracy and improving the quality of the classification results. [29]. In the thesis study, CfsSubsetEval, Classifier Att.Eval as the evaluator method for software test effort estimation., Correlation Att.Eval and Relief Att.Eval are used.

For the search method, Genetic Search, PSO Search, Random Search and Ranker search algorithms were used respectively.

In this way, it is seen how feature selection improves test effort estimation accuracy.

The first developed model was implemented in two different ways using the ML algorithms in the WEKA program on the COCOMO-81, COCOMONASA and COCOMONASA2 datasets.

In the first part; In the simulations where the default settings of the algorithms in the WEKA program were preferred, the Random Forest algorithm gave the best estimate in the COCOMO-81 dataset, and the IBK algorithm gave the worst estimate. In the COCOMONASA dataset, the best algorithm in estimation is M5P, the worst estimated one is Random Tree. In the COCOMONASA2 dataset, the KStar algorithm is the best estimate and the Linear Regression algorithm is the worst estimated.

In the second part, new models were created by making feature selections on the data sets. In the newly created models, the best estimated KStar algorithm in the COCOMO-81 data set. The Multilayer Perceptron algorithm gave the best estimate in the COCOMONASA dataset, and the KStar algorithm gave the best estimate in the COCOMONASA2 dataset.

Algorithms that give the best results for each data set, hybrid methods used and Correlation coefficient, Mae, Rae ratios are listed. When the datasets are compared, KStar has been the algorithm that has achieved the best result 2 times.

Table 4.4: Best Results for All Dataset

BEST RESULT					
DataSet	**Algorithm**	**Method**	**Correlation Coefficient**	**MAE**	**RAE (%)**
CocomaNasa	MultiLayer Perceptron	Cfs+ Random Search	0.9245	57.5215	33.3525
CocomaNasa-2	Kstar	ClassifierAttEval+Ranker	0.8046	132.714	51.3669
Cocomo-81	Kstar	Cfs+ Random Search	0.9097	183.117	50.4529

It was observed in Table 4.4 that the random search method was used for 2 datasets to achieve the best results for CocomoNasa and Cocomo-81 datasets. Even if the best result is obtained with Classifier Att.Eval. + Ranker in CocomoNasa-2 dataset, given Table 4.5 shows that the Correlation Coefficient value is close to the best result when CFS + Random Search is applied. The Correlation Coefficient value for CFS + Random search is 0.7433.

In addition, with the feature selections applied in Table 4.5, it is seen that successful results are obtained with reduced features compared to the results obtained with the original feature.

Table 4.5: Model Performance Results with Feature Selection

Dataset	Orginal Feature Set	Model	Correlation Coefficient Before Feature Selection	FeatureSelection	Selected Feature Set	Correlation Coefficien
CocomoNasa	**17**	**Multilayer Perceptron**	**0.8931**	**CFS+ RandomSearch**	**10**	**0.9245**
CocomoNasa	17	M5p	0.922	CFS+ PSO	12	0.9021
CocomoNasa	17	M5p	0.922	CFS+ GA	10	0.9118
CocomoNasa	17	M5p	0.922	ClassifierAttEval+ Ranker	14	0.9103
CocomoNasa	17	Multilayer Perceptron	0.8931	Corr. Att.Evaluation + Ranker	14	0.9147
CocomoNasa	17	M5p	0.922	Relief. Att.Evaluation + Ranker	14	0.9088
CocomoNasa-2	24	K Star	0.7437	Original Feature Set	24	0.7437
CocomoNasa-2	24	K Star	0.7437	CFS+ RandomSearch	21	0.7433
CocomoNasa-2	24	K Star	0.7437	CFS+ PSO	22	0.7505
CocomoNasa-2	24	K Star	0.7437	CFS+ GA	22	0.7505
CocomoNasa-2	**24**	**M5p**	**0.7092**	**ClassifierAttEval+ Ranker**	**21**	**0.8046**
CocomoNasa-2	24	Random Forest	0.667	Corr. Att.Evaluation + Ranker	21	0.6994
CocomoNasa-2	24	M5p	0.7092	Relief. Att.Evaluation + Ranker	21	0.7508
CocomoNasa-81	**17**	**K Star**	**0.5621**	**CFS+ RandomSearch**	**11**	**0.9097**
CocomoNasa-81	17	K Star	0.5621	CFS+ PSO	12	0.8597
CocomoNasa-81	17	K Star	0.5621	CFS+ GA	12	0.8597
CocomoNasa-81	17	Random Forest	0.7529	ClassifierAttEval+ Ranker	14	0.764
CocomoNasa-81	17	Random Forest	0.7529	Corr. Att.Evaluation + Ranker	14	0.8379
CocomoNasa-81	17	Random Forest	0.7529	Relief. Att.Evaluation + Ranker	14	0.7811

For the CocomoNasa dataset, best algorithm is Multilayer Perceptron. The comparison results of the best results are given Figure 4.1. CFS and Random Search methods were applied.

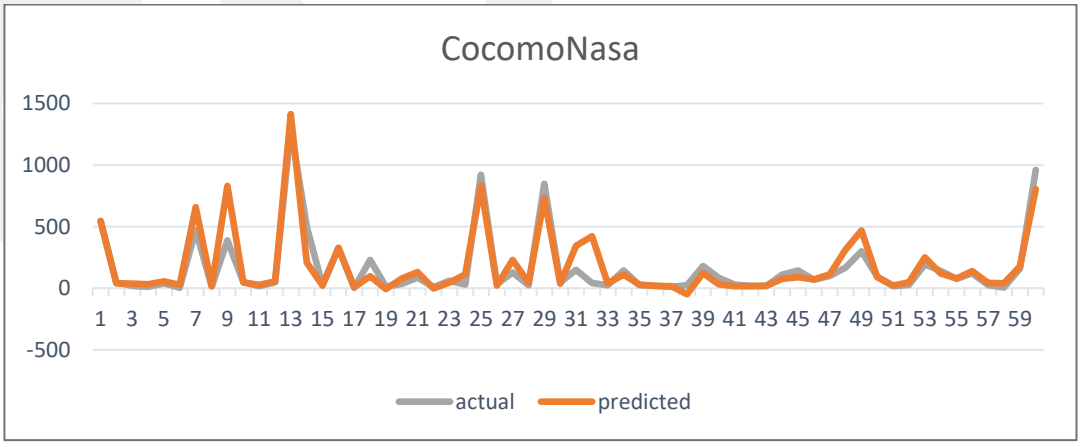

Figure 4.1: Comparison Graph for Best Result of CocomoNasa Dataset

For the CocomoNasa-2 dataset, best algorithm is KStar. The comparison results of the best results are given Figure 4.2. Classifier Attribute Evaluator and Ranker methods were applied.

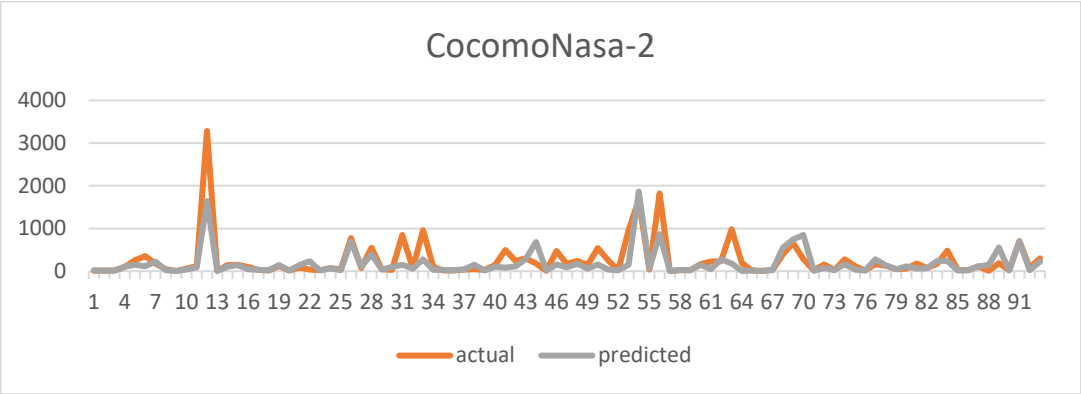

Figure 4.2: Comparison Graph for Best Result of CocomoNasa-2 Dataset

For the CocomoNasa-81 dataset, best algorithm is KStar. The comparison results of the best results are given Figure 4.3. CFS and Random Search methods were applied.

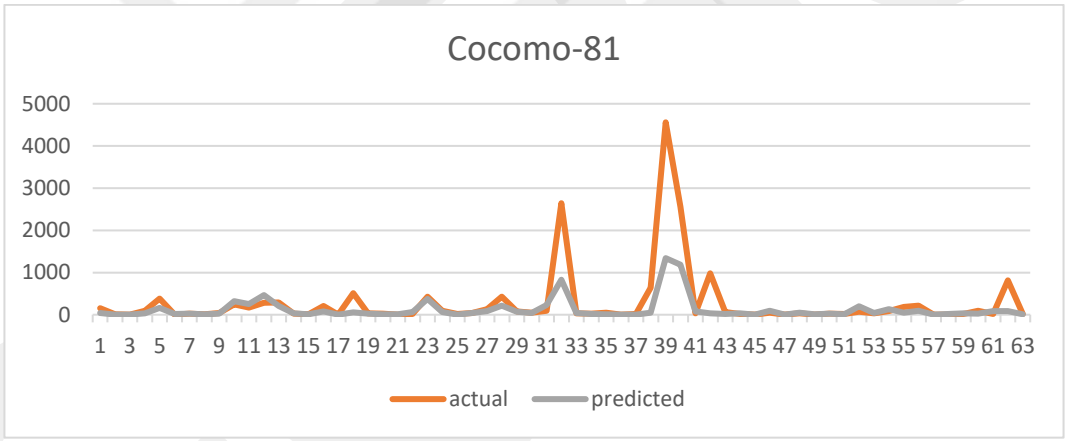

Figure 4.3 Comparison Graph for Best Result of Cocomo81 dataset

In addition, the models showing the most improvement are listed in Table 4.6. When all datasets are examined, improvements have been found in all models and algorithms in general. Although there are cases where the correlation coefficient value deteriorates in some cases, this was not so much.

In CocomoNasa CFS + Random Search Method was the hybrid method that showed the most improvement. In the CocomoNasa dataset, the Random Tree algorithm showed an improvement of 44%. In the Cocomo-81 dataset, the IBK algorithm showed 48% improvement with Classifier Att.Eval. + Ranker. On the other hand, the CocomaNasa-2 dataset showed an improvement of 18% with the Multilayer Perceptron algorithm and the Classifier Att.Eval. + Ranker hybrid method.

Table 4.6: The Improvement Result for All Datasets

BEST IMPROVEMENT RATE					
Dataset	**Algorithm**	**Method**	**Original Model for Correlation Coefficient**	**Highest Improvement for Correlation Coefficient**	**Improvement Rate**
CocomoNasa	RandomTree	CFS + Random Search	0.3915	0.8331	0.44
CocomoNasa-2	MultiLayerPerceptron	Classifier Att. Eval. + Ranker	0.5058	0.6806	0.18
Cocomo-81	IBk	Classifier Att. Eval. + Ranker	0.0768	0.5598	0.48

In this study, the performance of WEKA program and ML algorithms in software cost estimation using COCOMO81, COCOMONASA, COCOMONASA2 datasets in PROMISE data repository were examined.

When the estimation results are examined, it has been determined that the error rates and correlation coefficients of the algorithms vary according to the data sets they are applied to. It has been observed that an algorithm does not always produce the best results, while some algorithms produce very good results in some data sets, but may give bad results with different parameters and different data sets.

In addition, it has been noticed that the features in the data sets greatly affect the estimation result of the feature selection method used to determine the features. When the performance values are examined, it has been seen that the feature selection on the data sets used for software cost estimation provides improvement results in ML algorithms in general.

It has been observed that the model created with the Classifier Att. Eval. + Ranker hybrid method on the datasets discussed in this thesis shows more improvement than other models. In future studies, it can be aimed to increase the estimation accuracy of the model for more projects by multiplying the data sets.

REFERENCES

[1] LIU Qin and MINTRAM Robert (2005), "Preliminary Data Analysis Methods in Software Estimation", *Software Quality Journal,* Volume 13, pp. 91-115.

[2] AHMAD Nurain Sabrina, KHAN Mohammad G. M. and RAFI Loriza S. (2009), "Study of Testing-Effort Dependent Inflection S-Shaped Software Reliability Growth Models with Imperfect Debugging", *International Journal of Quality & Reliability Management,* Volume 27, No 1, pp. 89-100.

[3] CIBIR Esra (2021), *Savunma Sistemlerinde Test Efor Tahminlenmesi* (Master's Thesis), Başkent University Graduate School of Natural and Applied, Ankara.

[4] KAFLE Lava (2014), "An Empirical Study on Software Test Effort Estimation", *International Journal of Soft Computing and Artificial Intelligence,* Volume 2, No 2, pp. 48082-48087.

[5] HOURANI Hussam, HAMMAD Ahmad and LAFI Mohammad (2019), "The impact of artificial intelligence on software testing", *Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT),* pp. 565-570, Amman, Jordan.

[6] SHARMA Aditi and RANJAN Ravi (2017), "Software Effort Estimation using Neuro Fuzzy Inference System: Past and Present", *International Journal on Recent and Innovation Trends in Computing and Communication,* Volume 5, No 8, pp. 78-83.

[7] COTRONEO Domenico, PIETRANTUONO Roberto and RUSSO Stefano (2013), "A learning-based method for combining testing techniques", *35th International Conference on Software Engineering (ICSE),* pp. 142-151, San Francisco, CA, USA.

[8] BRIAND Lionel C., LABICHE Yvan and BAWAR Zaheer (2008), "Using Machine Learning to Refine Black-Box Test Specifications and Test Suites", *2008 The Eighth International Conference on Quality Software*, pp. 135-144, Oxford, UK.

[9] DURELLI Vinicius H. S., DURELLI Rafael S. and BORGES Simone S. (2019), "Machine learning applied to software testing a systematic mapping study", *IEEE Transactions on Reliability*, Volume 68, No 3, pp. 1189–1212.

[10] BOEHM Barry W., ABTS Chris, CHULANI Sunita, CLARK Bradford K., HOROWITZ Ellis, MADACHY Ray, REIFER Donald J. and STEEECE Bert (2000), *Software cost estimation with COCOMO II*, Prentice Hall, California.

[11] SINGH Sanjay Kumar and SINGH Amarjeet (2019), *Software Testing*, Vandana Publications, Lucknow, India .

[12] NAWAZ Ahsan and MALIK Kashif Masood (2008), *Software Testing Process In Agile Development* (Master's Thesis*),* Blekinge Institute of Technology ,Department of Computer Science School of Engineering, Ronneby, Sweeden.

[13] IEEE Standards Board and Standards Coordinating committee of the Computer Society of the IEEE (1990), *IEEE Standard Glossary of Software Engineering Terminology*, IEEE The Institute of Electrical and Electronics Engineer, New York.

[14] MYERS Glenford (2004), *The Art of Software Testing,* Second Edition, Wiley, New Jersey.

[15] CRAIG Rick D. and JASKIEL Stefan P. (2002), *Systematic Software Testing, Artech House Publishers*, Artech House, Boston.

[16] YAGCI Nurhan (2013), *Yazilim Kalite Metrikleri İle Test Eforu Arasindaki İlişkinin Belirlenip Tarihsel Verinin Olusturulmasi* (Master's Thesis), Sakarya University Graduate School of Natural and Applied Sciences, Sakarya.

[17] AKAGUNDUZ Serkan, KURNAZ Salih and SARI Mustafa (2013), "Factors That Make the Success of The Project in Software Project Management", *XV. Akademik Bilişim Konferans Bildirileri*, pp. 983-986 Akdeniz University, Antalya.

[18] BEIZER Boris (1995), *Black Box Testing: Techniques for Functional Testing of Software and Systems*, Second Edition, Wiley, New York.

[19] AFZAL Wasif (2007), *Metrics in Software Test Planning and Test* (Master's Thesis), School of Engineering Blekinge Institute of Technology, Karlskrona.

[20] KITCHENHAM Barbara A. and MENDES Emilia (2004), "Software Test Effort Estimation: A Review of Analytical Models", *Software Testing, Verification and Reliability*, Volume 14, No 2, pp. 105-154.

[21] JORDAN Micheal I. and MITCHELL Tom (2015), "Machine Learning: Trends, Perspectives, and Prospects", *Science*, Volume 349, No 6245, pp. 255-60.

[22] SINGH Amanpreet, THAKUR Narina and SHARMA Aakanksha (2016), "A review of supervised machine learning algorithms", *3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 1310-1315, New Delhi, India.

[23] ZHAO Yongheng and ZHANG Yanxia (2008), "Comparison of Decision Tree Methods for Finding Active Objects", *Advances in Space Research*, Volume 41, No 12, pp. 1955–1959.

[24] MOHAMMED Ahmed, RAFIQ Serwan, SIHAG Parveen, KURDA Rawaz, MAHMOOD Wael, GHAFOR Kawan and SARWAR Warzer (2020), "ANN, M5P-Tree and Nonlinear Regression Approaches with Statistical Evaluations to Predict the Compressive Strength of Cement-Based Mortar Modified with Fly Ash", *Journal of Materials Research and Technology*, Volume 9, No 6, pp. 12416-12427.

[25] HALLL Mark A. (1999), *Correlation-Based Feature Selection for Machine Learning* (Doctoral Dissertation), University of Waikato, Hamilton.

[26] DOKEROGLU Tansel, SEVINC Ender, KUCUKYILMAZ Tayfun and COSAR Ahmet (2019), "A Survey on New Generation Metaheuristic Algorithms", *Computers & Industrial Engineering*, Volume 137, pp. 106040, DOI: 10.1016/j.cie.2019.106040. DoA. 08.07.2023.

[27] CHAHAR Vikas and BHATIA Pradeep Kumar (2022), "Performance Analysis of Software Test Effort Estimation using Genetic Algorithm and Neural Network," *(IJACSA) International Journal of Advanced Computer Science and Applications*, Volume 13, No 10, pp. 101-107.

[28] SRIVASTAVA Praveen Ranjan (2016), “Estimation of software testing effort using fuzzy multiple linear regression”, *International Journal of Software Engineering Technology and Application (IJSETA)*, Volume 14, pp. 145-154.

[29] DOKEROGLU Tansel, DENIZ Ayca and KIZILOZ Hakan Ezgi (2022), "A comprehensive survey on recent metaheuristics for feature selection", *Neurocomputing*, Volume 494, No 14, pp. 269-296.