-ROYAUME DU MAROC-
-HAUT COMMISSARIAT AU PLAN-
INSTITUT NATIONAL DE STATISTIQUE ET D'ECONOMIE APPLIQUEE

Projet de Fin d'Etudes

Building Machine Learning Model for Software Effort Estimation

Préparé par : **LYOUSFI Youssef**

Sous la direction de : **Mme. EL BAJTA Manal**

Soutenu publiquement comme exigence partielle en vue de l'obtention du

Diplôme de Master en Systèmes d'Information et Systèmes Intelligents

Devant le jury composé de :

- **Mme. EL BAJTA Manal**
- **Mme. ELHARI Kaoutar**
- **M. SAIDI Mohamed Nabil**

Septembre 2022

Acknowledgements

First and foremost, praises and thanks to the God, the Almighty, for His showers of blessings throughout my research work to complete the research successfully.

At the end of the writing of this report, it seems appropriate to begin by presenting my warmest thanks to all the people who contributed to the elaboration of this work. I would also like to express my warmest thanks to all the teaching staff of the National Institute of Statistics and Applied Economics, and more particularly to all the trainers for their help that they brought me throughout the two years.

I would like to express my deep and sincere gratitude to my research supervisor, Dr. EL BAJTA Manal, professor at the National Institute of Statistics and Applied Economic, for giving me the opportunity to do research and providing invaluable guidance throughout this research. She has taught me the methodology to carry out the research and to present the research works as clearly as possible. It was a great privilege and honor to work and study under her guidance.

I would like to express my deep gratitude to all the members of the jury, Dr. ELHARI Kaoutar and Dr. SAIDI Mohamed Nabil, for having accepted to for having accepted to read this report and to evaluate this work.

I am extremely grateful to my parents for their love, prayers, caring and sacrifices for educating and preparing me for my future. Also I express my thanks to my brother and sister who supported and guided me throughout my studies.

Dedication to my colleagues in Master M2SI and especially my friends EL CHAABI Mohamed and Abdessalam BAHAFID. Dedication also to my friend Anouar who gave me advices during the whole internship period and to my friend Mohamed Amine FARFAOUA whom I met at INSEA. I do not forget to express my most sincere greetings to my friends Moncef, Abdelaziz and Yassine that I know for a long time.

Finally, my thanks go to all the people who have supported me to complete the research work directly or indirectly, and I hope that this final project report will live up to everyone's expectations.

Abstract

This report is a brief overview of my final thesis which lasted in the **SI2M laboratory at the National Institute of Statistics and Applied Economics**, in which we will approach the problematic of **Software Effort Estimation** in the field of software engineering. The work propose an estimation approach based on machine learning techniques, which will be partitioned into several phases. The first phase will be assigned to the literature review, where we will extract and analyze a set of articles related to our research topic, in order to collect the maximum of techniques and approaches used in the context of Machine Learning. And to ensure the collection of correct and useful information as well as orientate our research, we will rely on the research methodology that we will elaborate later. The second phase is dedicated to the implementation, where we will start with a brief description of the dataset used during this works, then we will apply statistical tests that will be used to select the relevant variables from the dataset. Then, we will apply the polynomial transformation of degree 3 on the extracted variables, so that we apply after the Sequential Feature Selection method. The next phase is dedicated to the estimation of the effort, through several machine learning techniques, namely Linear Regression, Support Vector Machines (SVMs), Random Forest (RF) and Artificial Neural Networks (ANNs) with their default parameters. We are applying genetic algorithms, which is an optimisation technique, in order to find the best hyper-parameters that will improve the estimation of the effort. Finally, the quality of the estimated effort will be assessed by several regression metrics, namely Mean Absolute Error (MAE), Mean Magnitude of Relative Error (MMRE) and PRED.

Keywords— Software Effort Estimation, Machine Learning, Neural Networks, Linear Regression, Support Vector Machines, Random Forest, Sequential Feature Selection, Genetic Algorithms.

Contents

List of Figures

List of Tables

Acronym

ANNs Artificial Neural Networks. 1, iii, 6

ANOVA Analysis of Variance. iii, 24

GA Genetic Algorithm. iii, 14

KNN k-Nearest Neighbours. iii, 5

Lasso Least Absolute Shrinkage and Selection Operator Regression. iii, 7, 28

MAE Mean Absolute Error. 1, iii

ML Machine Learning. iii

MLP Multilayer Perceptron. iii, 15

MMRE Mean Magnitude of Relative Error. 1, iii

PSO Particle Swarm Optimization. iii, 14

REPTree Reduced Error Pruning Tree. iii, 7

RF Random Forest. 1, iii, 42

SEE Software Effort Estimation. iii, 3, 4

SEERA Software EnginEeRing in SudAn. iii, 2, 16, 42

SFS Sequential Feature Selection. iii, 27, 42

SVMs Support Vector Machines. 1, iii, 5

SVR Support Vector Regression. iii, 7, 29, 42

Chapter 1

INTRODUCTION:

1.1 Context:

Nowadays, the use of software grows continuously. So, to keep up with this growth, companies must produce software of high quality and in time. The success of a software development project requires a good and early planning. Therefore, planning has become an important phase in each software development process. Among the planning phases that contribute to the success of a software development project, there is Estimation of Effort.

In software engineering, Effort Estimation has become an important task for companies. It is the process of forecasting how much effort is required to develop or maintain a software application. This effort is traditionally measured in the hours worked by a person, or the money needed to pay for this work (Cost). Effort estimation helps draft project plans and budgets in the early stages of the software development life cycle. This practice enables a project manager or a product owner to allocate the resources needed, as overestimation leads to make the resources remain unproductive and underestimation leads to incomplete features or defective products and eventual failure.

In this context, a great importance has been dedicated to this phase, to find the best methods to accurately predict the effort of software development process.

1.2 Problem statement:

Software Effort Estimation is a difficult phase, a step that must be characterized by the necessary precision, to avoid any mistake in the estimation. As mentioned above, any under or over estimation may lead to waste of resources or a project failure.

As solutions, many methods have attempt to make accurate and efficient estimations of the effort, which can be divided into non-parametric and parametric approaches, such as Expert Judgment, Estimation by Analogy, Function Point Analysis on one side, and COCOMO Model for example on the other side. But these methods, when applied to software effort estimation, have limitations, and considered as traditional, due to errors and bias that human can commit during the process of estimation. Function Point Analysis is a manual method that requires detailed information requirement for estimation of software size using function points and needs experience to make an accurate estimation. Estimation by Analogy relies on similar projects to the current one, a need of lots of information is required as well. In that case, a lack of information may lead to an inaccurate estimation. In Expert Judgment method, it is hard to document the factors used by the experts or experts-group, furthermore, the expert may be some biased, optimistic, and pessimistic, even though they have been decreased by the group consensus. COCOMO model, as well, has several drawbacks, as it is based on number of Lines of Code, so it ignores the hardware issues as well as the personal turnover level, it also ignores customer skills cooperation, and knowledge, factors that may be of great importance in the process of the software effort estimation.

Due to the rapidly growth of the software engineering field, as well as the emergence of new factors that affect the effort needed to a software development, it is hard to keep using these methods. This obliges us to develop new quick and more accurate solutions.

1.3 Proposed work:

In order to remedy the problems we have mentioned before, we will rely on modern methods known for their efficiency and simplicity in order to carry out the effort estimation in a relevant way. Our proposed work will focus on Machine Learning techniques, which are known for their accuracy since they are based on historical data from similar projects. So, we have implemented 4 Machine Learning models based on the following algorithms: SVM, Linear Regression, Random Forests and Neural Networks that we will use for effort estimation during this project and that we have trained on Software EnginEeRing in SudAn (SEERA) dataset.

The report is organized as follows:

- **Chapter 2** will be dedicated to the various works in the literature up to 2022. We will develop our research methodology on which we will rely to find works related to our research topic and objectives. Then, we will analyze the results of our research in order to extract the maximum of useful information that will be useful to us.
- **Chapter 3** is for the **Contribution** section, where we will, first of all, detail the information about the chosen database as well as the selection criteria. Next, we will explain the steps followed in the preprocessing of the data in order to start the modeling. Then, we will detail the architecture of the resulting models. And to improve the performance of our models, we will discuss in detail all the techniques used to optimize the hyper-parameters of each model, list the results and compare them before and after the optimization process.
- **Chapter 4** is the final chapter in this report, where we will summarize the entire work.

Chapter 2

LITERATURE REVIEW

2.1 Introduction:

The development of efficient effort estimation methods has been a research subject for a long time. These research that we will rely on to build a broad idea about the Software Effort Estimation (SEE) as well as all Machine Learning methods to solve this problem. For that purpose, we will try to investigate several works that may answer our questions, based on some criteria to select the relevant papers.

The entire process of our review will be described in the following steps: Firstly, we will raise a set of questions, to which we will try to find answers that may lead us to our objective. Secondly, we will propose a search strategy in which we will define a set of research terms and determine the selection criteria to select the relevant papers. Then, we will move to the step of data extraction, and finally discuss the results. The entire methodology is shown on the following figure:

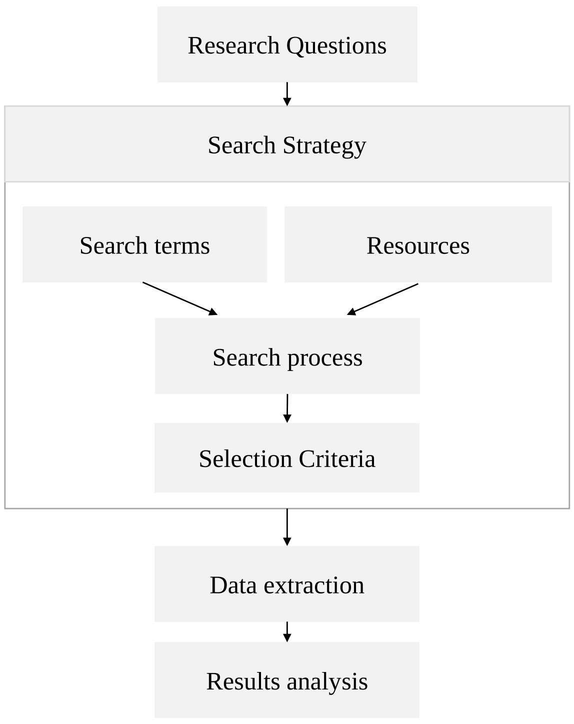


Figure 2.1: Search Methodology

2.2 Research question:

To target this research to our objective, and extract the maximum of relevant information from this review, we defined a set of questions described as:

1. **RQ1**: Which publication sources and channels are investigated to select the papers?
2. **RQ2**: How has the frequency of Software Effort Estimation research changed over the time?
3. **RQ3**: Which Machine Learning algorithms have been used for SEE?
4. **RQ4**: What are the metrics and the criteria to determine the best models?
5. **RQ5**: Which Machine Learning algorithms are ranked best?
6. **RQ6**: Which techniques are used to improve the accuracy of the estimation?

2.3 Search strategy

Defining a search strategy is an important step in our process, so we can search in reliable sources, as well as find relevant works. In this step, we determine search terms, sources, and criteria to select the papers.

First, we identified a set of search sources, where research was carried out during the period of 2006-2021:

- ACM (https://dl.acm.org/): The ACM Digital Library (DL) is the world's most comprehensive database of full-text articles and bibliographic literature covering computing and information technology. This renowned repository includes the complete collection of ACM publications plus an extended bibliographic database of core works in computing from scholarly publishers.
- SpringerLink (http://link.springer.com): offers electronic and printed literature from Springer-Verlag, a preeminent scientific publisher with a reputation for excellence spanning more than 150 years.
- IEEE Xplore Digital Library (https://ieeexplore.ieee.org): Research database for discovery and access to journal articles, conference proceedings, technical standards, and related materials on computer science, electrical engineering and electronics, and allied fields.
- ScienceDirect (https://www.sciencedirect.com): ScienceDirect is the world's leading source for scientific, technical, and medical research. Explore journals, books, and articles.

Next, to determine search terms, we derived major terms and keywords from research questions and defined synonyms or alternative spelling of each word. Then we used Boolean keywords to link these terms, such as **OR** and **AND**. The resulting search terms are as follows: Software **AND** (Effort **OR** Cost) **AND** (Estimation **OR** Prediction) **AND** Machine Learning **AND** (Techniques **OR** Algorithms **OR** Methods **OR** Methodologies **OR** Metrics).

After developing our search terms, we started our research based on some criteria, since many papers do not provide the information needed to answer our questions. So, we defined some inclusion and exclusion criteria for the selection. Furthermore, the selection study was carried out by reading titles, abstracts, conclusions, and visualizing figures.

Inclusion criteria:

- **IC1.** Papers related to the software effort estimation.
- **IC2.** Using machine learning techniques.
- **IC3.** Comparative analysis of two or more machine learning techniques.
- **IC4.** Combining two machine learning algorithms or more as a hybrid model.

Exclusion criteria:

- **EC1.** Estimating only software size, time.
- **EC2.** Estimating software testing effort.
- **EC3.** Papers in other languages than English.
- **EC4.** Papers not published in journals or conferences.

Therefore, by applying these criteria, we extracted several papers, in which we identified 19 that we judged relevant for our research.

2.4 Data extraction:

During this step, we analyzed the selected papers to collect data that will contribute to addressing the research questions of this review. The data extraction process was based on answering the research questions:

- **RQ1**: This question aims to identify the title, the year, and the source or the channel on which the paper was published.
- **RQ2**: We will classify the selected papers per publication year.
- **RQ3**: The purpose of this question is to identify ML algorithms that have been used to solve the problem of software effort estimation. As result, we identified 19 Machine Learning algorithms as follows:
  - SVMs: SVM is a set of machine learning methods used in many areas, such as classification and regression. SVM classifier separates the instances from two different classes by using a hyper-plane which tries to maximize the margin. This increases the generalization capability of the classifier. The instances that are close to or on the border are called the support vectors. The number of support vectors also represents the complexity of the model.

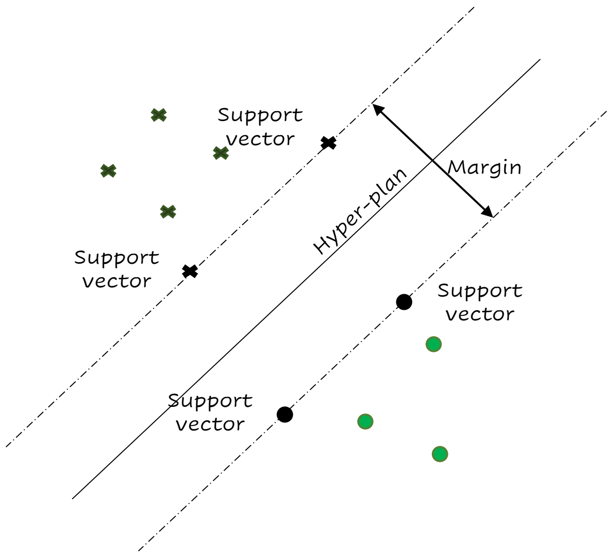


Figure 2.2: SVC linear kernel

  - k-Nearest Neighbours (KNN): KNN is one of the techniques used for classification problems, and it is one of the simplest classification techniques that should be the first option for a classification study when there is no past knowledge about data description. KNN works first by computing distance between an instance with other instances and find the k-nearest neighbor for that instance, then it predicts the class.

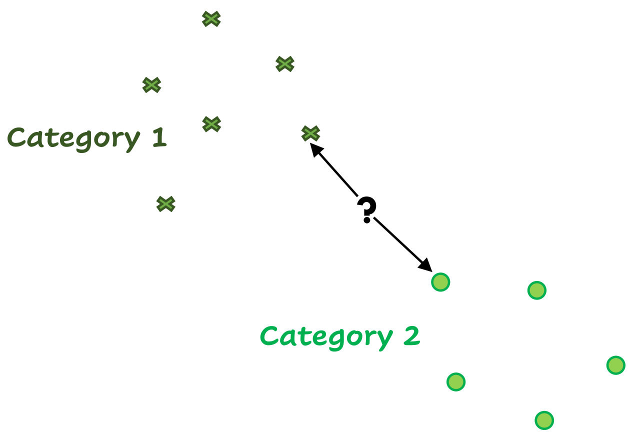


Figure 2.3: KNN algorithm

– Linear regression: This Algorithm is used to express the data and find the correlation between the dependent variable and one or more independent nominal, or level variables.

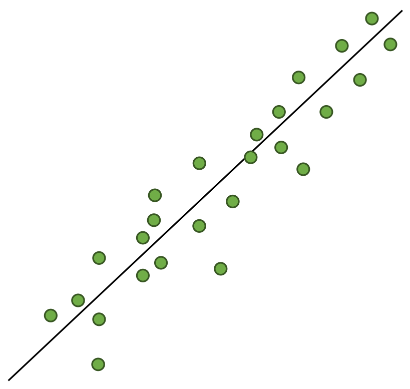


Figure 2.4: Linear Regression algorithm

– Decision Tree: Decision Tree classifier is a tree structure consisting of nodes and branches. Internal nodes represent attributes. Branches represent decisions and leaf nodes are the outcomes that can be either categorical or continuous variable. Thus, decision trees can be used for both categorical and regressive problems

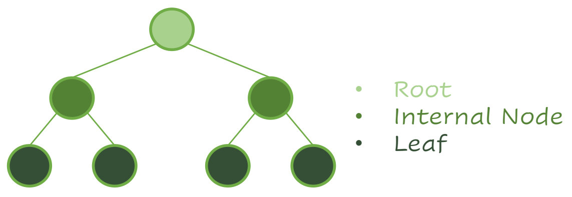


Figure 2.5: Decision tree

– Random Forest: A supervised learning algorithm that is used for both classification as well as regression. However, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, a random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution.

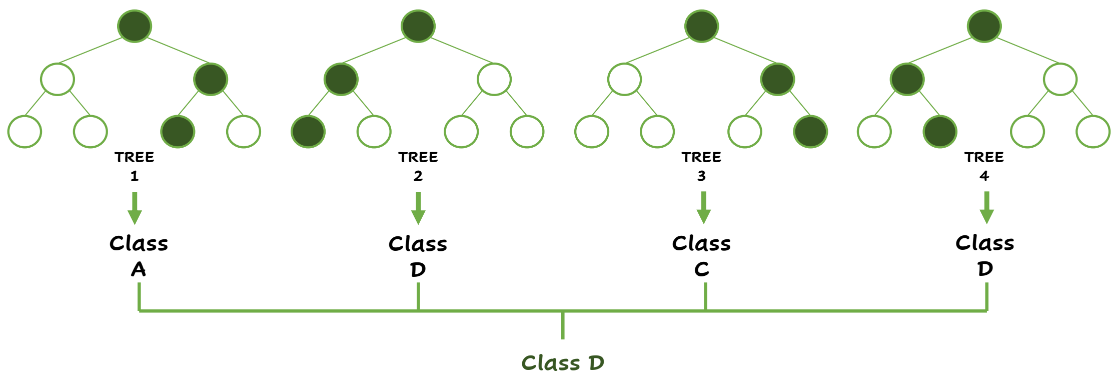


Figure 2.6: Random forest algorithm

– Neural Networks: a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another. ANNs are comprised of a node layer, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

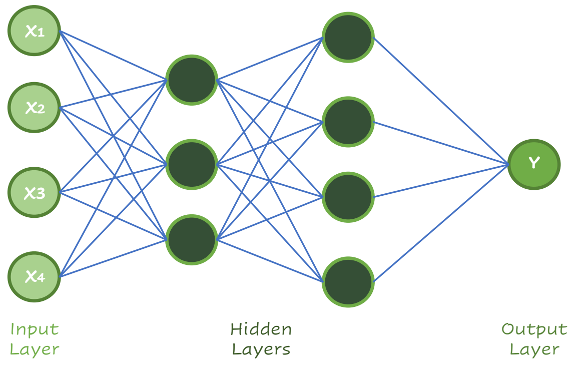


Figure 2.7: Neural network architecture

- K-Star: The K-star algorithm uses similarity measurements to classify the data based on the classes' likelihood. It acts as instance classifier based on an entropy-based distance function to calculate the distance between instances.
- Additive Regression: Additive Regression classifier improves the performance of classifiers that are based on regression. Each iteration done uses the residuals generated from previous iterations. It can overcome over-fitting problem, but it takes extra time.
- Reduced Error Pruning Tree (REPTree): The tree is being built in a fast and learnable way depending on the gained information. REPTree is another kind of decision trees that uses regression tree, which can create many trees in various rounds or iterations. Then, out of all the generated trees, the best one is being selected. To do the pruning process for the tree, the mean square error is measured based on the tree predictions.
- Logistic Regression: Logistic regression is based on creating non-linear mathematical model for the purpose of predicting categorical data. Logistic regression predicts the values of an outcome or dependent variable with a dichotomous value from a given set of independent variables.
- AdaBoost regressor: AdaBoost regressor is an ensemble learner that boosts weak learners and produces high accuracy. It begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction.
- Ridge Regression: Ridge regression is a method of estimating the coefficients of multiple-regression models in scenarios where linearly independent variables are highly correlated. It has been used in many fields including econometrics, chemistry, and engineering.
- Least Absolute Shrinkage and Selection Operator Regression (Lasso): Lasso Regression is similar to ridge regression, but it uses L1 regularization technique to minimize error between actual and predicted value.
- ElasticNet Regression: Elasticnet regression is the combination of both Ridge and Lasso regression. It uses both L1 and L2 regularization technique. This type of regression is used when there are more number of features and when they suffer with multi collinearity.
- Regression Tree: A regression tree is built through a process known as binary recursive partitioning, which is an iterative process that splits the data into partitions or branches, and then continues splitting each partition into smaller groups as the method moves up each branch.
- M5P: M5P method is another tree model that is being constructed based on Quinlan's M5 algorithm. Originally, in addition to adding the linear regression method to the tree leave nodes, M5 model is also based on the conventional decision tree. The trained data is used by the algorithm to form the nodes and represent the decision tree model.
- Naïve Bayes: Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features (see Bayes classifier). They are among the simplest Bayesian network models, but coupled with kernel density estimation, they can achieve high accuracy levels.
- Support Vector Regression (SVR): Support Vector Regression is a supervised learning algorithm that is used to predict discrete values. Support Vector Regression uses the same principle as the SVMs. The basic idea behind SVR is to find the best fit line. In SVR, the best fit line is the hyperplane that has the maximum number of points.

  - Radial Basis Function: A radial basis function network is a type of supervised artificial neural network that uses supervised machine learning (ML) to function as a nonlinear classifier. Nonlinear classifiers use sophisticated functions to go further in analysis than simple linear classifiers that work on lower-dimensional vectors. A radial basis function network is also known as a radial basis network

- **RQ4**: What are the metrics and the criteria to determine the best models?
  - Root Mean Square Error (RMSE): RMSE computes the difference between value estimated by a model and the value observed. It is the square root of the mean square error, as given in equation:

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{1}{N} * (X_i - Y_i)^2} \tag{2.1}$$

  - Magnitude of Relative error (MRE): MRE is a commonly-used measure which gives the difference between values estimated by suggested model and the values actually estimated

$$MRE = \frac{|X_i - Y_i|}{X_i} \tag{2.2}$$

  - Mean Magnitude of Relative Error (MMRE): MMRE, is the mean measurement of the absolute values of the relative errors from complete data set, as given in equation:

$$MMRE = \frac{1}{N} \sum_{i=1}^{N} \frac{|X_i - Y_i|}{X_i} \tag{2.3}$$

  - PRED: [1] PRED(x) considers the average fraction of the MRE's off by no more than x as defined by:

$$PRED(x) = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} 1 & \text{if } MRE_i \leq x. \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

  - Mean Absolute Error (MAE): The mean absolute error is a measure of how far the estimates are from actual values. MAE is defined as:

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |X_i - Y_i| \tag{2.5}$$

  - Mean Square Error (MSE): It is the average of square of errors (27) in the data set and is given by the following equation:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (X_i - Y_i)^2 \tag{2.6}$$

  - Accuracy: Classification accuracy is a metric that summarizes the performance of a classification model as the number of correct predictions divided by the total number of predictions. The formula is given by the following equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

  - Relative Absolute Error (RAE): The Relative Absolute Error is a relative measure that compares the performance of a predictive model with the performance of a simple model. The performance of the predictive model is defined as the total absolute difference between the realized and predicted values (i.e., the error):

$$RAE = \frac{\sum_{i=1}^{N} |X_i - Y_i|}{\sum_{i=1}^{N} |X_i - \bar{Y}_i|} \tag{2.8}$$

  - Precision: Precision quantifies the number of positive class predictions that belong to the positive class.

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

  - Recall: Recall quantifies the number of positive class predictions made of all positive examples

$$Recall = \frac{TP}{TP + FN} \tag{2.10}$$

  - Coefficient of determination: A measure of linear correlation between two sets of data. It is the ratio between the covariance of two variables and the product of their standard deviations. Thus it is essentially a normalized measurement of the covariance, such that the result always has a value between -1 and 1.
$$R^2 = 1 - \frac{\sum_{i=1}^{N}(X_i - Y_i)^2}{\sum_{i=1}^{N}(X_i - \bar{Y_i})^2} \tag{2.11}$$
  - Relative Squared Error (RSE): The relative squared error is relative to what it would have been if a simple predictor had been used. More specifically, this simple predictor is just the average of the actual values. Thus, the relative squared error takes the total squared error and normalizes it by dividing by the total squared error of the simple predictor.
$$RSE = \frac{\sum_{i=1}^{N}(X_i - Y_i)^2}{\sum_{i=1}^{N}(X_i - \bar{Y_i})^2} \tag{2.12}$$
  - Median Magnitude of Relative Error (MdMRE)

  where:

  - X_i: Actual value of data point i.
  - Y_i: The predicted value of data point i.
  - N: The total number of data points.
  - $\bar{Y_i} = \frac{1}{N}\sum_{i=1}^{N} X_i$ the average of the realized values.

- **RQ5**: The 5th research question aims to find which algorithms are ranked best in each experiment.
- **RQ6**: We will try, in this question, to determine which techniques are used to increase the performances of our models. In our case, we determined the following techniques:
  - Cross validation: Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model. In typical cross-validation, the training and validation sets must cross-over in successive rounds such that each data point has a chance of being validated against.
  - Feature selection: Feature selection consists of reducing the number of input variables when developing a predictive model. It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model.
  - Discretization: In statistics and machine learning, discretization refers to the process of converting or partitioning continuous attributes, features or variables to discretized or nominal attributes / features / variables / intervals.
  - Hyper-parameters tuning: Hyper-parameters correspond to the adjustment parameters of machine learning algorithms. In order to find the best parameters, we apply hyper-parameters optimization, which is the process of finding the hyper-parameter configuration that produces the best performance.
  - Ensemble learning: In Machine Learning, an ensemble is a machine learning model that combines the predictions from two or more models.

2.5 Results:

- **RQ1. Source and Channel of publication:** As mentioned above, we identified 19 papers to analyze, published either in a journal or a conference, during the period of 2006-2022. The following table shows more details:

	Title	Year	Journal/Conference
[2]	Software Development Effort Estimation Using Ensemble Machine Learning	2017	Int'l Journal of Computing, Communications Instrumentation Engg.
[3]	Predictive analytics approaches for software effort estimation: A review	2020	Indian Journal of Science an Technology
[4]	Comparative Analysis on prediction of Software Effort Estimation Using Machine Learning Techniques	2020	SSRN Electronic Journal
[5]	Software Effort Estimation Using Machine Learning Techniques	2014	Indian Software engineering Conference
[6]	GA-based method for feature selection and parameters optimization for Machine Learning regression applied to software effort estimation	2010	Indian Software engineering Conference
[7]	Features-level Software Effort Estimation Using Machine Learning algorithms	2018	2018 International Conference on Innovation and Intelligence for informatics, Computing, and Technologies
[8]	Software Effort Estimation Using Machine Learning methods	2007	2007 22nd international symposium on computer and information sciences.
[9]	ENNA: Software effort estimation using ensemble of neural networks with associative memory.	2008	Proceedings of the 16th ACM SIGSOFT International Syposium of Foundations of Software Engineering.
[10]	Predicting Software Effort Estimation Using Machine Learning Techniques	2018	2018 8th International Conference on Computer Science and Information Technology (CSIT).
[11]	Comparative analysis of Machine Learning and Deep Learning algorithms for Software Effort Estimation	2021	Journal of Physics Conferences series.
[12]	Machine Learning Models for Software Cost Estimation	2019	2019 International Conferences on innovation and Intelligence for Informatics, Computing, and Technologies.
[13]	Machine Learning Classification to Effort Estimation for embedded Software Development projects.	2019	2019 International Conferences on innovation and International Journal of Software Innovation
[14]	Software Effort Estimation using Machine Learning Techniques with Robust Confidence Intervals.	2007	19th IEEE International Conference on Tools with artificial Intelligence
[15]	Estimation of software project effort with Support Vector regression.	2006	Neurocomputing
[16]	Enhanced Software Effort Estimation using Multi Layered Feed Forward Artificial Neural Network Technique.	2016	Twelfth International Multi-Conference on Information Processing-2016
[17]	A Principled Evaluation of Ensembles of Learning Machines for Software Effort Estimation	2011	Promise '11: 7th International Conference on Predictive Models in Software Engineering.
[18]	Investigating the use of random forest in software effort estimation	2019	Second International Conference on on Intelligent Computing in Data Sciences (ICDS 2018).
[19]	Hyperparameters tuning of ensemble model for software effort estimation	2020	Journal of Ambient Intelligence and Humanized Computing
[20]	Bayesian Hyperparameter Optimization and Ensemble Learning for Machine Learning Models on Software Effort Estimation	2022	International Journal of Advanced Computer Science and Applications

Table 2.1: General information of analyzed papers

- **RQ2. Publication distribution per year:** The following figure shows the number of publications per year. We can see that the interest of research in Software Effort Estimation SEE during the period of 2018-2020.

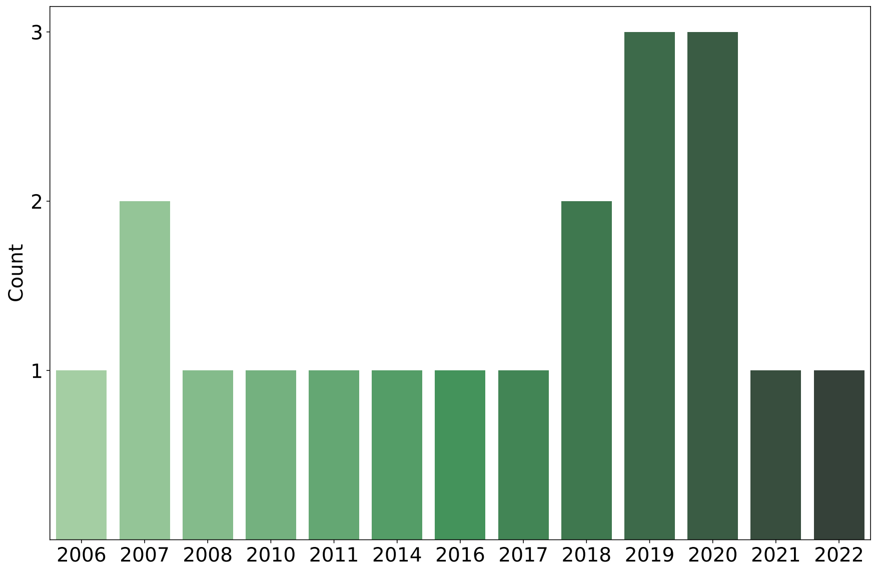


Figure 2.8: Publication per year

- **RQ3. Machine Learning algorithms:** There are 19 different algorithms that have been used during the experiments, and we can see, in the figure bellow, that Neural Networks is the most frequent used one. We can see that the use of Random Forest, SVR, Linear Regression, and Regression Trees has been essential in many works as well. The following figure shows the number of uses of each algorithm:

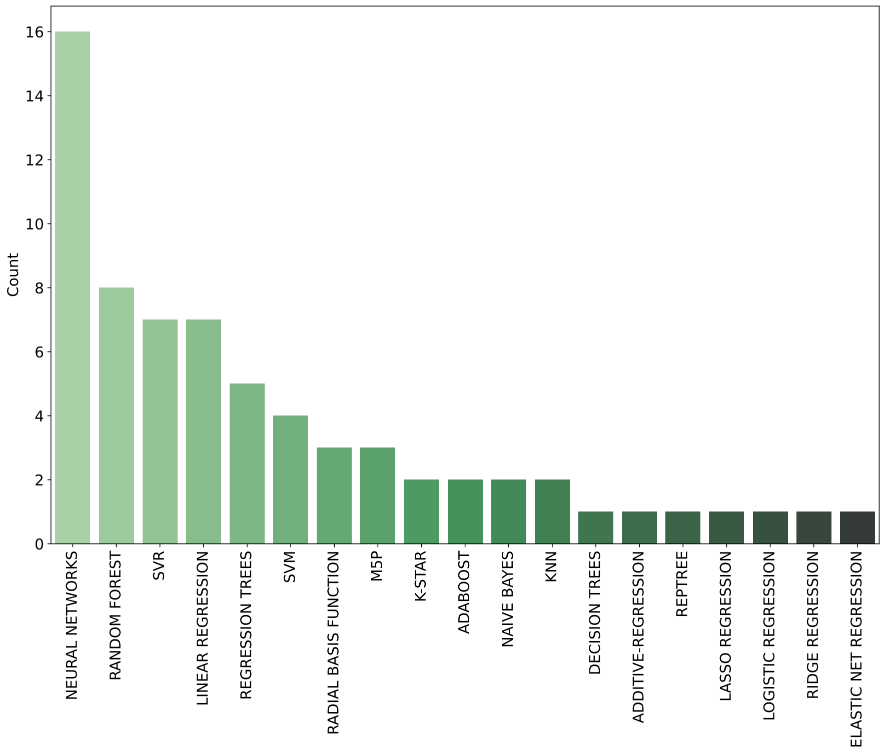


Figure 2.9: Machine Learning algorithms rates

The following table shows the details about which Machine Learning Techniques were used in each work:

Machine Learning Algorithm	Paper
SVR	[6], [8], [13], [14], [15], [19], [20]
SVM	[2], [3], [7], [11]
Regression Trees	[8], [9], [17], [18], [20]
Random Forest	[3], [11], [10], [11], [12], [18],[19] ,[20]
Neural Networks	[3], [4], [5], [6], [7], [8], [9], [11], [13], [12], [14], [15], [16], [17], [19], [20]
Linear Regression	[3], [13], [16], [4], [7], [12], [19]
K-Star	[7], [12]
Naïve Bayes	[10], [5]
Logistic Regression	[10]
KNN	[2], [20]
M5P	[12], [12], [14]
Additive-Regression	[12]
Decision Trees	[16]
ADABOOST	[20], [19]
RepTree	[12]
Ridge Regression	[16]
Lasso Regression	[16]
Elastic Net Regression	[16]
Radial Basis Function	[8], [15], [17]

Table 2.2: The use of algorithms in each paper

- **RQ4. Score of Machine Learning Algorithms:** As result, Neural Networks were ranked 5 times as the best algorithm, followed by Random Forest and SVR with a score of 4, and so on as shown in the figure below. We can see that assembling many algorithms may be a great solution to make a strong and accurate model as well.

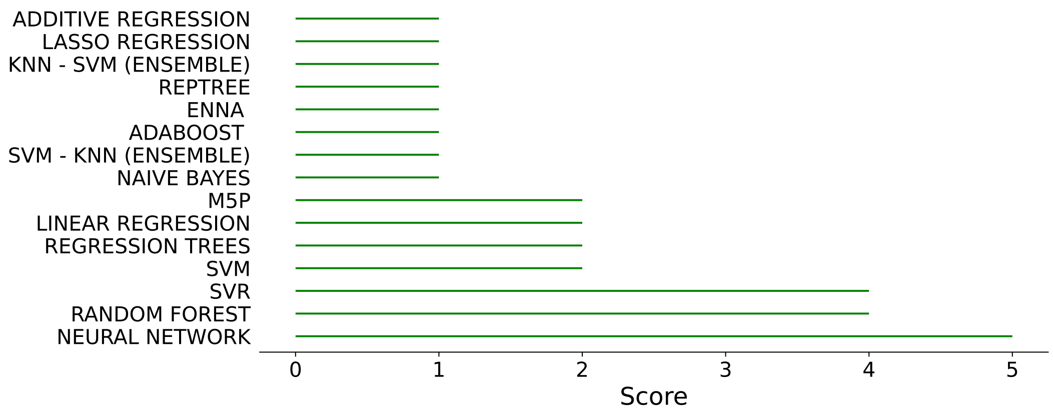


Figure 2.10: Score of Machine Learning algorithms

- **RQ5. Evaluation metrics:** And besides, metrics are essential to decide if a model is good or not, as they measure errors between the real values and the predicted ones. In this benchmark, MMRE has been used several times with a frequency of 20%, followed by PRED, MAE and R-SQUARE. The figure below shows all metrics used in this benchmark, as well as the frequency of use each.

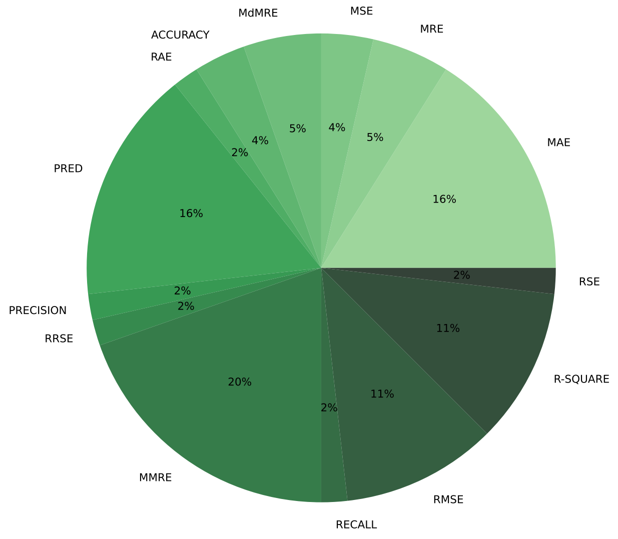


Figure 2.11: Metrics

- **RQ6. Machine Learning models quality improving techniques:**
  - **Discretization:** Discretization may be a solution for the software effort estimation, as it was used in few works. But we cannot judge it as a great solution as only 3 works out of 19 only used this technique.

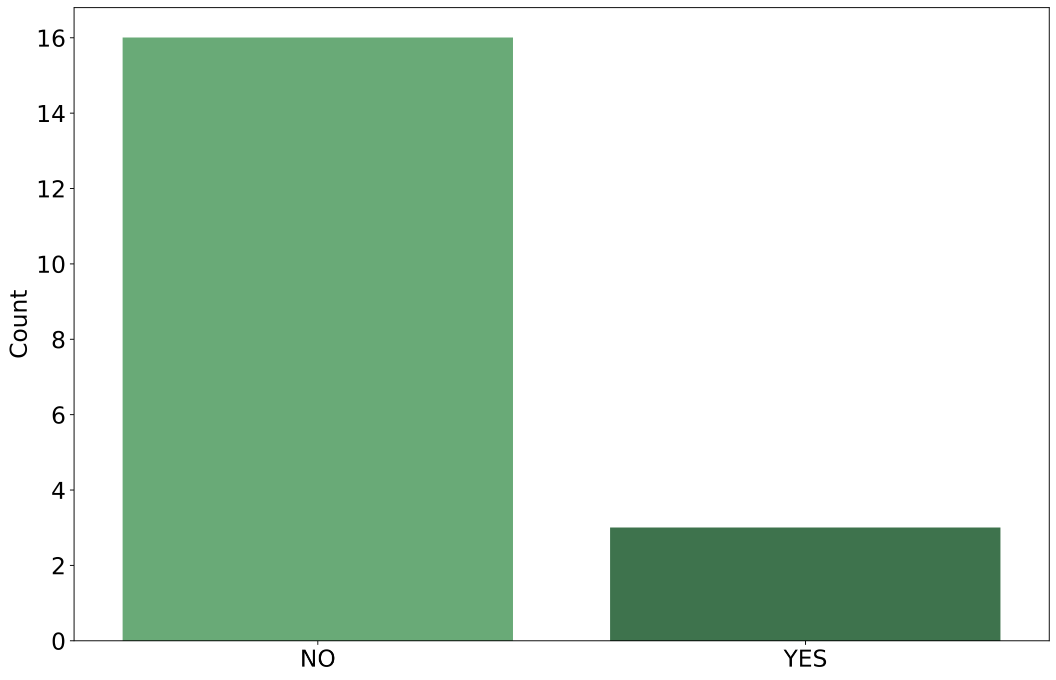


Figure 2.12: Discretization use rate

– **Cross-Validation:** Cross-Validation technique is one of the most techniques used to validate the performance of a Machine Learning model, a re-sampling method that uses different portions of the data to test and train a model on different iterations. This technique has been often used in these experiments, as 12 of 19 adopted this method.

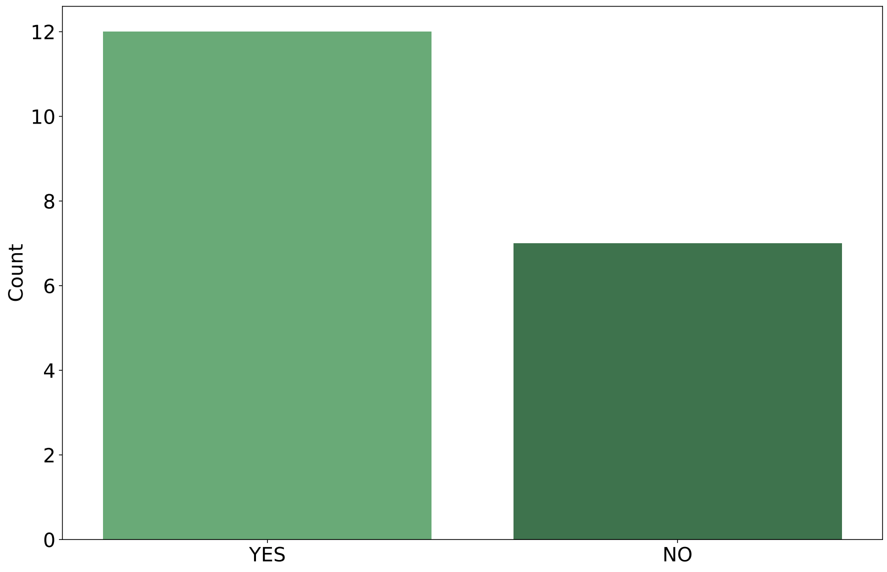


Figure 2.13: Cross-Validation use rate

– **Feature selection:** In the experiments discussed above, [5] considered Rough Set Analysis for feature reduction of dataset. This method reduced the number of input features from 15 to 6 including the effort. An important property of rough set is that the reduced set of attributes provides the same quality of information as the original set of attributes. The set of attributes of the reduced information system, known as Reduct, is independent and no attribute can be further eliminated without losing some information from the system. [8] used Principal Analysis Component PCA as a method for dimension reduction, which produced a decrease of MMRE error. Note that PCA is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest. On the other hand, to eliminate independent variables which has least correlation with target variable, [19] has applied Least Absolute Shrinkage and Selection Operator (LASSO), which is a regression analysis method that performs both variable selection and regularization to enhance the prediction accuracy and interpret-ability of the resulting statistical model. Finally, Genetic Algorithm is a solution proposed by [6] for feature selection and parameters optimization. Genetic algorithm is a meta-heuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover and selection.

– **Hyper-parameters optimization:** There are many techniques applied to hyper-parameter tuning. For instance, Bayesian with Gaussian Process Optimization is a technique used by [20] to optimize the hyper-parameters of machine learning models, which aims to minimize an objective function $f : X \to R^+$, where X is the hyper-parameters domain. This technique consists on predicting the target value in historical values of hyper-parameters, by applying a number of iterations until a certain stopping criteria is reached. This process is as follows: Define the historical model, 2) Find the optimal parameter, 3) Apply the detected hyper-parameter to the objective function, 4) Update the model with new result, and 5) 2-4 steps are repeated until the threshold value is reached or the process exceeds the limited time.
Meanwhile, Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) are used to minimize the MMRE value of the stacked models. Concerning the PSO, it is a stochastic approach that consists of initializing each particle (parameters) in a random way and using a fitness function to

evaluate each one. At each iteration, the function evaluates the current solution and compares it with the one of the previous iteration, if there is an improvement, the current solution is stored as local solution, otherwise we keep the previous one. On the other hand, Genetic algorithm initializes a 64 chromosomes (A chromosome is a set of parameters) named population and evaluate each chromosome by a fitness function. The chromosomes are sorted according to their scores in order to pick the top 16 as parents. These parents will subjected to cross-over and mutation operations and repeat the process until the terminal condition is reached [19] [6].

– **Ensemble Learning:** Combining Machine Learning models is also a solution for increasing the accuracy of prediction, as it consists of combining two or more Machine Learning models in a strong one.
  For example, [2] opted for Adaptive Boosting algorithm, known also as AdaBoost, by boosting by SVM algorithm then KNN and by KNN then SVM, which increased the accuracy of the estimation compared to using each single algorithm alone. [9] idea consists of working with ensembles of neural networks with associative memory ENNA, by using 20 Multilayer Perceptron (MLP) trained on different training on different training sets obtained by bootstrapping method and applying the concept of Associative Memory to correct and estimate the bias, as there may still be considerable bias of the ENN model. Bagging is one of the most well-known ensembles learning approaches in the literature. It creates diversity by training learning each base learner with a different training set generated by sampling with replacement from the available training data. In this context, [20] used the following machine learning algorithms: Bagging with MLPs, with RBFs and with RTs; Random with MLPs; and Negative Correlation Learning (NCL) with MLPs. Bagging with MLPs was able to outperform the other Machine Learning algorithms. [19] used stacking ensemble which combines diverse base machine learning algorithms via a Meta learner. The hyperparameters of the base learners and the meta learner is tuned using PSO and GA which explores the vast hyperparameter space and identifies the optimum hyperparameter values of the ensemble model. PSO method yields slightly better performance in terms of accuracy than GA, which shows significantly better results than the setup without hyperparameter tuning.
  Finally, [20] developed an AdaBoost Ensemble learning based on Random Forest algorithm using Bayesian hyperparameter optimization.

2.6 Discussion:

This section has been focused on the literature review of the use of Machine Learning techniques for Software Effort Estimation. First, we have developed a methodology that we will be following during the research process. Then, we have defined the research questions that we will answer to extract the data from the papers. Next, we determined the search strategy, by defining the search terms, sources, inclusion and exclusion criteria to select the appropriate papers. This led us to extract 19 papers that met the criteria previously set, to start the data extraction phase. We have described in detail the objective of each research question and have given a brief definition of each Machine Learning algorithm, each metric, and other useful techniques in the field of Machine Learning.

As results, during our analysis, we found that the research interest in SEE was focused on the period of 2018-2020, that several Machine Learning algorithms have been used in this field, but only 4 frequently used, namely Neural Networks, Support Vector Machines, Linear Regression and Random Forest. In addition, they were ranked among the best algorithms that give better results. Likewise for the metrics, several have been used to validate the performance of the final models, but MMRE, PRED, MAE and R-SQUARE are the most used. Furthermore, there are other techniques that may increase the accuracy of the predictions, such as feature selection, assembling many ML algorithms and hyper-parameters optimization.

2.7 Conclusion:

From this literature review, we conclude that the use of SVR, Linear Regression, Random Forest and Neural Network has been very frequent and ranked as better than other algorithms. So, we will implement 4 models based on these 4 algorithms. We will apply statistical tests to select the features that are significantly related to the target variable. Then we will make a polynomial transformation of degree 3 in order to apply to select the definitive variables with which we will train our models. Finally, to get better results, we will apply the optimization techniques for hyper-parameters tuning.

Chapter 3

CONTRIBUTION

3.1 Introduction:

This chapter is dedicated to the description of the proposed worked. It is structured in the following way:

- Description of the data used for Software Effort Estimation.
- A brief look at the implementation tools used, including the programming language, software and libraries.
- The data pre-processing & features selection steps.
- The set of Machine Learning algorithms used and their architectures.
- Basic concepts and the architecture of genetic algorithms for hyper-parameters tuning.
- Results of each Machine Learning model before and after optimization.
- Discussion and conclusion.

3.2 SEERA dataset description:

SEERA is a dataset for technically and economically constrained environments. It is the result of the collection of 120 software development project data from 42 organizations in Sudan. The SEERA dataset contains 76 attributes with both qualitative and quantitative attributes. This dataset is considered in this study to perform effort estimation based on several criteria:

1. SEERA dataset is a set of software development projects from a technically and economically constrained environment, containing detailed attributes and factors that are more suitable to the cost and schedule information available within these environments [21].
2. During our search process, we did not find any studies or papers that worked on SEERA.
3. SEERA dataset is a new open-source one, collected from June 2019 to February 2020. [21].
4. It contains a high number of projects (120 projects) unlike many open-source datasets (e.g., Desharnais (81), Maxwell (62), Albrecht (24), Cocomo81 (63), Keremer (15)) and a high number of features (76 features) (e.g., Desharnais (12), Maxwell (27), Albrecht (7), Cocomo81 (17), Kemerer (7)) [22].

To collect this data, a questionnaire was designed that incorporated internal software project costing factors and factors specific to the software development industry in developing countries [21]. The following table gives general information about this questionnaire:

Subsection	**Description**	**Number of questions**
Organization environment	Income policies, development environment, impact of public policy and economic instability.	15
Users	Requirements stability and flexibility, top management support, user availability and resistance.	13
Team	Team experience, cohesion, continuity, and capability	18
Project Management	Scheduling, outsourcing, reuse, technical stability, risk management, use of standards.	20
Product	Reusability and documentation.	5
Product complexity	Technical and quality constraint	5

Table 3.1: Questionnaire subsections: Factors affecting software

Finally, **Table 3.2** contains the list of both categorical and numerical attributes.

Category	**Categorical attributes**	**Numerical attributes**
General Information	Organization type Role in organization Size of organization Size of IT department Customer organization type Development type Application domain	ProjID Year of project Organization id Actual duration % Project gain(loss)
Size	——	Object points Other sizing method Estimated size
Effort	——	Estimated effort Actual effort
Environment	Government policy impact Organization management structure clarity Developer hiring policy Developer training	Contract maturity Economic instability impact Developers incentives policy Development team management
Users	Top management opinion of previous system Clarity of manual system User computer experience Users' stability	Top management support User resistance Requirement stability Requirements flexibility
Developers	Project manager experience Consultant availability DBMS expert availability Software tool experience Programmers experience in programming language Team selection Income satisfaction	Precedentedness Programmers capability Analysts capability Team size Dedicated team members Daily working hours Team contracts Team continuity Team cohesion
Project	Schedule quality Methodology Programming language used DBMS used Open source software Level of outsourcing Outsourcing impact Degree of software reuse Process reengineering	Development environment Adequacy Tool availability Multiple programing languages Technical stability Degree of risk management Degree of standards usage
Product	Requirement accuracy level Technical documentation Comments within the code User manual Required reusability Product complexity Reliability requirements Specified H/W	Performance requirements Security requirements

Table 3.2: SEERA dataset attributes

3.3 Implementation tools:

The success of an IT project requires good programming tools to speed up and facilitate the tasks that can be encountered throughout the development process. For a Machine Learning project, we will need different tools that will allow us to visualize the data in a simple and efficient way, as well as different libraries for the implementation of the models in order to optimize the data processing complexity, and of course a programming software. Here is the list of tools we used:

- **Python:** Python is an interpreted programming language, multi-paradigm and cross-platform. It supports structured, functional and object-oriented imperative programming. object-oriented programming. This language has propelled itself to the forefront of infrastructure management, data analysis or in the field of software development. Python is the most used programming language programming language used in the fields of Machine Learning, Big Data and Data Science.
- **Sklearn:** Scikit-learn is a free Python library for machine learning. learning. It is developed by many contributors, especially in the academic world by French higher education and research institutes. It includes functions for estimating random forests, logistic regressions, classification algorithms classification algorithms, and support vector machines. It is designed designed to harmonize with other free Python libraries, notably NumPy and SciPy.
- **Tensorflow:** TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow was developed by the Google Brain team for internal Google use in research and production. The initial version was released under the Apache License 2.0 in 2015. Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019.
- **Keras:** Keras is a neural network API written in Python. It is an open source library, running on top of frameworks such as Theano and TensorFlow.
- **Numpy:** Numpy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- **Pandas:** pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term ”panel data”, an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase ”Python data analysis” itself.
- **Matplotlib:** is a library of the Python programming language for plotting and visualizing data in graphical form. It can be combined with the NumPy and SciPy scientific computing libraries.
- **Seaborn**:Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- **VS Code:** Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

3.4 Data pre-preprocessing & Feature Selection:

The quality of the final data used to train the machine learning models is important in order to obtain good results. For this purpose, we must perform what is known as data pre-preprocessing, followed by feature selection. The first version of the data could be incomplete, irrelevant, or redundant. In addition, we must find the set of the features that explains the relationship between the dependent and independent variables in an effective way.

The following figure describes the framework of data pre-preprocessing & feature selection:

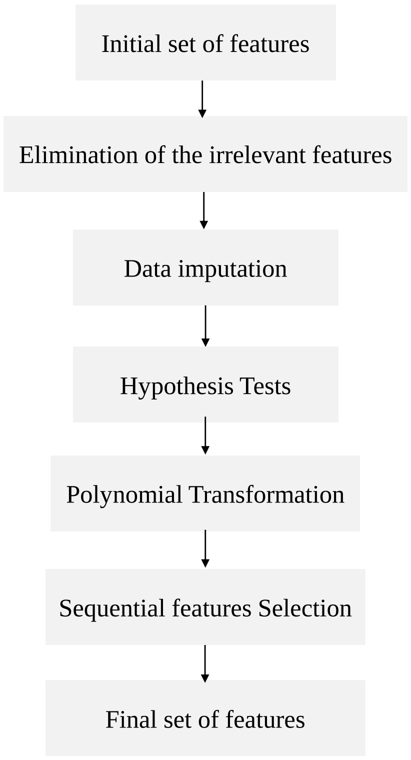
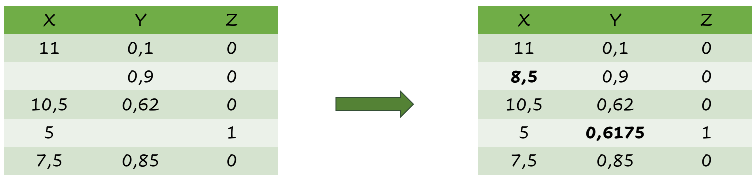

Initial set of features
Elimination of the irrelevant features
Data imputation
Hypothesis Tests
Polynomial Transformation
Sequential features Selection
Final set of features

Figure 3.1: Framework of data pre-preprocessing & feature selection

3.4.1 Basic concepts:

– **Data imputation:** In statistics, imputation is a technique used for replacing the missing data with some substitute value to retain most of the data/information of the dataset.



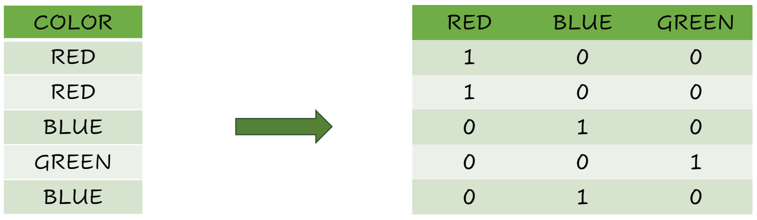
X	Y	Z
11	0,1	0
	0,9	0
10,5	0,62	0
5		1
7,5	0,85	0

X	Y	Z
11	0,1	0
8,5	0,9	0
10,5	0,62	0
5	**0,6175**	1
7,5	0,85	0

Figure 3.2: Example of imputation by mean

For our dataset, the continous numerical features are filled by the mean, while the rest of the dataset is imputed by the mode.

– **OneHot Encoding:** OneHot encoding is a pre-processing technique for categorical variables, such that for each categorical variable, the classes belonging to this variable are represented by new binary columns. And so we break down the initial variables into several sub-variables, creating as many columns as we have classes.



COLOR
RED
RED
BLUE
GREEN
BLUE

RED	BLUE	GREEN
1	0	0
1	0	0
0	1	0
0	0	1
0	1	0

Figure 3.3: Example of OneHot Encoding

- **Polynomial Transformation**: Polynomial features transformation is a type of features engineering techniques that consists in creating new features from the original features. For instance, a dataset with only one feature $\mathbf{X}$ gets a new feature which is $\mathbf{X}^2$ if the degree of the polynomial transformation is 2. The figure below shows the difference before and after the transformation in a ML model.

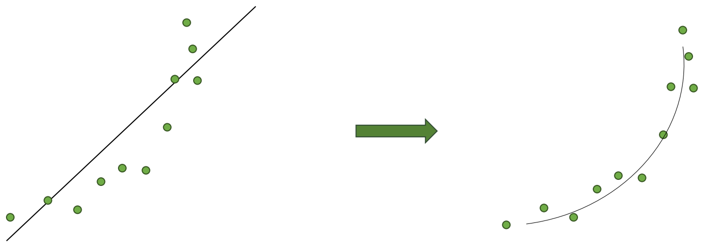


Figure 3.4: Simple example of Polynomial Transformation (degree = 2)

- **Data Scaling:** Data scaling is an essential step in data prepossessing, whose objective is to handle highly varying magnitudes or values or units. It consists in reducing the range of the data into [0,1].The formula of data scaling is as follows:

$$newX_i = \frac{X_i - \mu}{\sigma} \tag{3.1}$$

where:

  - X_i: The current value of X_i.
  - μ: The mean of the sample.
  - σ: The standard deviation of the sample.

3.4.2 Irrelevant columns:

The first stage of data pre-processing is to identify the irrelevant columns. The columns that we considered irrelevant are:

- **ProjID:** The id of the project.
- **Organization id:** The id of the organization.
- **Year of project:** The year the project was carried out.
- **Estimated effort**: Represents the estimation of the effort for a project, but since the estimate may not be 100% correct, our models should not rely on human estimations to perform predictions.
- **Estimated duration**: Same as Estimated effort, we have judged Estimated duration as irrelevant column.

Then, we calculated the percentage of null values in each column, and decided to delete the columns with a percentage higher than 40%. The following table shows the percentage of each dropped column:

Column name	Null values percentage
Outsourcing impact	90.8333%
Estimated size	89.1667%
Degree of standards usage	82.5%
% project gain (loss)	46.6667%

Table 3.3: Dropped columns based on null values percentage

In addition, columns with a variance of less than 30% were also removed.

3.4.3 F regression test for numerical data:

F regression test is an univariate linear regression test which uses a linear model for testing the individual effect of each of many features. This test is implemented in **SKlearn**, which starts by calculating the correlation between the target and the features and and converts it to an F score with a p-value. The f regression score of a dependant and an independent variables is calculated by the following formula:

$$f\_regression\_score = \frac{r(X,Y)^2}{1 - r(X,Y)^2} * k \tag{3.2}$$

where:

- $r(X,Y)$: the correlation coefficient of X and Y.
- X: The independent variable (Feature) .
- Y: The dependent variable (Target).
- $\bar{X}$: The mean value of X.
- $\bar{Y}$: The mean value of Y.
- k: degree of freedom.

In our case, we selected features with a p-value less than 0.05. The details are shown in the following table:

Column	f_regression score	p-value
Team size	287.441018	0.0000
Dedicated team members	280.423008	0.0000
Object points	103.135779	0.0000
Actual duration	21.600395	0.0000
Requirment stability	11.883554	0.0008
Technical stability	9.598140	0.0024
Team cohesion	9.437774	0.0026
Tool availability	8.008831	0.0055
Economic instability impact	7.151680	0.0086
Developer incentives policy	7.073320	0.0089
User resistance	6.478658	0.0122
Development environment adequacy	5.933972	0.0163
Users stability	5.780526	0.0178
Contract maturity	5.306773	0.0230
Precedentedness	4.020347	0.0472

Table 3.4: F regression test results

The **Figure 3.5** shows the scatter plots of the selected numerical features and the Actual Effort.

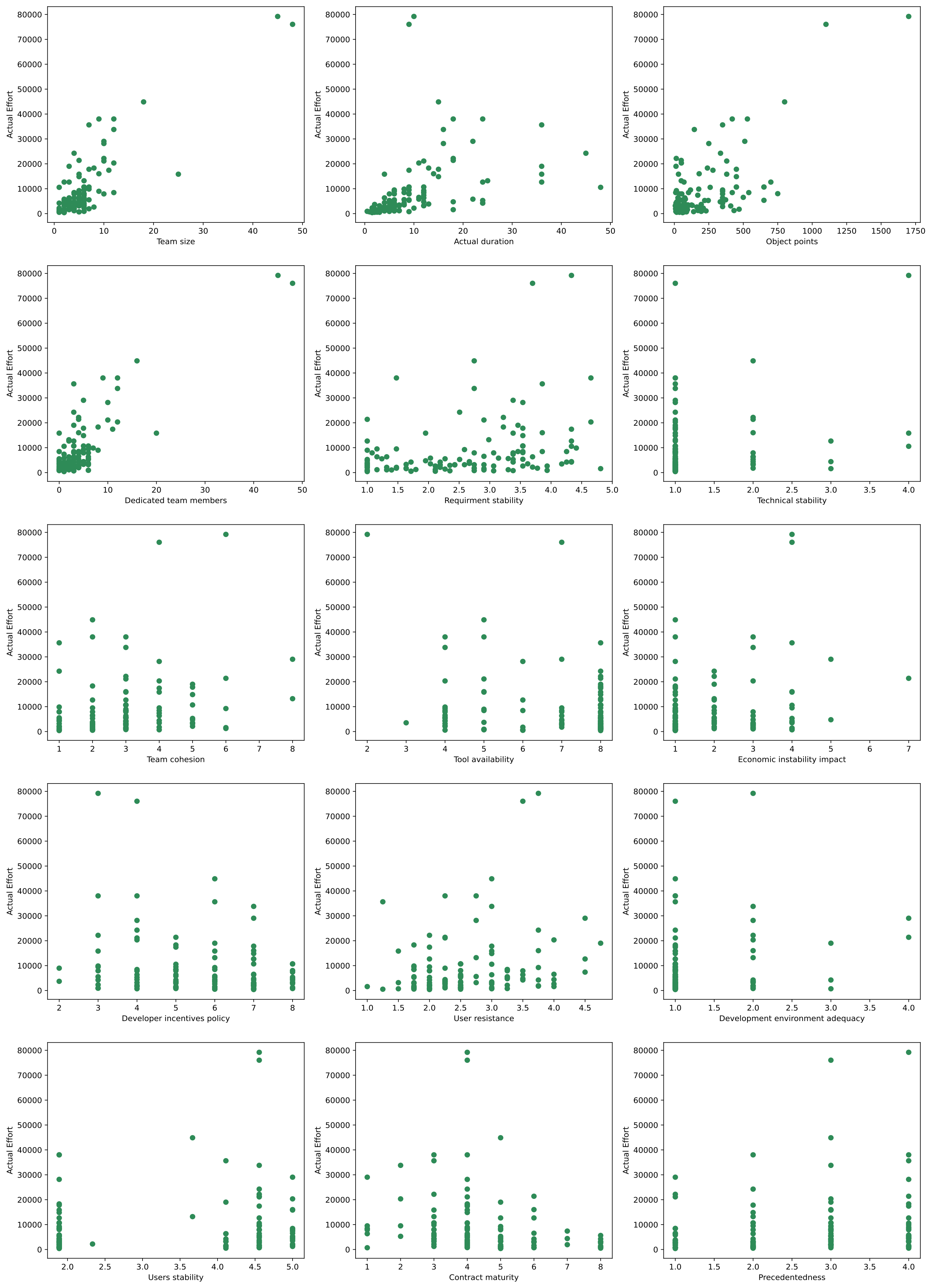


Figure 3.5: Scatter plots of Actual effort and numerical features

3.4.4 Anova test for categorical data:

Analysis of Variance (ANOVA) is statistical test developed by Ronald Fisher, which is used to analyze the differences among means. In other words, it allows to find out whether the differences between groups of data are statistically significant.

The general formula of Anova test is as follows:

$$F = \frac{MeanSquares_{Between}}{MeanSquares_{Within}} \tag{3.3}$$

Where:

- $MS_w = \frac{SS_w}{df_w}$ The mean squares within the groups.
- $MS_b = \frac{SS_b}{df_b}$ The mean squares between the groups.
- $SS_b = \sum_{j=1}^{k}(\bar{X}_j - \bar{X})^2$ The sum of squares between the groups.
- $SS_w = \sum_{j=1}^{k}\sum_{j=1}^{l}(X - \bar{X}_j)^2$ The sum of squares within the groups.
- $df_w = k - 1$ Degree of freedom.
- $df_b = n - k$ Degree of freedom.
- X: Individual observation.
- $\bar{X}_j$: Sample mean of the jth group
- N: The total number of samples in a population

Seera dataset several categorical features, and by applying the Anova test on these variables, we could select 4 variables. The table below contains the score of each selected variable as well as the p-value.

Column	Anova score	p-value
Role in organization	3.962587	0.0000
Development type	2.947634	0.0001
Size of organization	1.908943	0.0087
Requirement accuracy level	1.791512	0.0158

Table 3.5: Anova test results

The Figure 3.6 shows the box plots of the selected categorical features and the Actual Effort.

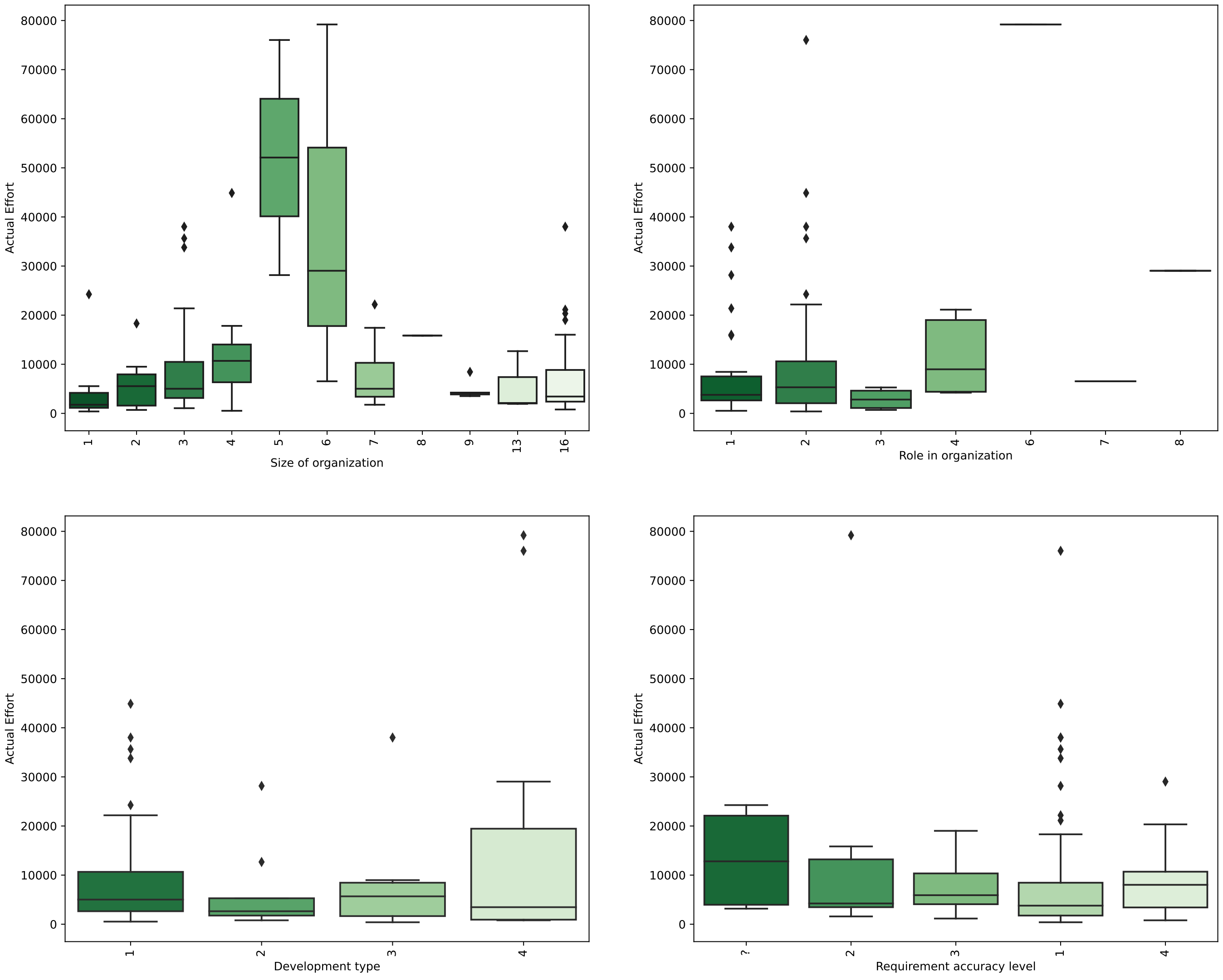


Figure 3.6: Box Plots of Actual effort and categorical feautres

After having determined the categorical features, we have applied the OneHot encoding to get new columns.

3.4.5 The new dataset:

As mentioned above, the categorical features are OneHot encoded, which allowed the creation of new columns from the modalities that exist in the categorical variables. We obtained 41 columns as a result in the new dataset, that are described in the following table:

Original Feature	**New Feature**
Team size	Team size
Dedicated team members	Dedicated team members
Object points	Object points
Actual duration	Actual duration
Requirment stability	Requirment stability
Technical stability	Technical stability
Team cohesion	Team cohesion
Tool availability	Tool availability
Economic instability impact	Economic instability impact
Developer incentives policy	Developer incentives policy
User resistance	User resistance
Development environment adequacy	Development environment adequacy
Users stability	Users stability
Contract maturity	Contract maturity
Precedentedness	Precedentedness
Role in Organization	Developer Planning coordinator Project manager Company Manager System administrator Technical consultant Technical manager
Development type	Customization of imported software Modifying existing software New software development Upgrading existing software
Size of Organization	1-5 6-10 11-20 21-30 31-40 41-50 51-100 101-150 151-200 351-400 > 500
Requirement Accuracy level	Accurate requirements specifications used to develop the software system Inaccurate requirements specifications and required re-programming the software system Inaccurate requirements specifications and required the re-analysis of the software requirements Inaccurate requirements specifications and required the re-design of the software system

Table 3.6: New dataset features

After defining the new dataset, we have applied a **Polynomial Transformation of degree 3** on these features which created several new features, so we can apply the **Sequential Feature Selection** in the next section in order to select the final features for training our models.

3.4.6 Sequential Feature Selection:

Sequential Feature Selection (SFS) is a wrapper method is whose objective is to extract a subset of variables from an original set that will maximize the performance of a Machine Learning model.

The concept of this method is to create an empty set in which we will insert the best features, and starts by adding the first variable that will produce the maximum performance. Next, the remaining features are added one by one to current subset and then we evaluate the performances. The selected feature to be added to the best features set is the one that maximises the performance of the model. The process is repeated until we get the subset of features that will maximise the model's performances[23] [24].

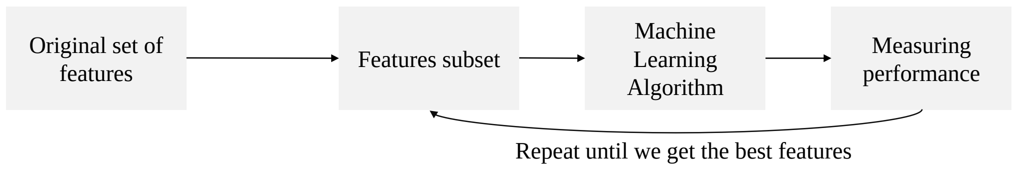

Original set of features
Features subset
Machine Learning Algorithm
Measuring performance
Repeat until we get the best features

Figure 3.7: Wrapper method

In our case, the implemented algorithm trains a Linear Regression model with each variable in the remaining variables with the best variables selected in the previous iterations. And for each remaining variable, we calculate its p-value and if it is less than 0.05, we add this variable to those previously selected. This loop ends when all the variables in the dataset have been run.

Note that, before starting the variable selection process, the categorical variables have been OneHot encoded and then the degree 3 polynomial transformation was applied on the whole features set.

Algorithm 1 Sequential Feature Selection

```
Require: initialFeatures, targetVariable, pValueThreshold
Ensure: bestFeatures
  bestFeatures ← $\emptyset$
  while length(initialFeatures) $\geq 0$ do
      remainingFeatures ← initialFeatures \ bestFeatures
      pValues ← $\emptyset$                    ▷ Empty dictionary with remainingFeatures as keys
      for each feature $\in$ remainingFeatures do
          currentFeatures ← bestFeatures $\cup$ feature
          Fit a model with currentFeature and targetVariable
          pValues(feature) ← The pValue of feature in the model
          minPValue ← $min$(pValues)           ▷ Get the lowest pValue in the pValues dictionary
          if minPValue $\leq$ pValueThreshold then
              Append the feature with minPValue to bestFeatures
          else
              break
          end if
      end for
  end while
  return bestFeatures
```

Finally, this entire process allowed us to identify 42 features that we will be using to train the models.

3.5 Machine Learning algorithms:

3.5.1 Multiple Linear regression:

Multiple linear regression, also known as multiple regression, is a statistical technique derived from simple linear regression **(2.4)** that aims to find the appropriate mathematical equation between two or more features and the dependent variable in order to minimize the error between the real values and the predictions. The following formula describes the equation given for a problem with p features and n observations:

$$\hat{Y}_i = \theta_0 + \theta_1 X_{i1} + \theta_2 X_{i2} + \theta_3 X_{i3} + ... + \theta_p X_{ip} + \epsilon \tag{3.4}$$

where:

- $i = 1, ..., n$: The number of observations.
- $p = 1, .., , p$: The number of independent variables.
- $\hat{Y}_i$: The predicted value.
- X_p: The independent variables.
- θ_p: The weight of each independent variable.
- θ_0: Constant term.
- ϵ: The error.

The cost function for the linear regression is equal to:

$$Cost(\theta) = \frac{1}{2N}[\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2] \tag{3.5}$$

Figure 3.8 shows a 3d plot of linear regression model with 2 features.

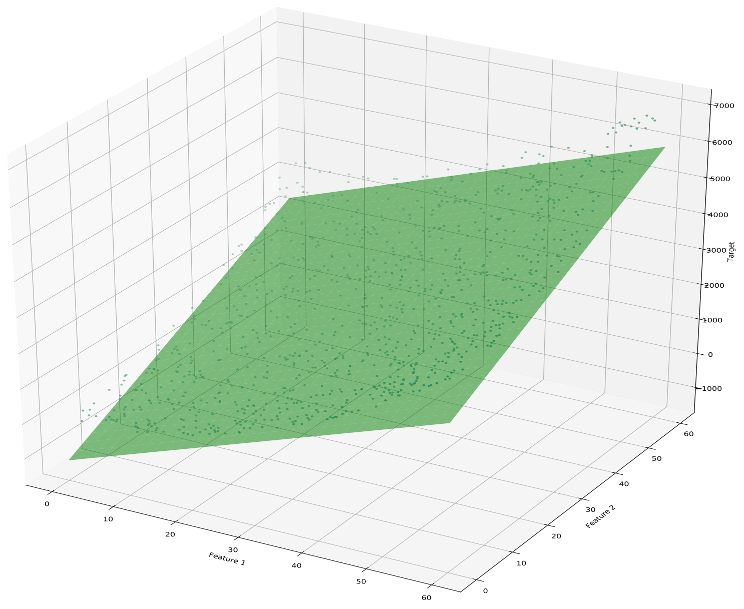


Figure 3.8: Example of 3D plot of Linear regression model with 2 features

In order to improve the linear regression performances, we will apply the linear regression with regularization technique or what is called Lasso, where the model is penalized for the sum of absolute values of the weights. Thus, the absolute values of weight will be reduced, and many will tend to be zeros. The difference is in the cost function, where we add another term to the normal linear regression 3.5:

$$Cost(\theta) = \frac{1}{2N}[\sum_{i=1}^{N}(Yi - \hat{Y}_i)^2 + \lambda \sum_{j=1}^{M} \theta_j^2] \tag{3.6}$$

where:

- $\lambda \sum_{j=1}^{M} \theta_j^2$: The regularization term.
- λ: The regularization coefficient.

3.5.2 Linear Support Vector Regression:

Besides classification, SVM's **(2.4)** are used as well for regression problems, with what is called Support Vector Regression.

The concept of Support Vector Regression is to maximise the margin, which means minimize the equation **(3.7)** which is the sum of the norm of the weights and errors measured by the slack variables. This function is subject to two constraints that indicate that all observations must be inside the margin defined by ϵ [15].

In this work, we consider SVR with linear kernel, which is computed as $K(x_i, x_j) =< x_i, x_j >$.

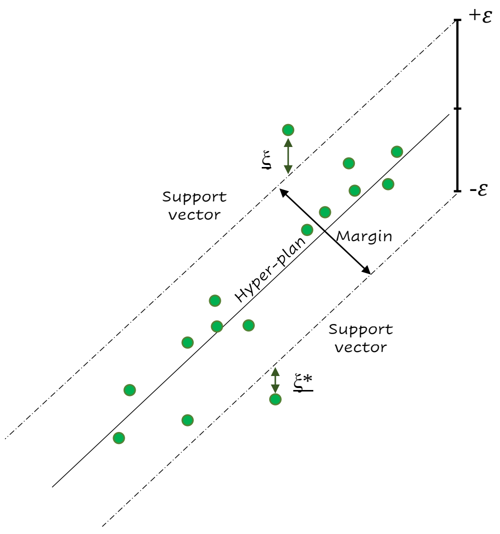


Figure 3.9: Support vector regression with linear kernel

$$\begin{aligned}
\min_{\theta,b,\xi,\xi^*} \quad & \frac{1}{2} < \theta, \theta > + C \sum_{i=1}^{N} (\xi_i + \xi_i^*) \\
\text{s.t.} \quad & Y_i - \hat{Y}_i \leq \epsilon + \xi_i \\
& \hat{Y}_i - Y_i \leq \epsilon + \xi_i^* \\
& \xi_i, \xi_i^* \geq 0
\end{aligned} \tag{3.7}$$

where:

- $\hat{Y}_i = \theta x_i + b$: The predicted value.
- Y_i: The real value.
- $\xi, \xi*$: The slack variables that measure the errors of points outside the support vectors.
- C: The penalisation to balance the trade-off.
- $< \theta, \theta >$: The norm of the weights.

3.5.3 Random Forest Regression:

Random Forest Regression a bagging technique that combines multiple regression trees to perform the predict value. For a given number of trees N, Random Forest Regressor builds and trains N Regression Trees in order to get the predict value of each one and provide the average as final result.

$$\hat{Y} = \frac{1}{N} \sum_{i=1}^{N} RT_i(X) \tag{3.8}$$

where:

- $\bar{Y}$: The final predict value.
- N: The number of regression trees.
- $RT_i(X)$: The predicted value of tree i.
- X: A set of features.

Bagging technique consists on training each regression tree on a portion of the data, by creating bootstrap samples from the training data set and then built trees on bootstrap samples and then aggregating the output from all the trees and predicting the output.

3.5.4 Neural Networks:

The architecture of the proposed neural network is made up of a first layer of units that allow to read the data: each unit corresponds to one of the input variables. We can add a bias unit which is always activated (it transmits 1 whatever the data). These units are connected to a single output unit, which receives the sum of the units connected to it, weighted by connection weights. The following equation describes the sum of N units by weights:

$$\theta_0 + \sum_{i=1}^{N} \theta_i x_i \tag{3.9}$$

Since we are solving a regression problem, the activation function used is linear, expressed by the following equation:

$$f(x) = x \tag{3.10}$$

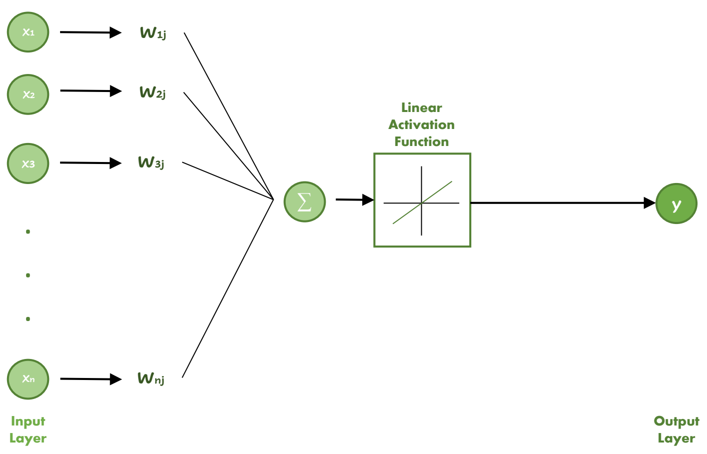


Figure 3.10: Single-Layer Neural Network with linear activation function

Note that this architecture is initial, since we will apply optimization techniques to find the optimal parameters to improve the results.

3.6 Hyper-parameters tuning:

3.6.1 Genetic Algorithms:

Genetic algorithms are optimization method belonging to the family of evolutionary algorithms, introduced by **John Holland** in the 70's. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

Genetic algorithms have been used in several fields, such as machine learning, because their concept allows to develop very efficient solutions to a given problem, in a rather short time. The main advantage of these algorithms is that they do not need any starting examples to learn, no base is required for their learning.

In our current situation, we will use these algorithms to optimize the hyper-parameters of our models, in order to improve the results and obtain better scores, but first we will detail some basic notions:

1. Initial population: The process starts by generating a random population of possible solutions to our optimization problem, where a population is a set of individuals (chromosomes) and each individual is composed of genes (Parameter). **Figure 3.11** shows a population of N individuals, where each individual is composed of 5 genes.

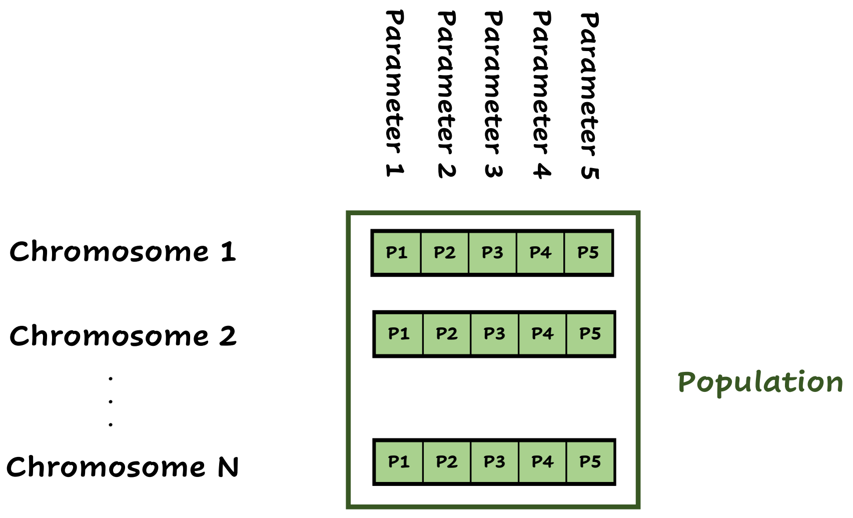


Figure 3.11: Example of population

2. Fitness function: This is the function to be optimized, which returns the score of each individual and determines the chances for an individual to be chosen for the selection operation. Our fitness function returns the 5 KFlod cross validation score of the individual on the train set.
3. Selection: The selection operation allows to select the two best individuals which have the highest scores, in order to pass their genes to next generations. The type of selection we will use is Elitist selection, which is a technique where the individual who obtains the best score in his generation automatically passes to the next generation. We select the first two individuals with the best scores.
4. Crossover: The concept of crossover is to create new chromosomes called offsprings, by swapping genes from previously selected parents. There are several crossover techniques, the one used in our problem is uniform crossover. The principal of uniform crossover is to flip a coin for each chromosome, if the result is equal to 0, the parents swap the genes, otherwise they keep them. The coin in genetic algorithms is represented by the Mask.

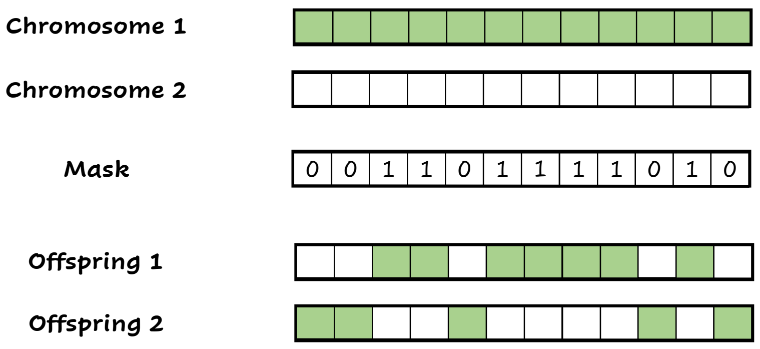


Figure 3.12: Example of uniform crossover

5. Mutation: The offsprings created in the crossover operation are subject to a mutation, where for a certain probability, we randomly select a gene, and we change its value.

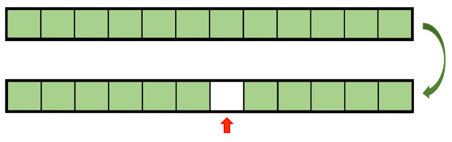


Figure 3.13: Example of mutation

Figure 3.14 summarizes the process of the developed genetic algorithm program, which begin with generating a random population and sort it with the fitness function. If stopping criteria are reached, the program ends. Else, the population is subjected to selection, crossover and mutation, and redo the same process until convergence.

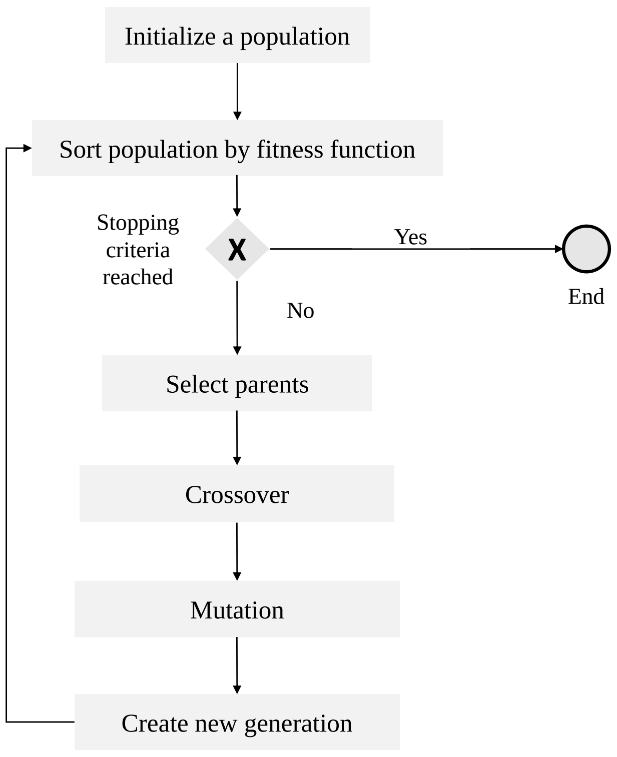


Figure 3.14: The framework of the implemented genetic algorithm

Finally, here is the pseudo-code of the algorithm, which takes as input the estimator, features and target, the fitness limit and time limit that determine the criteria for stopping the program, the probability and number of mutations.

Algorithm 2 Genetic algorithm program

```
Require: estimator, parameters, X, y, populationSize, fitnessLimit, timeLimit, mutationProbability, mutNum
Ensure: population
  population ← Initial Population    ▷ Generate a random and sorted population by the function score
  while (FitnessFunction(population(0)) < fitnessLimit & time < timeLimit) do
      parent1, parent2 ← SelectionPair()    ▷ Return two parents for cross over and mutation
      offSpring1, offSpring2 ← UniformCrossover(parent1, parent2)
      mutantOffSpring1, mutantOffSpring2 ← Mutation(off_spring1), Mutation(off_spring2)
      population ← {parent1, parent2, offSpring1, offSpring2, mutantOffSpring1, mutantOffSpring2}
      population ← SortPopulation(population)    ▷ Resort the population
  end while
  return population    ▷ Returns the final population which contains the best individuals
```

where:

- estimator: The estimator to tune.
- parameters: The set of parameters as shown in **Table 3.6.2**.
- X: Features (Train set).
- y: Target (Train set).
- populationSize: The number of chromosomes in population.
- fitnessLimit: The maximum score to be reached to stop the program.
- timeLimit: The maximum running time to stop the program.
- mutationProbability: The minimum probability to make a mutation.
- mutNum: The number of parameters to be muted.

3.6.2 Parameters and ranges:

Table 3.6.2 contains the details of the set of parameters used to tune the models we used, a description of each parameter and the ranges.

Model	Parameter name	Description	Range
Lasso	alpha	Controlls regularization strength	0-2
	max_iter	Maximum number of iterations.	10, 10^2, 10^3, 10^4
	positive	Whether the coefficients are positive	True, False
	precompute	Use a precomputed Gram matrix	True, False
	selection	Controlls the convergence speed	cyclic, random
	tol	The tolerance for the optimization	10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}
Random Forest Regressor	n_estimators	The number of trees	1-250
	max_depth	Depth of the tree	1-20
	min_samples_split	Min of samples to split an internal node	2-10
	min_samples_leaf	Min of samples to be at a leaf node	1-10
	max_features	Features size to consider when splitting a node	sqrt, log2, None
	bootstrap	Bootstrapping samples when building trees	True, False
	criterion	Measures the quality of a split	squared_error, absolute_error, poisson
Neural Network	momentum	Leads to faster converging	10^{-3}, 10^{-2}, 10^{-1}
	batch_size	Num samples processed before the model is updated	10, 11, 12, 15
	learning_rate	Learning rate for weight updates	10^{-3}, 10^{-2}, 10^{-1}
	dropout	Regularization method	0-0.5
LinearSVR	epsilon	Epsilon-insensitive loss function	0-2
	tol	Tolerance for stopping criteria	0-0.1
	max_iter	Maximum number of iterations to be run	10, 10^2, 10^3, 10^4
	C	Regularization parameter	0-100

Table 3.7: Description of hyper-parameters

3.7 Results:

This section will contain all the results obtained by running the genetic algorithms. We will first quote the configuration of the genetic algorithm for each model, as well as the maximum score reached. Then, we will list all the hyper-parameters obtained by the algorithm. And finally, we will make a comparison of the results before and after the application of the optimization algorithm.

3.7.1 Genetic algorithm results:

First of all, the objective of the implemented genetic algorithm is to maximize the R2 value on the cross-validation for each estimator, and to guarantee the maximum results, we set the fitness limit in 0.999, the number of population per generation in 6, and the execution time which differs according to the estimator and its complexity. The best scores are in **bold**, while the weakest are <u>underlined</u>.

Model	Duration	Num Generations generated	Max Score
Linear Regression	**2 hours**	≥ 10000	**0.99872**
SVR	**2 hours**	≥ 8000	0.99850
Random Forest	**2 hours**	≥ 120	<u>0.88915</u>
Neural Network	<u>5 hours</u>	<u>7</u>	0.98486

Table 3.8: General information about genetic algorithms results

Appendices section 4 contains the evolution of the R2 cross-validation through the generations.

3.7.2 Best Hyper-parameters:

Finally, after applying the optimization technique on each estimator, these are the best hyper-parameters found:

Model	Parameter name	Best value
Lasso	alpha	0.4321608040201005
	max_iter	1000
	positive	False
	precompute	False
	selection	cyclic
	tol	0.0001
Random Forest Regressor	n_estimators	220
	max_depth	11
	min_samples_split	2
	min_samples_leaf	1
	max_features	None
	bootstrap	True
	criterion	absolute_error
Neural Network	momentum	0.1
	batch_size	10
	learning_rate	0.01
	dropout	0.01
LinearSVR	epsilon	0
	tol	0.03526763381690846
	max_iter	1000
	C	1.250625312656328

Table 3.9: Best Hyperparamters

3.7.3 Scores:

R-SQAURE:

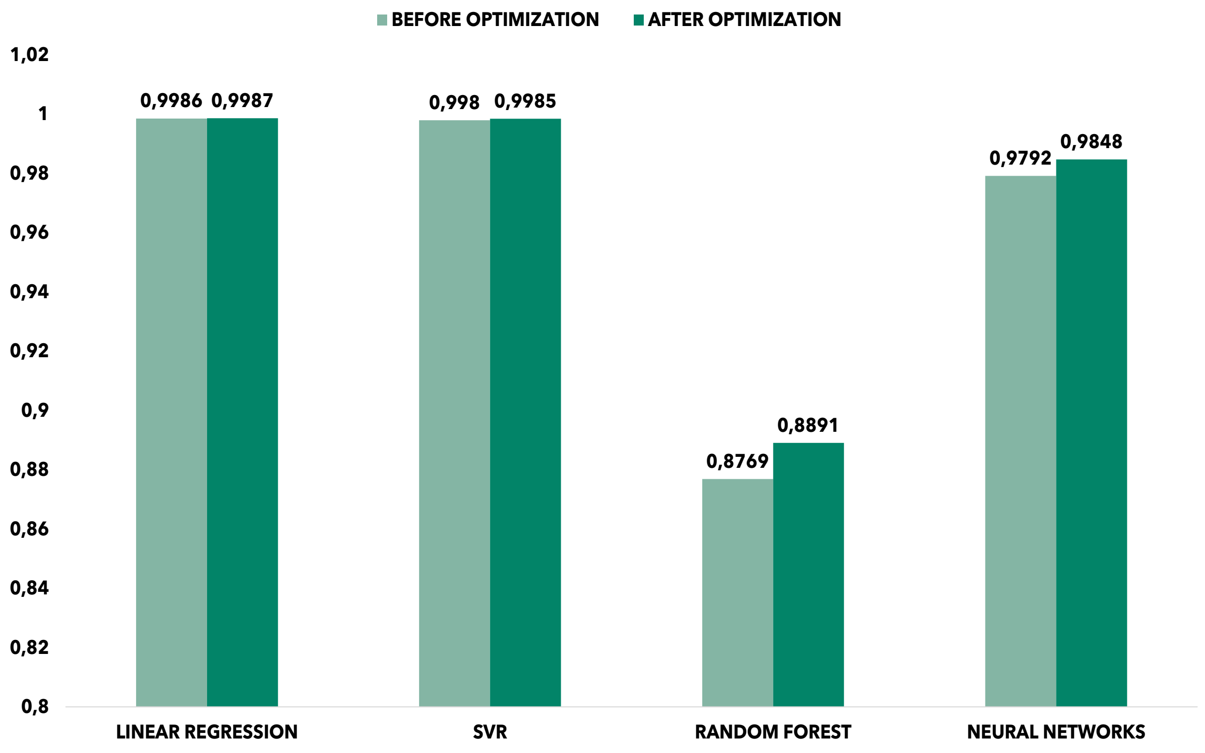


Figure 3.15: R-SQUARE 5K Cross-validation score (Train Set)

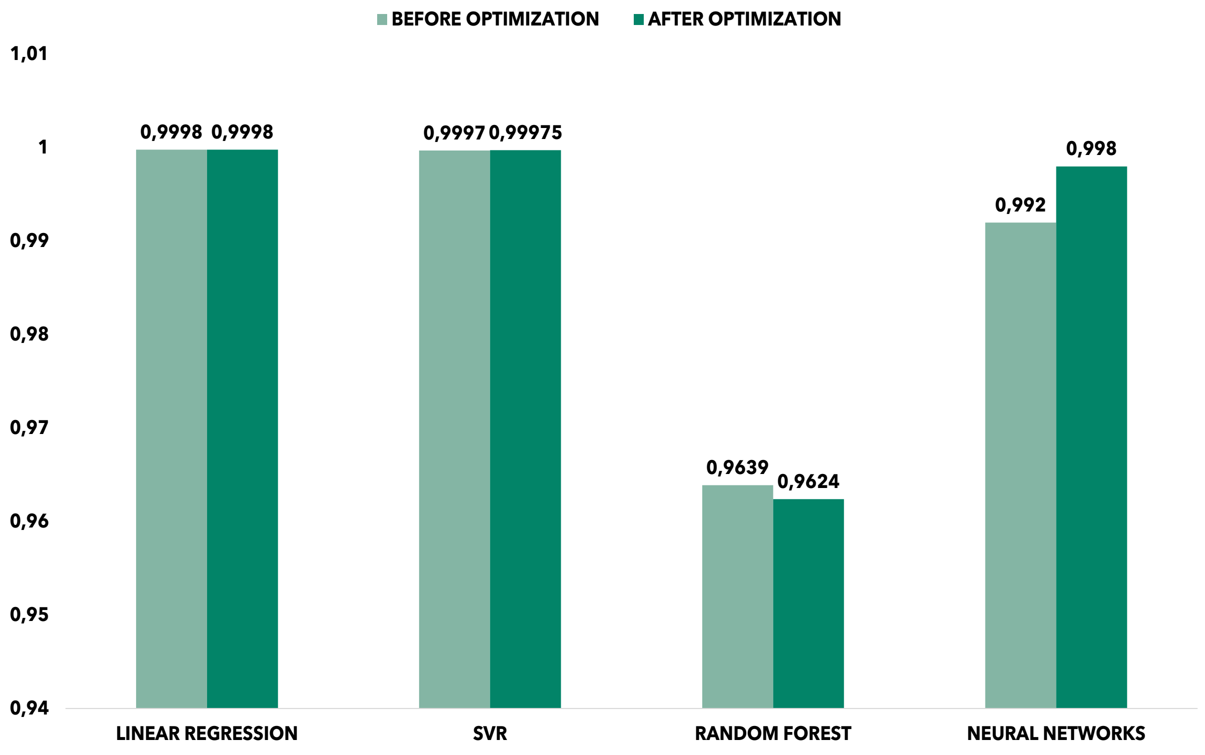


Figure 3.16: R-SQUARE Train Set Score

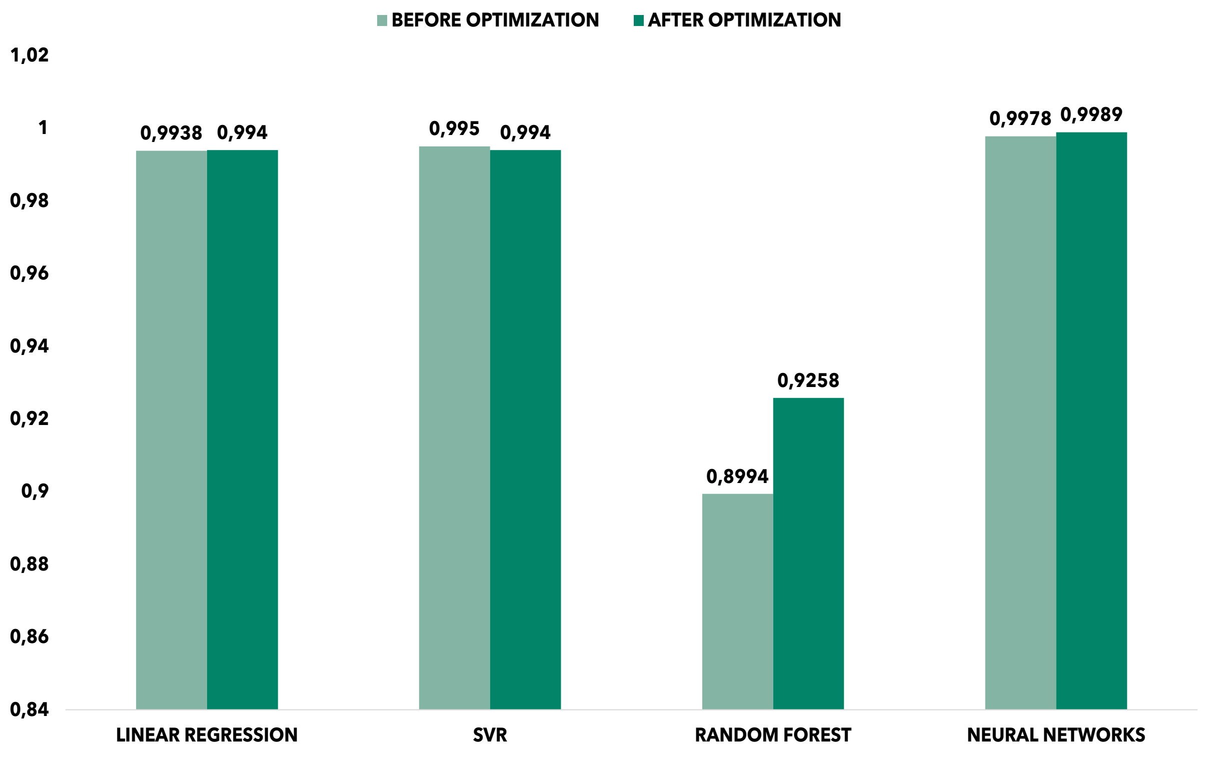


Figure 3.17: R-SQUARE Test Set Score

Mean Absolute Error:

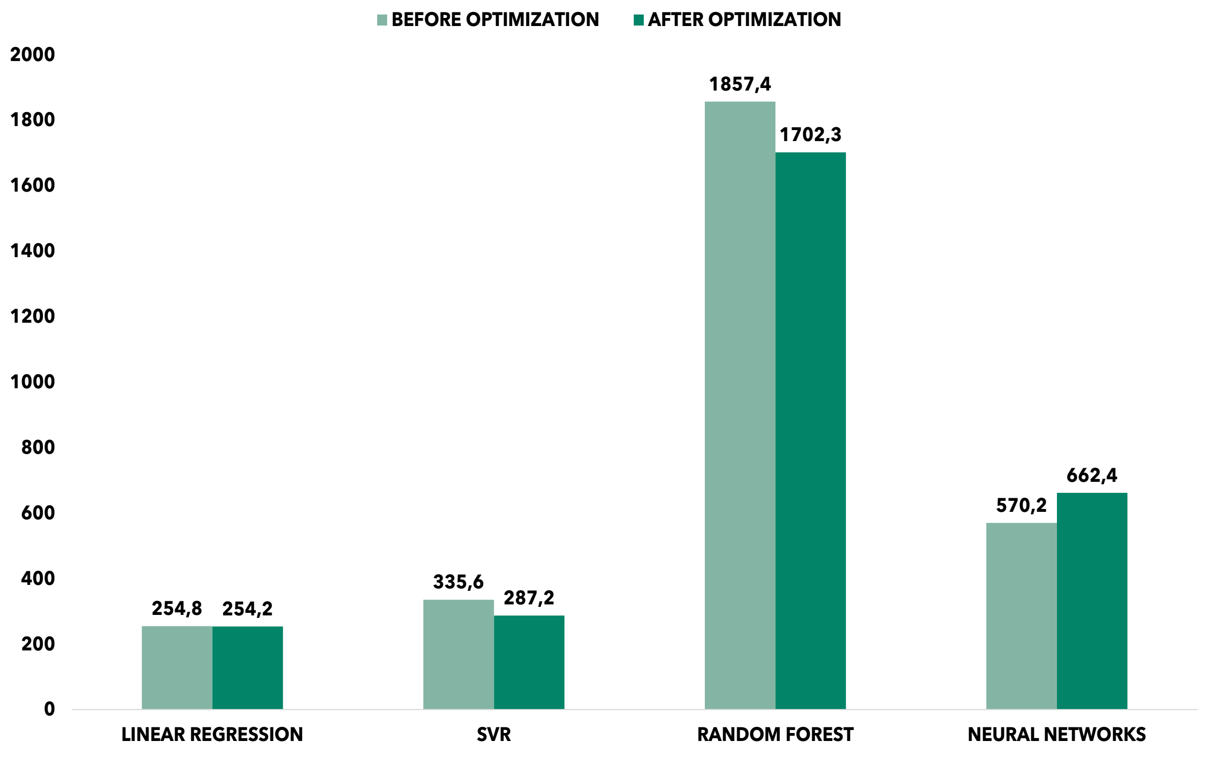


Figure 3.18: MAE 5K Cross-validation score (Train Set)

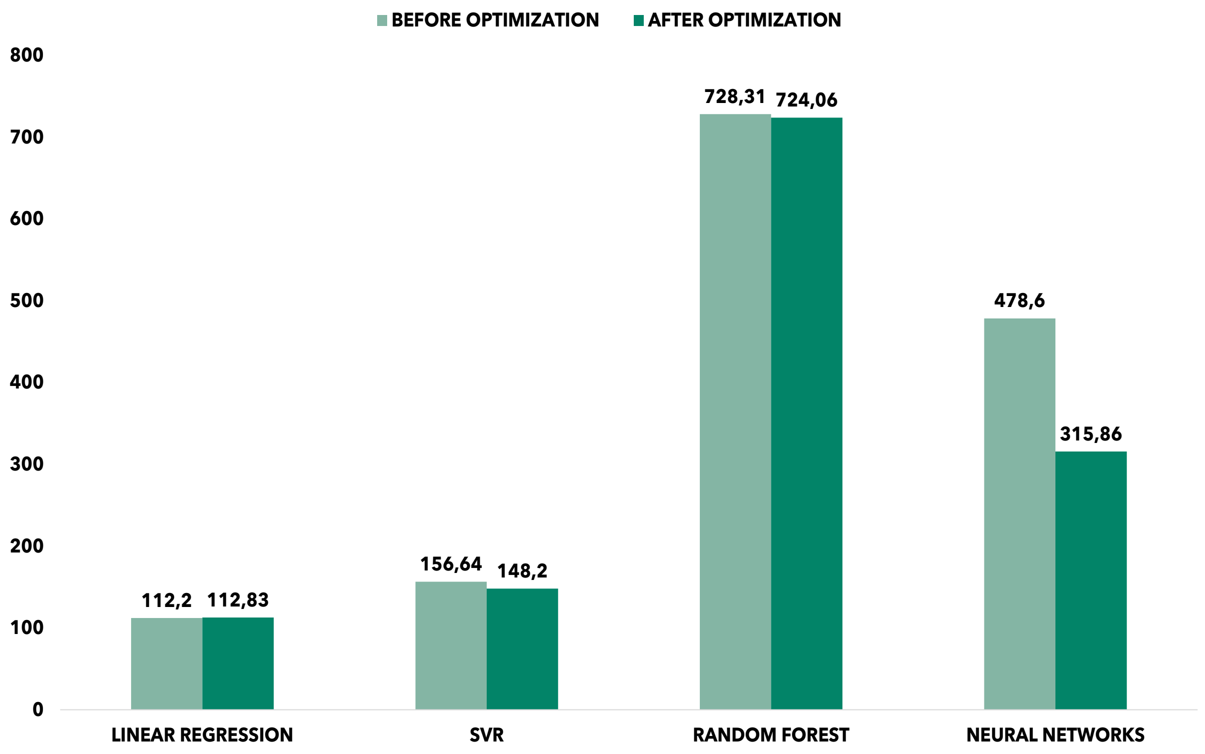


Figure 3.19: MAE Train Set Score

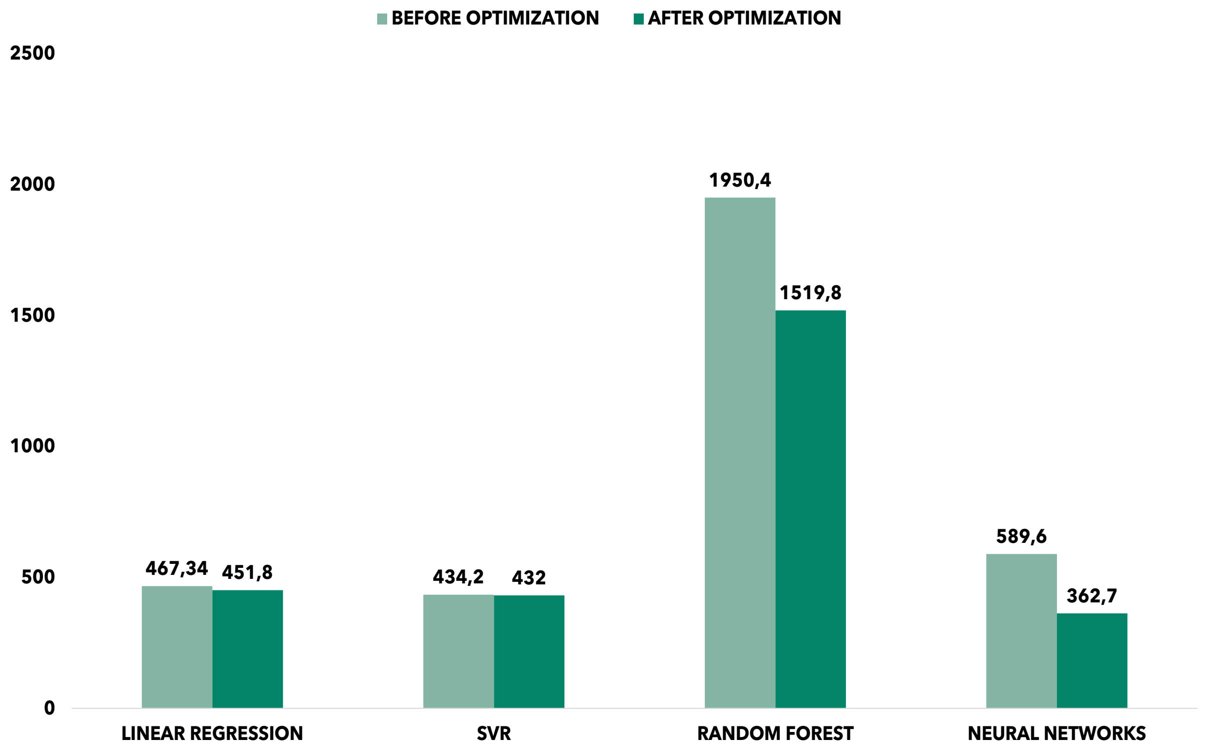


Figure 3.20: MAE Test Set Score

Mean Magnitude of Relative Error:

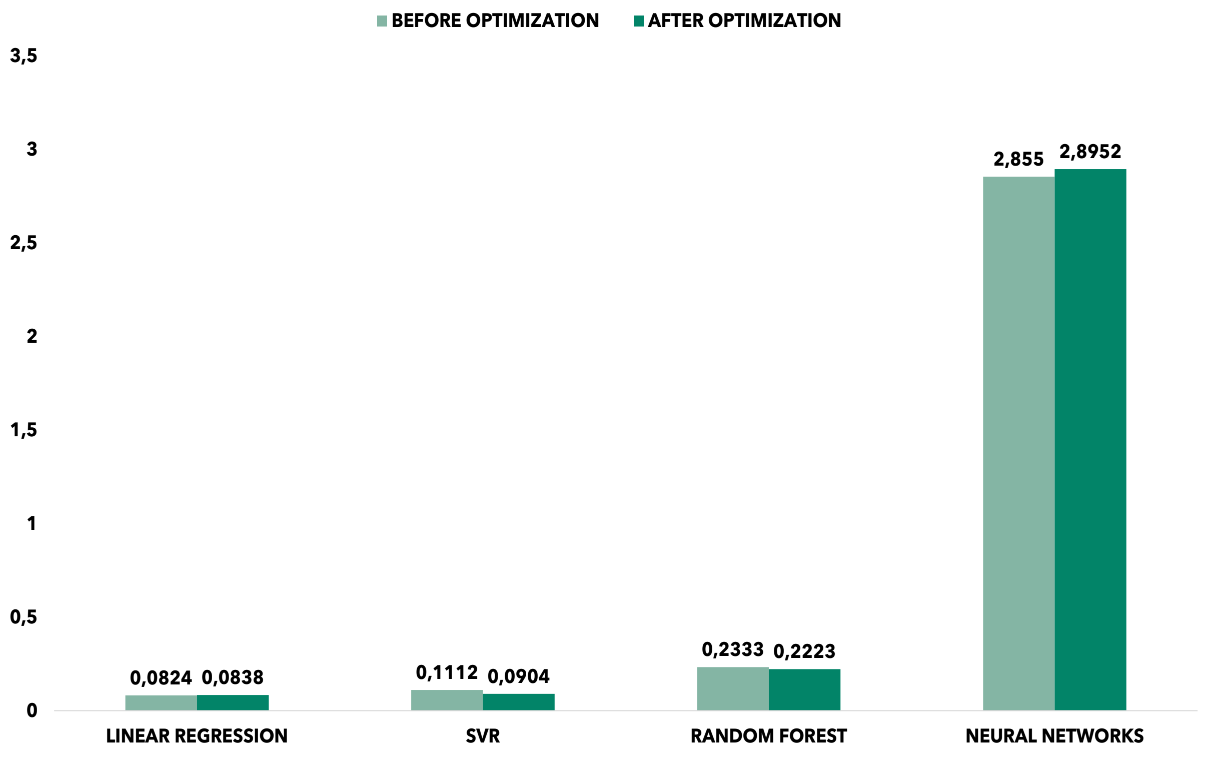


Figure 3.21: MMRE 5K Cross-validation score (Train Set)

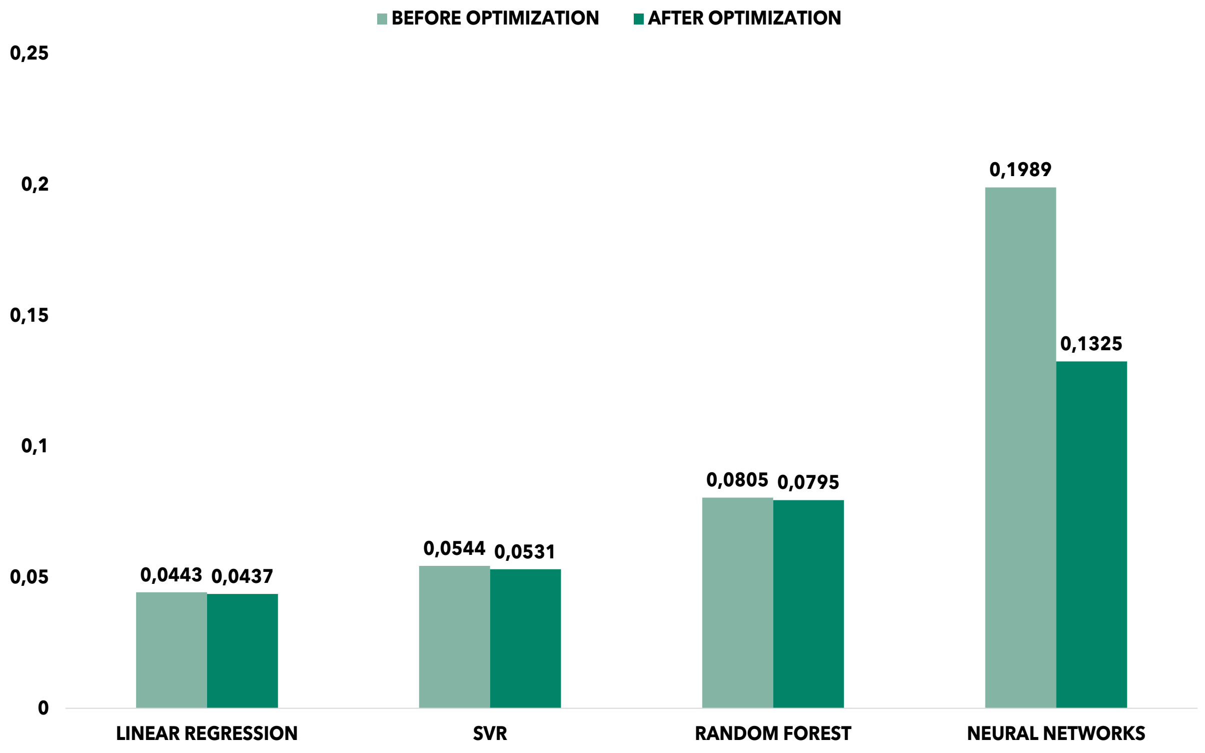


Figure 3.22: MMRE Train Set Score

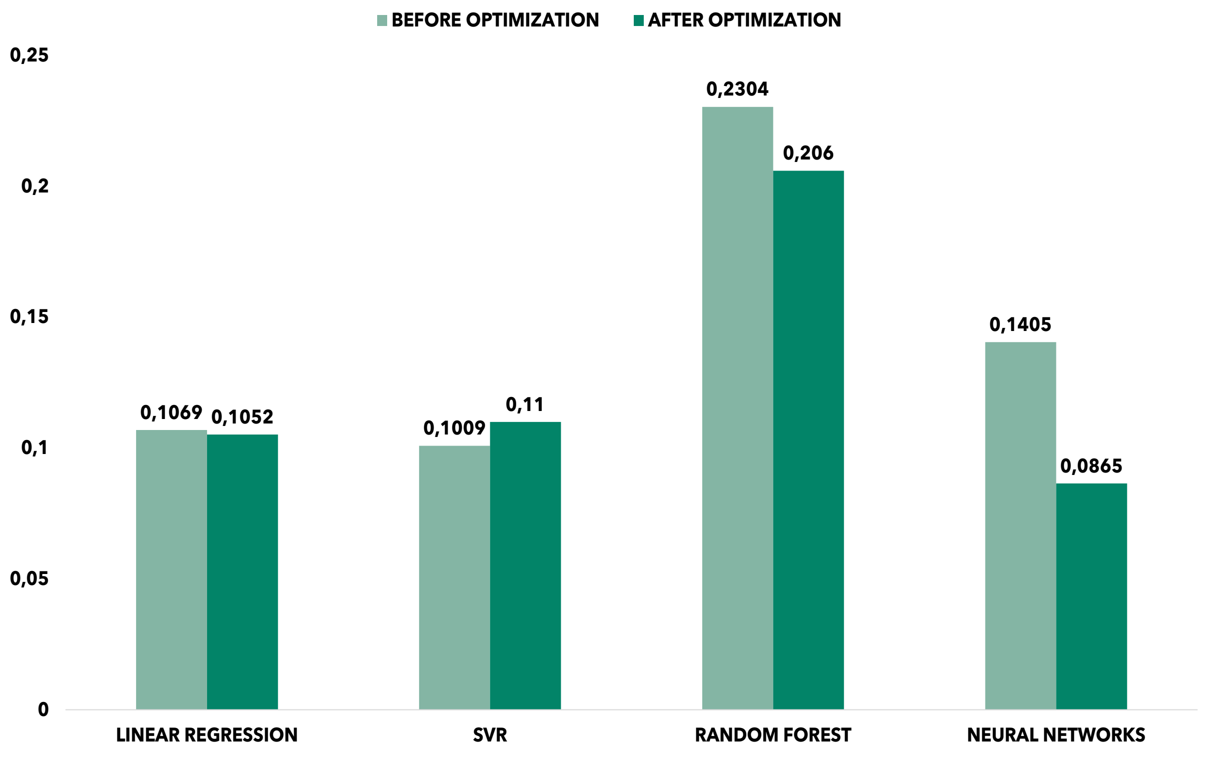


Figure 3.23: MMRE Test Set Score

PRED:

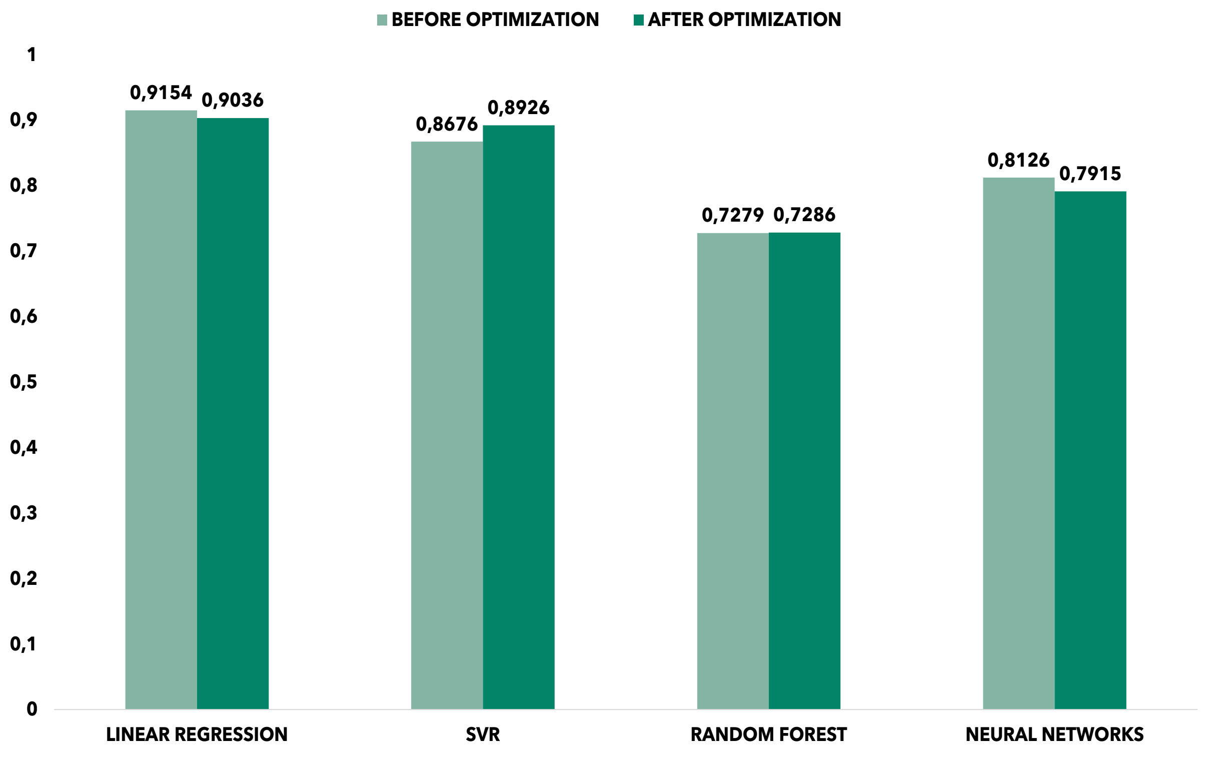


Figure 3.24: PRED 5K Cross-validation score (Train Set)

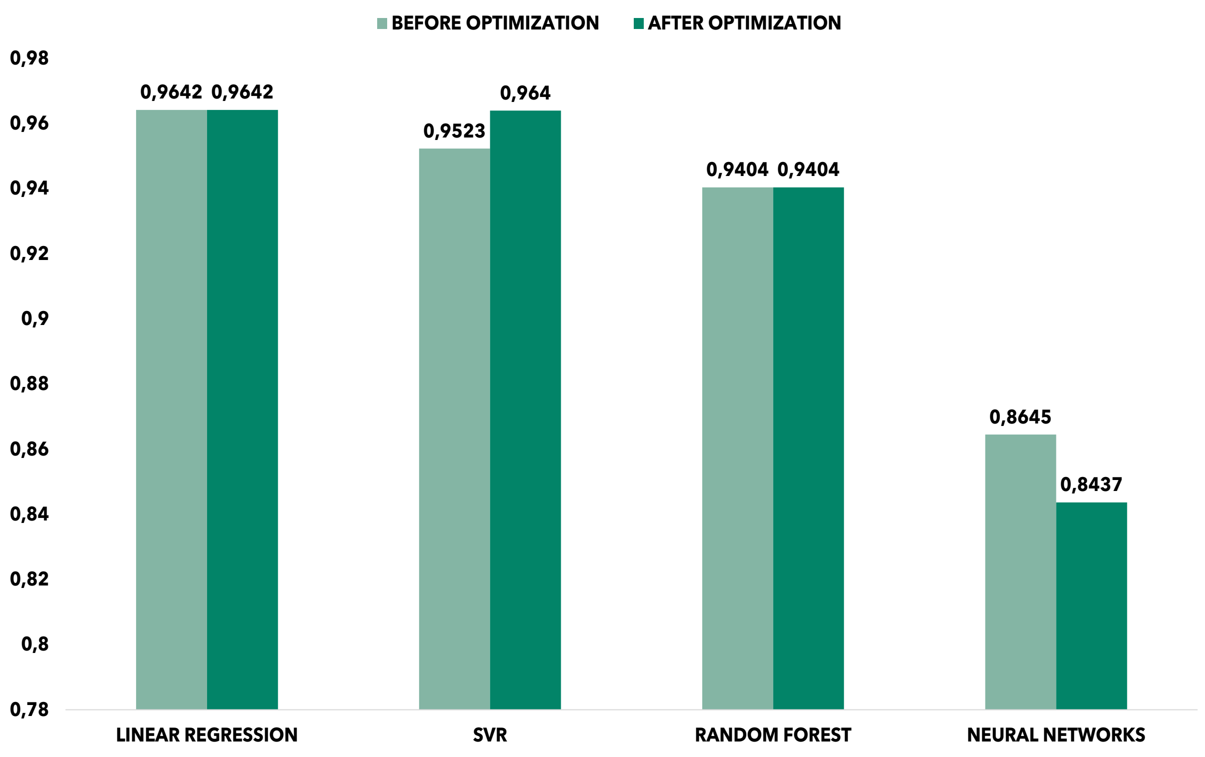


Figure 3.25: PRED Train Set Score

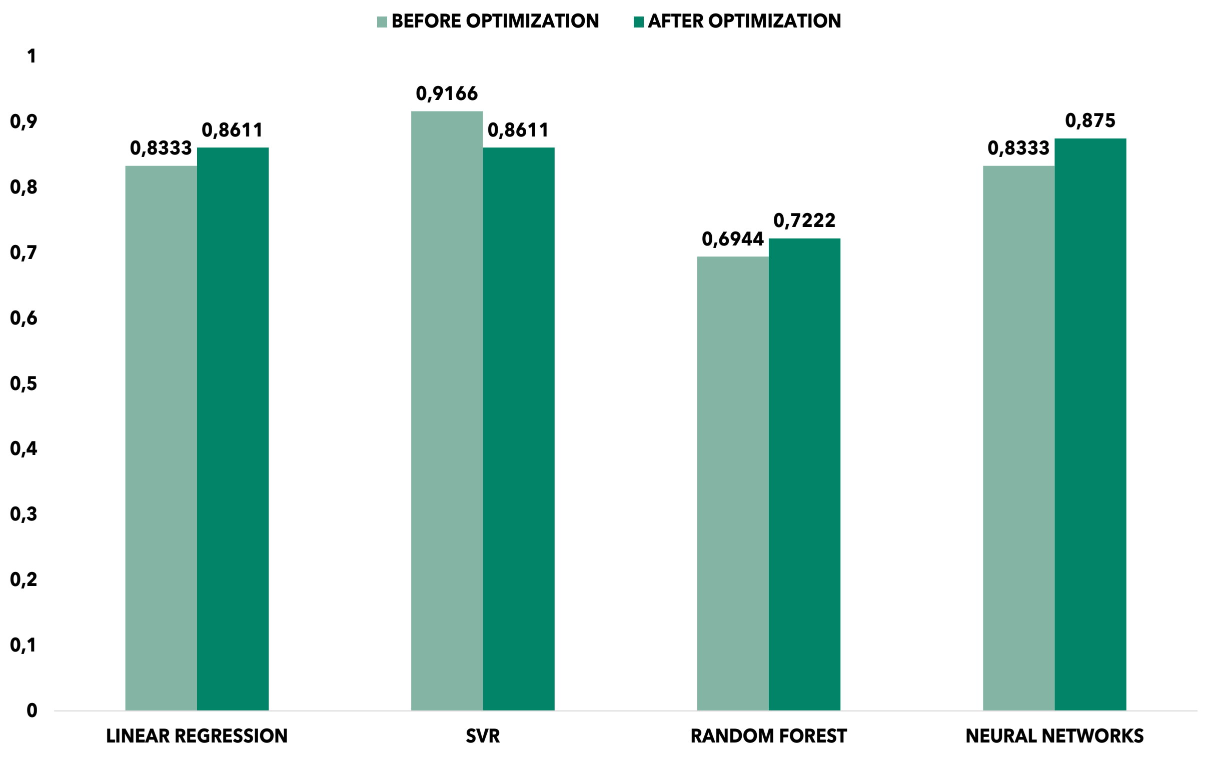


Figure 3.26: PRED Test Set Score

3.8 Discussion and Conclusion:

Generally, As we can see in **Section 3.7**, our models perform well, since they provide good results, especially the linear regression and the support vector regression which have about the same scores. Before optimizing the hyper-parameters, linear regression outperforms the other models in terms of 5K Fold cross-validation scores, either for the R2 or the other metrics. On the other hand, Random Forest has the weakest scores, the lowest R2 and PRED, and the highest MAE, while the highest MMRE is that of neural networks.

After validating our models with cross validation, we fitted our models again on the train set in order to see how they perform on the test set. We can see the results of each estimator on train and test sets, where linear regression has the highest scores for the train set, but the svr outperforms the linear regression on the test set. For neural networks and random forest, they shared the lowest scores.

By applying the optimization techniques for these models, we can see that there is an improvement on the results of each estimator in terms of cross validation. We notice that the value of R2 and PRED have increased, while the errors MAE and MMRE have decreased. In addition, linear regression with regularization had the highest scores on all metrics.

We fitted again our optimized models, to check if the test scores are improved. We notice first that the test scores are improved compared to our models before the optimization. First, the linear regression with regularization outperforms again the other models generally speaking, where we find that for almost all the metrics it has the most optimal values, except for the MAE and MMRE for the test set where the neural networks has the lowest value. In addition, linear regression and svr have the same scores for the PRED metric for both train and test sets.

Basically, our models scores are satisfying either before or after the optimization. Data prepossessing step was important, and especially the Sequential Feature Selection which allowed us to extract the minimum of important features. Linear regression was able to outperform the other models during all the stages, either before optimizing or after, but we can not deny that the other models perform well. In addition, regularization technique for linear regression helped to minimize the errors between the real values and the predictions. On the other hand, genetic algorithms succeeded in optimizing the performances of our models, where the R2 and PRED are increased and the errors are decreased in the cross-validation. **Appendices section 4** contains the evolution of the R2 cross-validation through the generations.

Finally, we conclude that linear regression model was able to outperform the other models, while the random forest results were not very good compared to the other models.

Chapter 4

CONCLUSION

This work represents the result of a training that lasted 2 years in the National Institute of Statistics and Applied Economics, where we applied all the techniques of machine learning, statistics and optimization that we have seen to be able to achieve this project. The objective of this thesis was to develop machine learning models for the Software Effort Estimation, by working on SEERA dataset. The project can be summarized in 2 phases:

– The first one consisted in carrying out a literature review, whose objective was to analyze all the works carried out within the framework of the Software Effort Estimation, by choosing a set of articles which corresponds to our orientation and which is well on the field of the Machine Learning. To do this, we have set a research approach on which we will base ourselves. The search process resulted in a selection of 19 articles, which we analyzed in detail to extract the maximum of statistical and machine learning techniques. We found, through this analysis, that the use of SVR, Linear regression, RF and Neural Networks was dominant in several works and that, moreover, they were ranked as the most profitable algorithms among the others. As a result, we decided to create models with these algorithms.

– The second phase was the realization of the project, where we chose SEERA as the dataset on which we will work. We performed the statistical tests to extract the important variables that are correlated to the target variable. Then we applied the polynomial transformation of degree 3 on these variables to apply SFS later in order to extract the final set of features that we will use to train the machine learning models. This process allowed us to obtain good results on all the models, and especially the linear regression which was the most accurate. Genetic algorithms has improved the performances of our models and showed their capability to tune the hyper-parameters of each estimator.

As result, we found that linear regression was the best performing model that was able to approximate the predictions to actual values.

Finally, this project is considered as a first version, and it would be interesting to improve this work by introducing other techniques to improve the predictions and approximate them as much as possible, as well as to optimize the whole process of data preprocessing and model optimization. The variable selection process was very time consuming, since the polynomial transformation caused the creation of several new variables, and as a result the sequential feature selection algorithm took a long time to find the final dataset. Genetic algorithms also took a long time to converge, especially for neural networks, since for 5 hours of execution the algorithm could generate only 7 generations, while for example for linear regression, 2 hours of execution generated more than 10000 generations. And so, we have to develop another optimization method that will converge quickly and that will find the best hyper-parameters for better scores. We will try to work on the factorial analysis of mixed data FAMD, which is a dimensionality reduction technique used for both categorical and numerical features, and which we used during early stages of this project but did not give good results. As for machine learning models, we will try another estimators that we found in the literature review, as well as ensemble techniques like boosting and stacking.

To close this work, I would like to thank everyone who has read this report.

Bibliography

[1] M. Korte and D. Port, "Confidence in software cost estimation results based on mmre and pred," in *Proceedings of the 4th international workshop on Predictor models in software engineering*, pp. 63–70, 2008.

[2] O. Hidmi and B. E. Sakar, "Software development effort estimation using ensemble machine learning," *Int. J. Comput. Commun. Instrum. Eng*, vol. 4, no. 1, pp. 143–147, 2017.

[3] A. Priya Varshini and K. Anitha Kumari, "Predictive analytics approaches for software effort estimation: A review," *Indian J Sci Technol*, vol. 13, no. 21, pp. 2094–2103, 2020.

[4] A. Singh and M. Kumar, "Comparative analysis on prediction of software effort estimation using machine learning techniques," in *Proceedings of the International Conference on Innovative Computing & Communications (ICICC)*, 2020.

[5] J. Shivhare and S. K. Rath, "Software effort estimation using machine learning techniques," in *Proceedings of the 7th India Software Engineering Conference*, pp. 1–6, 2014.

[6] A. L. Oliveira, P. L. Braga, R. M. Lima, and M. L. Cornélio, "Ga-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation," *information and Software Technology*, vol. 52, no. 11, pp. 1155–1166, 2010.

[7] M. Hammad and A. Alqaddoumi, "Features-level software effort estimation using machine learning algorithms," in *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pp. 1–3, IEEE, 2018.

[8] B. Baskeles, B. Turhan, and A. Bener, "Software effort estimation using machine learning methods," in *2007 22nd international symposium on computer and information sciences*, pp. 1–6, IEEE, 2007.

[9] Y. Kultur, B. Turhan, and A. B. Bener, "Enna: software effort estimation using ensemble of neural networks with associative memory," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pp. 330–338, 2008.

[10] A. BaniMustafa, "Predicting software effort estimation using machine learning techniques," in *2018 8th International Conference on Computer Science and Information Technology (CSIT)*, pp. 249–256, IEEE, 2018.

[11] A. P. Varshini, K. A. Kumari, D. Janani, and S. Soundariya, "Comparative analysis of machine learning and deep learning algorithms for software effort estimation," in *Journal of Physics: Conference Series*, vol. 1767, p. 012019, IOP Publishing, 2021.

[12] M. M. Al Asheeri and M. Hammad, "Machine learning models for software cost estimation," in *2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pp. 1–6, IEEE, 2019.

[13] K. Iwata, T. Nakashima, Y. Anan, and N. Ishii, "Machine learning classification to effort estimation for embedded software development projects," in *Research Anthology on Agile Software, Software Development, and Testing*, pp. 1652–1665, IGI Global, 2022.

[14] P. L. Braga, A. L. Oliveira, and S. R. Meira, "Software effort estimation using machine learning techniques with robust confidence intervals," in *7th international conference on hybrid intelligent systems (HIS 2007)*, pp. 352–357, IEEE, 2007.

[15] A. L. Oliveira, "Estimation of software project effort with support vector regression," *Neurocomputing*, vol. 69, no. 13-15, pp. 1749–1753, 2006.

[16] P. Rijwani and S. Jain, "Enhanced software effort estimation using multi layered feed forward artificial neural network technique," *Procedia Computer Science*, vol. 89, pp. 307–312, 2016.

[17] L. L. Minku and X. Yao, "A principled evaluation of ensembles of learning machines for software effort estimation," in *Proceedings of the 7th international conference on predictive models in software engineering*, pp. 1–10, 2011.

[18] H. Mustapha, N. Abdelwahed, *et al.*, "Investigating the use of random forest in software effort estimation," *Procedia computer science*, vol. 148, pp. 343–352, 2019.

[19] S. K. Palaniswamy and R. Venkatesan, "Hyperparameters tuning of ensemble model for software effort estimation," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 6, pp. 6579–6589, 2021.

[20] R. Marco, S. S. S. Ahmad, and S. Ahmad, "Bayesian hyperparameter optimization and ensemble learning for machine learning models on software effort estimation," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 3, 2022.

[21] E. I. Mustafa and R. Osman, "Seera: a software cost estimation dataset for constrained environments," in *Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering*, pp. 61–70, 2020.

[22] J. Keung, E. Kocaguneli, and T. Menzies, "Finding conclusion stability for selecting the best effort predictor in software effort estimation," *Automated Software Engineering*, vol. 20, no. 4, pp. 543–567, 2013.

[23] N. El Aboudi and L. Benhlima, "Review on wrapper feature selection approaches," in *2016 International Conference on Engineering & MIS (ICEMIS)*, pp. 1–5, IEEE, 2016.

[24] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

Appendices

This section contains the figures that we have not inserted before. It addresses the evolution of the score optimization of algorithms used by genetic algorithms, as well as the history plots of neural networks before and after optimization.

This figure shows the increase in score for the **Random Forest** by applying the genetic algorithms, which generate more than 120 generations.

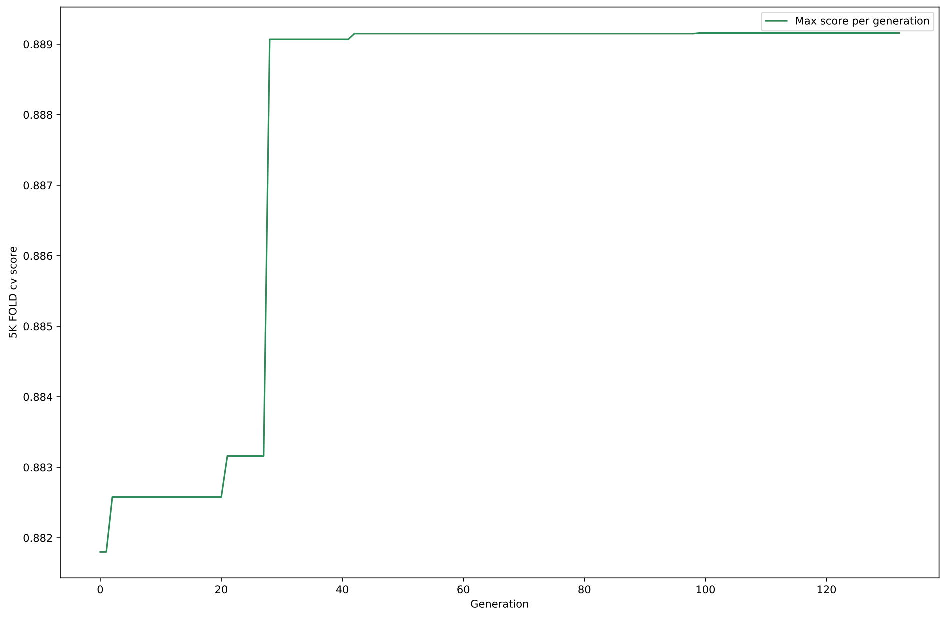


Figure 1: Random Forest Scores through generations (Genetic algorithm)

This figure shows the increase in score for the **Support Vector Regression** by applying the genetic algorithms, which generate more than 8000 generations.

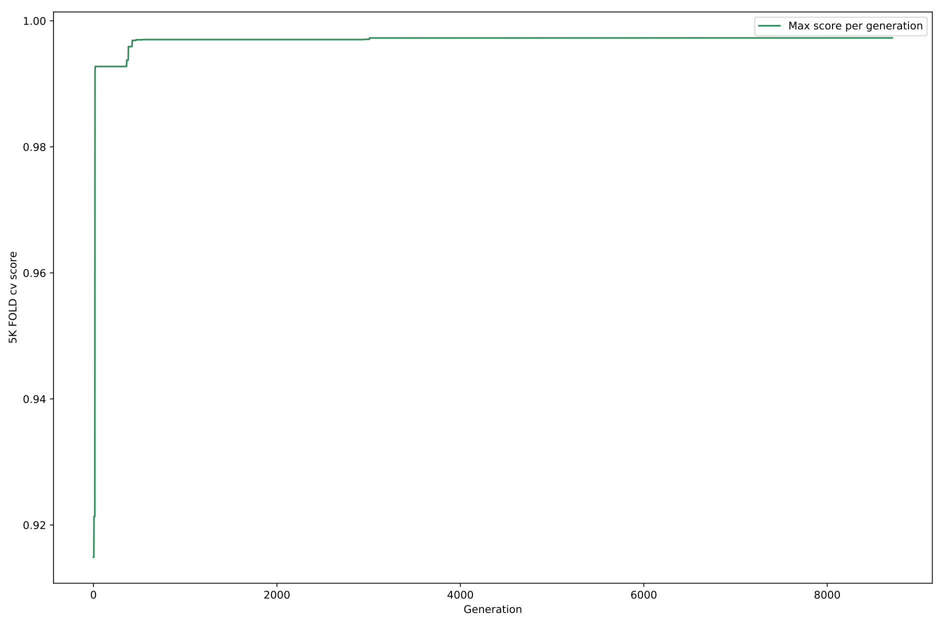


Figure 2: Support Vector Regressor Scores through generations (Genetic algorithm)

This figure shows the increase in score for the **Linear Regression with regularization** by applying the genetic algorithms, which generate more than 10000 generations.

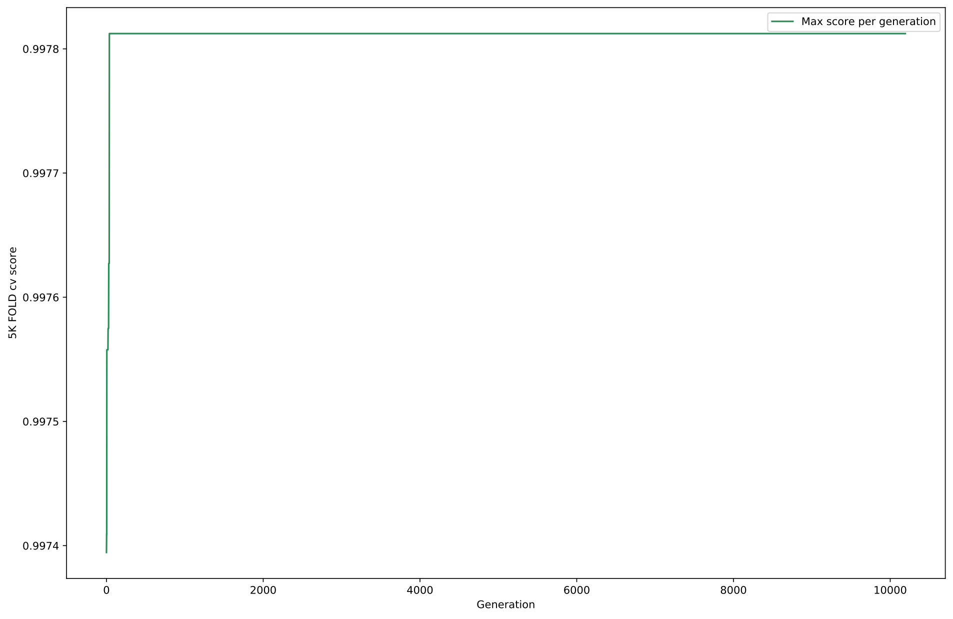


Figure 3: Linear Regression with regularization Scores through generations (Genetic algorithm)

This figure shows the increase in score for the **Neural Networks** by applying the genetic algorithms, which generate 7 generations.

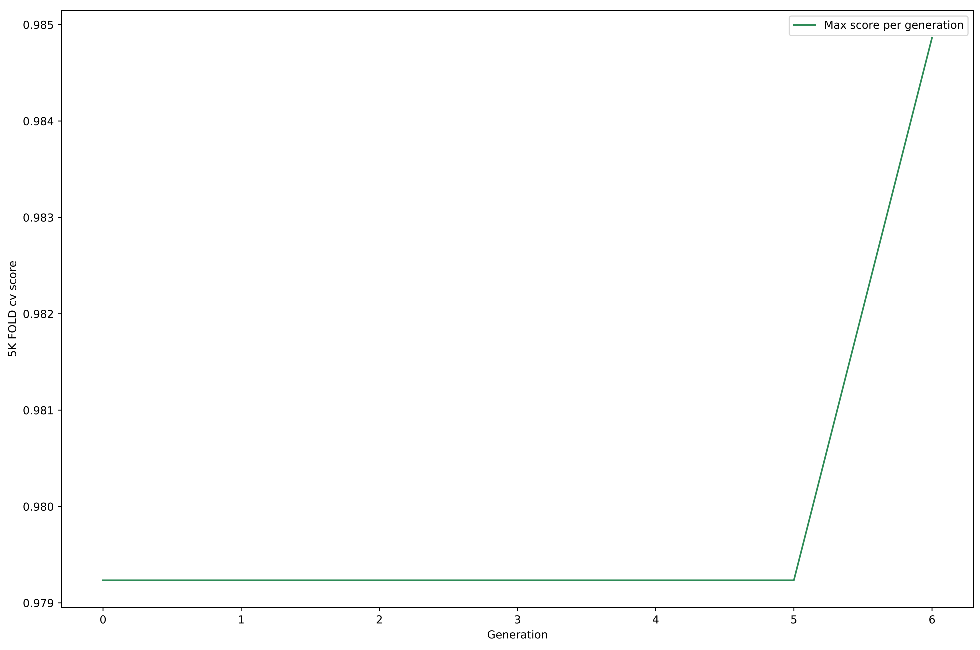


Figure 4: Neural Networks Scores through generations (Genetic algorithm)

Finally, the history plots of train and validation sets for neural networks before and after optimization, which was fitted on 3000 epochs.

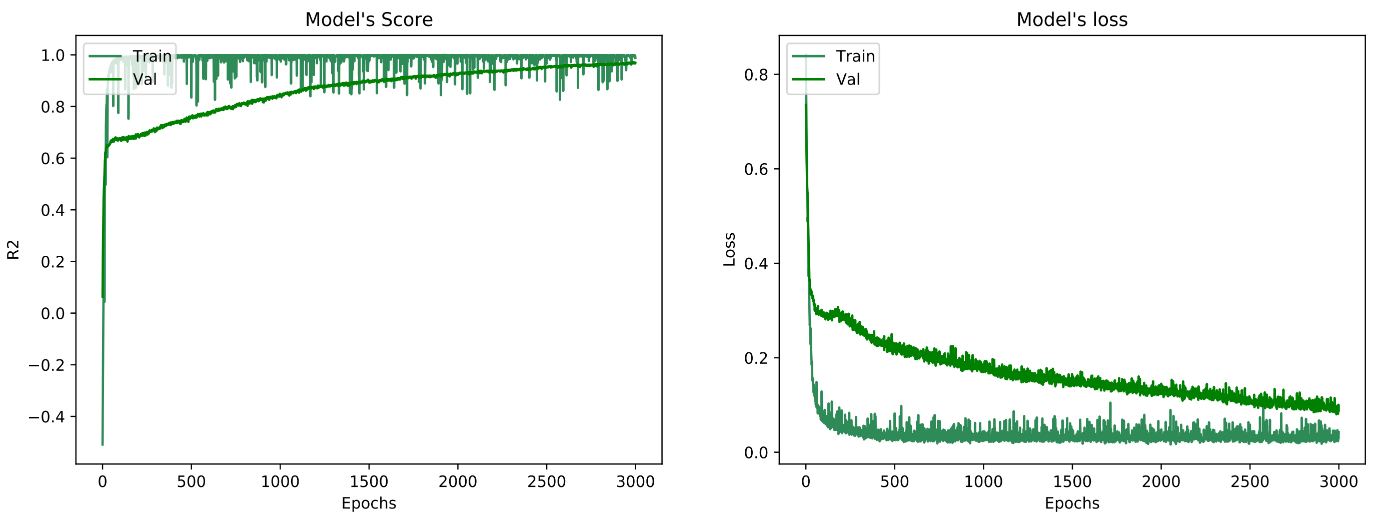


Figure 5: Neural networks history before optimization

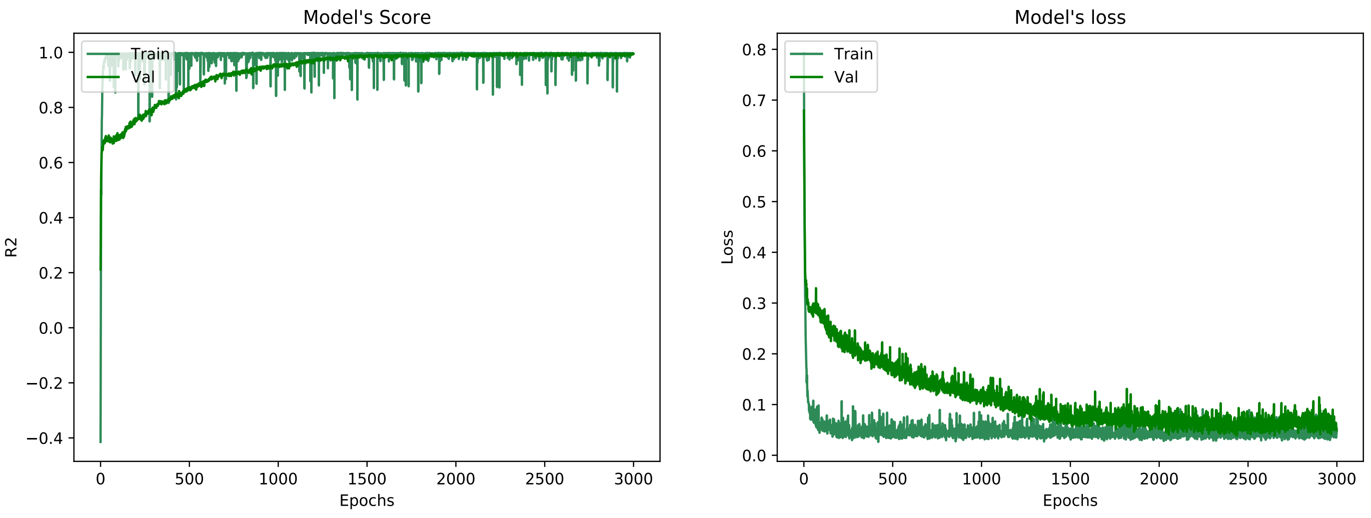


Figure 6: Neural networks history after optimization