Optimization of Random Forest Based Methods Applying the Genetic Algorithms

Zahra Aback

A Thesis

In

The Department

of

Mathematics and Statistics

Presented in Partial Fulfilment of the Requirements

for the degree of Master of Science (Mathematics) at

Concordia University

Montreal, Quebec, Canada

July, 2018

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Zahra Aback**

Entitled: **Optimization of Random Forest Based Models Applying Genetic Algorithms**

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Mathematics)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Examiner

Dr. Yogendra P. Chaubey

_____ Examiner

Dr. Frédéric Godin

_____ Thesis Supervisor

Dr. Arusharka Sen

Approved by _____

Chair of Department or Graduate Program Director

_____

Dean of Faculty

Date _____

ABSTRACT

Optimization of Random Forest Based Models Applying Genetic Algorithms

Zahra Aback

In this century access to large and complex datasets is much easier. These datasets are large in dimension and volume, and researchers are interested in methods that are able to handle this type of data and at the same time produce accurate results. Machine learning methods are particularly efficient for this type of data, where the emphasis is on data analysis, and not on fitting a statistical model. A very popular method from this group is Random Forests which have been applied in different areas of study on two types of problems: classification and regression. The former is more popular, while the latter can be applied for data analysis. Moreover, many efficient techniques for missing value imputation were added to Random Forest over time. One of these methods which can handle all types of variables is MissForest. There are several studies that applied different approaches to improve the performance of classification type of Random Forests, but there are not many studies available for regression type. In the present study, it is evaluated if the performance of regression type of Random Forests and MissForests could be improved by applying Genetic Algorithms as an optimization method. The experiments were conducted on five datasets to minimize the mean square error (MSE) of the Random Forest and imputation errors of the MissForest. The results showed the superiority of the proposed method in comparison to the classical Random Forest methods.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my adviser Dr. Arusharka Sen for his guidance and all his support for my Master's studies.

Besides my adviser, I would like to thank my examiners: Dr. Yogendra P. Chaubey and Dr. Frédéric Godin for their precious comments which helped me to improve my research.

I would also like to express my gratitude to the faculty members and graduate students who helped me through my studies at Concordia University.

Last but not the least, I would like to thank my family especially my mom and friends for supporting me throughout the writing of this thesis and my life in general.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the era where we have access to the large volume of data, researchers are looking for methods or algorithms that are flexible enough to handle this type of data for analysis and at the same time produce efficient statistics (Biau & Scornet, 2016). These methods are recently developed in statistics and computer science in parallel which are linked to an area of study called machine learning. This field of study enables computers to learn without being programmed explicitly (Samuel, 1959). Many methods such as classification and regression trees, bagging, and an improved version of them, Random Forests (RF), fall in this category (James et al., 2013).

RF which was introduced by Breiman et al. (2001) has shown a great success in recent decades to handle large datasets. RF has become very popular in recent years since it is applicable in many prediction problems, and much simpler than some other machine learning methods to run due to small number of parameters for tuning. It is also recognized as a method that can be executed in parallel (Biau & Scornet, 2016). RF is a combination of lots of trees that are grown from different pieces of datasets and provides an estimate by averaging over trees (Biau & Scornet, 2016).

Since most of the time it is very common to have datasets that include missing values, we have to decide how to deal with them. This is a very important issue when RF is applied for data analysis since it cannot perform and produce results when missing values are present. In this case, missing values are either discarded if applicable or a decision is made to impute them. There are many imputation methods, but a very few of them

can handle missing values for both categorical and continuous variables. In this regard, we used MissForest which was introduced by Stekhoven and Bühlmann (2011). Using this method, missing values were imputed and imputation errors were reported for both continuous and categorical variables.

There are two types of Random Forests: classification and regression for categorical and continuous responses, respectively. The regression type of RF produces a predicted value which is the average of the results of all the trees, and its performance is measured by mean square error (MSE) of that estimation.

There are lots of studies focusing on improving the performance of RF or studying the options for its parameter settings that help produce results in a reasonable time frame that are accurate enough. Oshiro et al. (2012) and Latinne et al. (2001) studied the number of trees required in a Random Forest and the first study suggested 64 trees, and the second one suggested limited numbers for different datasets. Their study showed that the number of trees larger than what they suggest will not improve RF performance. Bader-El-Den and Gaber (2012) and Elyan and Gaber (2017) also studied the impact of using Genetic Algorithms (GA) on the improvement of the performance of RF. These studies focused on classification type of RF, the response of which is categorical, and they suggest different approaches.

In this study the focus is on improving the performance of regression RF by applying Genetic Algorithms. In fact, GA is applied to minimize the mean square error of RF by searching over number of trees and number of features at each split. Moreover, it has been studied if GA has an impact on the performance of MissForest to minimize its imputation errors.

1 Random Forest

The RF was devised by Breiman et al. (2001) since he wanted to improve the performance of Bagging which is the combination of large number of trees that are produced from bootstrap samples of the original dataset. The improvement in comparison to Bagging occurred to produce a RF by adding another type of randomness in the model beside

bootstrap sampling. The other randomness is CART-split, which is choosing the best variable for splitting. The variable is selected from a number of variables in the subset M_{try} of total number of variables p to divide the sample space to sub-samples.

The RF is an ensemble method too, which is a method that produces the prediction model (RF) from the strength of the collection of simpler ones (Decision Trees). In fact, ensemble learning includes two procedures. It develops a large number of individual models and then it produces the final prediction by combining them (Friedman, Hastie, & Tibshirani, 2001).

The RF is very efficient when the number of variables is larger than the number of observations, and it is called "off the shelf" method which means it can give a very preliminary result about the dataset in a very short time (Cutler, Cutler, & Stevens, 2012). It does not need any formal assumption about the distribution of the data, and produces proximity matrices which can be used for missing values imputation and for the detection of outliers. It is robust to the outliers. The performance of a RF is measured internally during the implementation of its function, and it ranks the variables by their importance (Cutler et al., 2012). There are many softwares that can implement the RF analysis in their environment such as R. There is a package in R called Random Forest which can be installed to carry out its function and produce the results. In addition to these cases, there are some other softwares that have been developed just for machine learning methods such as Salford Systems Data Mining and Predictive Analytics Software (SPM) and Waikato Environment for Knowledge Analysis (WEKA).

2 Genetic Algorithms

There are many situations in our lives where we are looking for optimization of our performance. For instance, we would like to have maximum sleep but at the same time arrive at work on time. We could think of which route to take or how to be well prepared the night before for the next morning. In fact, we look for a way to minimize our cost and maximize our output or result. Optimization is a process in which the inputs of a mathematical function or experiment or characteristics of a device are adjusted to find

the maximum or minimum results, see Haupt and Haupt (2004).

Haupt and Haupt (2004) addressed some issues in optimization problems such as finding solutions in the case where the derivative of the objective function (the function that needs to be optimized) does not exist or difficult to derive. Also, the type of solution to the optimization problem is crucial too. It is important to know if the optimum value obtained by this process is local or global.

There are several approaches for optimization with different advantages and disadvantages. Haupt and Haupt (2004) discussed many algorithms that help to find the maximum or minimum of some functions that were not either linear nor having a derivatives (it is hard to find their derivative). However, they had one thing in common: they could not produce the global minimum or maximum. Therefore, later some other algorithms were introduced to overcome this problem, among which GA proposed. The GA does not need derivative of the functions and could produce the global optimized solutions to the problems.

John Holland was the first person who tried to develop a theoretical basis for GA. He developed it between 1960s and 1970s and the work was summarized in his book "Adaptation in Natural and Artificial Systems (1975)". However, this method was made very popular through one of his students, David Goldberg (1989), whose dissertation solved a problem which was the control of gas pipline transmission (Haupt & Haupt, 2004).

In their book (Haupt & Haupt, 2004) defined GA as an optimization procedure that imitates the reproduction process in organisms and uses its natural selection method to find the optimum solution. In this process a large population is evolved through a specific selection method to a point that optimizes the fitness function. GA mostly has been used for optimizing hard and complex problems.

3 Missing values

Most of the time datasets are not complete and they contain missing values. There are three most important type of missing values: Completely Missing at Random, Missing at

Random, and Not missing at Random which are explained as follows:

- Missing Completely At Random (MCAR): If the probability for a variable to have some missing values is independent of other variables, that is called MCAR. In fact, all the observations for a random variable have the same chance of being missing. In this case, missing values can be removed from the dataset without imposing a bias to the estimation, but this type of missing is not very common (Tang, 2017).

- Missing At Random (MAR): If the probability of being missing for a random variable depends on the information of the other variables, it is called Missing At Random. For instance, the information for occupation is missing for some races or ethnic groups (Young, 2017).

- Missing Not At Random (MNAR): If the probability of being missing for a variable is because of that variable itself, it is called MNAR. In this case the respondent is not willing to answer the question because they are embarrassed or they do not want to share their information for that question. Most of the time it is difficult to identify this type of missing since it depends on some unobserved values (Young, 2017).

There are different strategies to deal with missing values. The first one is to remove all the missing values and work with the complete dataset. In this case some information will be lost and we will have larger standard errors. Also, it causes a problem when there is a large number of variables with lots of missing values. In this case discarding these observations will lead to dealing with many variables and few observations. The other approach is to keep missing values and impute them by some current methods, so those parts of the data will not be lost but adjusted (Young, 2017).

There are different methods for imputation, and some of them are RF based methods. The two methods which can be implemented through the RF package in R are rough imputation and RF imputation. The first model replaces missing values by the median or the mode and produces the imputed values. The second model uses proximity measures in a RF to impute the missing values, but this method cannot be applied when missing

values are present for the response variable.

The third RF based method is MissForest, some characteristics of which was highlighted by Stekhoven and Bühlmann (2011). Firstly, this method can handle both categorical and continuous variables simultaneously. Secondly, the imputation error can be computed internally during the imputation process. Finally, it performs really well when non-linear relations or complex interactions are present. A comparison among these three methods shows that the later can produce better results. Therefore, in this study, missing values are imputed by MissForest, and it was also studied if GA can optimize its performance by minimizing imputation error (Stekhoven & Bühlmann, 2011).

4 Limitation

There are two types of RFs, a classification RF and the regression one, and most of the time the focus in studies was always on the classification type. Some of these studies focused on two parameters in RF that are considered as the most important factors which have an effect on the its performance: the number of trees and the number of features considered at each split (Elyan & Gaber, 2017). There are some studies that looked for reducing the number of trees to produce accurate results but at the same time reduce the processing time (Oshiro et al., 2012). Some other studies tried to find the best values for the number of features considered at each split that could produce better results than default values of a given RF implementation (Bernard et al., 2008). Also, two studies focused on finding the best values for both parameters mentioned above to minimize the model error by applying an optimization method, GA (Bader-El-Den & Gaber, 2012), Elyan and Gaber (2017).

In spite of the fact that regression RF follows the same routine as the classification type, according to my knowledge, there are not many studies available on evaluating the performance of a regression RF. Therefore, the objective of the current study is to evaluate if the mean square error of the regression RF could be minimized by searching over the optimal numbers of trees and number of features at each split. GA was applied as an optimization method to produce the minimum error and report its corresponding

solutions. This method was selected since it could optimize complex functions, it could produce the global optimal solution, and it could be run in parallel. Moreover, since MissForest is also an RF based model for the imputation, the other objective is to study if its performance could be improved (minimizing the imputation error) by finding the optimal solutions for the number of trees and the number of features at each split.

Chapter 2

Random Forest

The early definition of a Random Forest that is given by Breiman (2001) was for a classification RF and was expressed as follows:

A Random Forest is a classifier that is created by a group of trees $\{h(x, \Theta_k)\ k = 1, 2, \cdots\}$ where the $\{\Theta_k\}$ are identically independent random vectors, and each tree votes for the most popular class for an input x. The random vector Θ_k is not defined in this work, but implicitly it represent two types of randomness in an RF including the Bagging and splitting a node which will be explained later.

A regression Random Forest is similar to a classification one, except that for an input x, each tree produces an average as the estimation. The process of producing the estimation for a regression RF will be explained in details later. The most important factors in building a RF are CART-split of a Decision Tree and Bagging which creates predictors from bootstrap samples of the original dataset and produces the final estimation by averaging them (Biau & Scornet, 2016).

1 Decision Tree

The most popular Decision Tree is a Classification and Regression Tree (CART) that was introduced in 1984 by Leo Breiman, Jerome Friedman, Richard Olshen and Charles Stone. In this method the response could be continuous or discrete that leads to a regression or classification tree respectively.

1.1 CART-Split

In tree-based methods, the sample space is divided into a set of sub-spaces which are called regions or nodes and each of them creates an estimate (Friedman et al., 2001). The process of dividing the whole sample space into regions is called CART-split which is a procedure that a variable and best value of that is selected from a whole set of variables to minimize the mean square error of a region (Biau & Scornet, 2016).

CART-split of a Decision Tree is a binary partitioning of a sample dataset of size n, $\mathbb{D}_n$, and in each step it splits each region into two regions on the left and right by minimizing the mean square error of those regions for a regression tree. For instance for a (subset) region A suppose the number of observation is $N_n(A)$ and the pair of (j, z) as a cut point in which the splitting occurs. The component j of the pair refers to the co-ordinate variable X^j so that, this variable is selected from a whole set of independent variables $(X^1, X^2, \cdots, X^p)$ where p is the number of predictors. The second component z is a value from the domain of variable X^j where the splitting occures. This region will be divided into two more regions based on the best pair from whole predictors and their best splitting values. If $\mathbb{C}_A$ denotes a set of all pairs of cuts, the regression CART-split criterion that leads to growing a tree from top to down for each pair $(j, z)\,\epsilon\,\mathbb{C}_A$ by considering $X_i = (X_i^1, X_i^2, \cdots, X_i^p)$ where $i = 1, 2, \cdots, n$ is in the from in Equation (1.1): (Biau & Scornet, 2016):

$$L_{reg,n}(j,z) = \frac{1}{N_n(A)} \sum_{i=1}^{n} (Y_i - \bar{Y})^2 I_{X_i \epsilon A} - \frac{1}{N_n(A)} \sum_{i=1}^{n} (Y_i - \bar{Y}_{A_L} I_{X_i^j < z} - \bar{Y}_{A_R} I_{X_i^j \geq z})^2 I_{X_i \epsilon A} \quad (1.1)$$

The variable Y_i is the response variable correspond to all the X_i in each region and $\bar{Y}$ is the average of all the Y_i in that region, which is obtained by minimizing the mean square error of that region. The Equation (1.1) above has to be maximized to find the best variable and best value from its domain as the cut.

Region A is divided into two regions on the right $A_R = \{X \epsilon A : X^j \geq z\}$ and on the left $A_L = \{X \epsilon A : X^j < z\}$. For region A, $\bar{Y}$ is obtained from all the Y_i corresponds to X_i in region A and $\bar{Y}_{A_L}$ and $\bar{Y}_{A_R}$ respectively calculated from Y_i corresponding to the X_i in

regions A_L and A_R. This process will be repeated for the next regions and each time a best cut at the pair of $(j^\star, z^\star)$ will be selected by maximizing the $L_{reg,n}(j, z)$ over set of all the pairs so that the best cut is defined as below:

$$(j^\star, z^\star) \in \arg \max_{\substack{j \in \{1,2,\cdots,p\} \\ (j,z) \in \mathbb{C}_A}} L_{reg,n}(j, z) \tag{1.2}$$

As it can be observed, at each node the criterion (1.1) is evaluated and the best cut at each split is determined. The best cut in Equation (1.2) is not a unique pair and each time for each split it changes and produces different values. Equation (1.1) was first used by (Breiman et al, 1984) which measured the variance before and after splitting (Biau & Scornet, 2016).

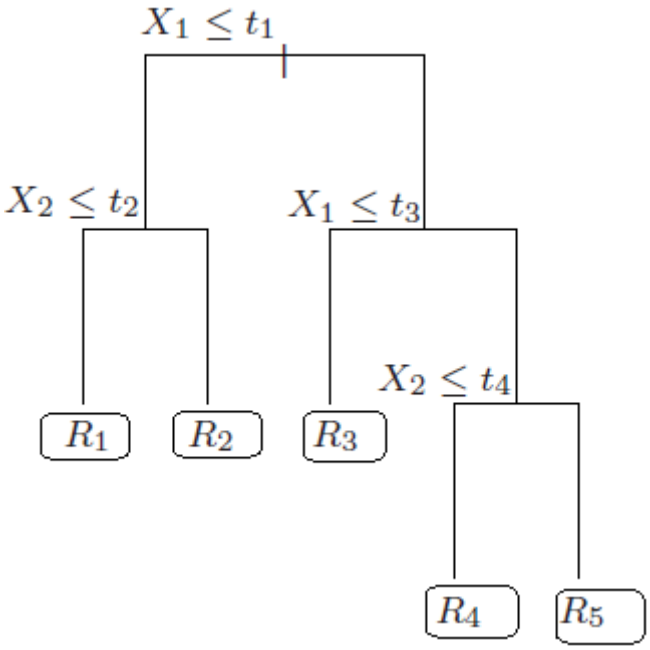
1.2 Estimation

The general idea is to produce a non-parametric regression estimation. The goal is to predict the random response $Y \in \mathbb{R}$ where $E(Y^2 < \infty)$ by estimating this regression function $m(\mathbf{x}) = E[Y|X = \mathbf{x}]$ for an input Random vector $X \in \chi$ where χ is an input space. Having this in mind, for a sample dataset $\mathbb{D}_n = ((X_1, Y_1), (X_2, Y_2), \ldots (X_n, Y_n))$ the estimate $m_n : \chi \longrightarrow \mathbb{R}$ is constructed for the function m. The estimate for a regression random tree at a value $\mathbf{x}$ is defined as follows:

$$m_n(\mathbf{x}, \mathbb{D}_n) = \sum_{i \epsilon \{1,2,\cdots,n\}} \frac{I_{X_i \epsilon A_n(\mathbf{x}, \mathbb{D}_n)} Y_i}{N_n(\mathbf{x}, \mathbb{D}_n)} \tag{1.3}$$

where $A_n(\mathbf{x}, \mathbb{D}_n)$ is the region that contains $\mathbf{x}$ and $N_n(\mathbf{x}, \mathbb{D}_n)$ is the number of observations that belong to the region $A_n(\mathbf{x}, \mathbb{D}_n)$. This estimation is the average of the responses in a region, $A_n(\mathbf{x}, \mathbb{D}_n)$, where the new input $\mathbf{x}$ has fallen (Biau & Scornet, 2016). The process of CART-split (Friedman et al., 2001) for a sample dataset with two random variables X_1 and X_2 is depicted in Figure 2.1 (Friedman et al., 2001).

Figure 2.1: Sample Regression Decision Tree



As displayed in Figure 2.1, at first, the sample space was divided into two regions. Next, as depicted in Figure 2.1, the next region on the left also splitted to two more regions and stopped from splitting in regions R_1 and R_2 which are called terminal nodes. Then, the corresponding region on the right was divided into two more regions and it continued splitting until the number of observations in each node (node size) is less than 5 or the stopping criterion is met (Friedman et al., 2001).

Decision Trees produce estimations with high variance, such that a small change in the dataset will result in a different splitting, which leads to an unstable interpretation of the tree (Friedman et al., 2001), so Bagging was introduced to alleviate this problem.

2 Bagging

Bagging which is the abbreviation of a bootstrap aggregation is known to work well for low-bias and high-variance models like Decision Trees (Friedman et al., 2001). It was introduced by Breiman (1996) and is a procedure in which many trees are built by drawing several bootstrap samples from the dataset and by applying CART-split procedure from Decision Trees on each bootstrap sample. In this process, each tree produces an estimate,

and the final prediction is obtained by averaging over these estimates. Bootstrap sampling which introduced by Efron (1979) is a process of producing several samples of the original dataset with replacement.

In a Bagging procedure, trees are grown from bootstrap samples. In this process, a bootstrap sample of $\mathbb{D}_n^j,\ \ j \in \{1, 2, \cdots, M\}$ from a sample dataset $\mathbb{D}_n$ is drawn and a tree is grown for this bootstrap sample. Then, if prediction of a bagging tree is $m_n^\star(\mathbf{x}, \mathbb{D}_n^j)$ as defined in Equation (1.3), then, the final prediction for Bagging from all trees, $ntree = M$, is:

$$m_{M,n}(\mathbf{x}, \mathbb{D}_n) = \frac{1}{M}\sum_{j=1}^{M} m_n^\star(\mathbf{x}, \mathbb{D}_n^j)$$

The most important point in Bagging is to obtain a smaller variance by averaging over many noisy but approximately unbiased trees. Trees are a perfect option for Bagging because they can capture the complex interactions in the data with relatively low bias if they grow deep enough (Friedman et al., 2001). Trees that are grown in Bagging are identically distributed, and the expectation of the average of trees is the same as a Decision Tree, so the only hope is to reduce variance. Since trees are identically distributed but not necessarily independent, the variance of the average of their estimate for $ntree = M$ trees is (Friedman et al., 2001):

$$\begin{aligned} Var(\frac{1}{M}\sum_{j=1}^{M} m_n^\star(\mathbf{x}, \mathbb{D}_n^j) &= \frac{1}{M^2} Var(\sum_{j=1}^{M} m_n^\star(\mathbf{x}, \mathbb{D}_n^j)) \\ &= \frac{1}{M^2}\{M\sigma^2 + M(M-1)\rho\sigma^2\} = \rho\sigma^2 + \frac{1-\rho}{M}\sigma^2 \end{aligned} \tag{2.1}$$

In Equation (2.1), ρ is the correlation between the predictions for an input $\mathbf{x}$ provided by two trees from the bagging procedure. Also, in this equation, σ^2 represents the variance of the response prediction for an input $\mathbf{x}$ for each tree in the Bagging model. The variance of Bagging as it is depicted is a composite of two terms. Friedman et al. (2001) explains that when number of trees increases, the second term disappears, and the variance of the predictor just depends on the first term. However, even if second term disappears by increasing the number of trees, it will not be helpful when trees are highly correlated. In

this case RF was introduced to reduce the correlation between trees without increasing the variance from predictions of the various trees too much.

3 Regression RF and its characteristics

A regression RF is built by applying Bagging and CART-split procedures from last two sections. In the CART-split process the variable for splitting is selected from whole set of variables $\{X^1, X^2, \cdots, X^p\}$. However, in a RF the splitting variable is selected from $\{X^1, X^2, \cdots, X^{M_{try}}\}$ where M_{try} is the number of selected variables at each split.

3.1 CART-Split

The CART-split process for a RF is different from Bagging such that for all the pair of $(j, z)\, \epsilon\, \mathbb{C}_A$ the dimension $j^\star$ is selected from M_{try}. Therefore, the best cut for a RF is depicted in Equation (3.1) (Biau & Scornet, 2016):

$$(j^\star, z^\star) \in \arg \max_{\substack{j \in M_{try} \\ (j,z) \in \mathbb{C}_A}} L_{reg,n}(j, z) \tag{3.1}$$

When the splitting process starts, each time the criterion (2.1) is evaluated at each node, and the best cut which changes at each split is determined (Biau & Scornet, 2016).

3.2 Estimation

The estimation for a regression RF is very similar to Bagging, but the only difference is that in a RF another type of randomness is injected to the model by choosing a random variable from a subset of variables $\{X^1, X^2, \cdots, X^{M_{try}}\}$ to start splitting at each node. If the $\mathbb{D}_n^j,\ \ j \in \{1, 2, \cdots, M\}$ is the bootstrap sample for a sample dataset $\mathbb{D}_n$, the estimate for a regression random tree, j, in a RF is defined at a value $\mathbf{x}$ of random variable X as follows (Biau & Scornet, 2016):

$$m_n(\mathbf{x}; \Theta_j, \mathbb{D}_n^j) = \sum_{i \epsilon \{1,2,\cdots,n\}} \frac{I_{X_i \epsilon A_n(\mathbf{x}; \Theta_j, \mathbb{D}_n^j)} Y_i}{N_n(\mathbf{x}; \Theta_j, \mathbb{D}_n^j)} \tag{3.2}$$

where $A_n(\mathbf{x}; \Theta_j, \mathbb{D}_n^j)$ is the region that contains $\mathbf{x}$ and $N_n(\mathbf{x}; \Theta_j, \mathbb{D}^j{}_n)$ is the number of observations that belong to that region. Also, $\Theta_1, \Theta_2, \ldots, \Theta_M$ are the parameters that inject the randomness by Bagging and by selecting a variable X^j, a random variable from a subset of variables, that maximizes Equation (1.1) (Cutler et al., 2012). The estimate for the RF is the average of $ntree = M$ trees estimates and it is depicted in Equation (3.3):

$$m_{M,n}(x; \Theta_1, \ldots, \Theta_M, \mathbb{D}_n) = \frac{1}{M} \sum_{j=1}^{M} m_n(x; \Theta_j, \mathbb{D}_n^j) \tag{3.3}$$

The average is obtained over the whole number of trees. The algorithm of producing a Random Forest (Friedman et al., 2001) is displayed below (see Table 2.1):

Table 2.1: Regression Random Forest Algorithm

1) For $j = 1$ to M (number of trees)

Draw bootstrap sample $\mathbb{D}_n^j$ of size n from sample dataset $\mathbb{D}_n$

Grow a random tree from the bootstrap sample by repeating the next steps recursively for each terminal node until the size of each node is $n_{min} = 5$

i) Select M_{try} variables randomly from the p variables

ii) select the best variable with the best split value from M_{try} that maximizes the Equation (1.1)

iii) split the node into two sub-nodes according to the criterion in step *ii*

2) Output the estimation for all the trees

Prediction for an RF:

$m_{M,n}(x; \Theta_1, \ldots, \Theta_M, \mathbb{D}_n) = \frac{1}{M} \sum_{j=1}^{M} m_n(x; \Theta_j, \mathbb{D}_n^j)$

Since M could be any number, it could approach to infinity, so the Equation (3.3) for the RF will approach to the parameter in Equation (3.4):

$$\lim_{M \to \infty} m_{M,n}(x; \Theta_1, \Theta_2, \ldots, \Theta_N, \mathbb{D}_n) = m_\infty(x; \mathbb{D}_n) = \mathbb{E}_\Theta[m_n(x; \Theta, \mathbb{D}_n)] \tag{3.4}$$

The expectation is over the parameter Θ conditional on the sample $\mathbb{D}_n$ and here the

law of large number justifies the operator $\lim_{N\to\infty}$ (Biau & Scornet, 2016).

A RF and a Decision Tree are varying in three different ways. Firstly, in an RF the best variable X^j and best value of its domain is selected from number of variables in the subset M_{try} while in a Decision Tree it is selected from whole set of variables. Second, in an RF, trees are fully grown without pruning[1], but a Decision Tree is pruned to have a better estimate. Finally in a RF, trees are built from a bootstrap sample of the dataset, but a Decision Tree is built from a sample dataste (Biau & Scornet, 2016). Friedman et al. (2001) explained that having the ability to change M_{try} could have an effect on reducing the correlation between trees and reducing the variance. They pointed out Breiman et al. (2001)'s suggestion for M_{try} which was $\sqrt{p}$ for classification and $p/3$ for regression. However these numbers could change with respect to problem at hand, so sometimes it can be observed that some other numbers for M_{try} produce better results.

In the R package the default value for number of trees is 500, and the default number for the M_{try} is $p/3$ for a regression RF. Also, the maximal node size which is the stopping point for splitting for a regression RF is 5; at this point the node will not be split further and the node is regarded as a terminal node. Moreover, the repeating sample size for bootstrap sampling is the same as the size of the data set. Sometimes parameters like node size, size of a tree, and the size of the bootstrap considered as tuning parameters that have an effect on the performance of the RF (Biau & Scornet, 2016).

4 Measuring the performance of a regression model

In Statistical learning the goal is to estimate a response Y by discovering its relationship with some predictors $\{X^1, X^2, \cdots, X^p\}$. The relationship between these components is displayed in Equation (4.1) James et al. (2013) :

$$Y = f(X) + \varepsilon \tag{4.1}$$

As it was defined by James et al. (2013) f would be considered as an unknown function

[1]Pruning is a process in which some of the nodes are removed from the tree and the model error is evaluated so that the new pruned tree with the least error is selected.

but fixed and ε was considered a random error with $E(\varepsilon) = 0$ and $Var(\varepsilon) = \sigma^2$. Also, f represents a mechanism in which $\{X^1, X^2, \cdots, X^p\}$ disclose some information about Y. The prediction of Y is $\hat{Y} = \hat{f}(X)$ where $\hat{f}$ is an estimation of f. There are two quantities that have effect on the accuracy of $\hat{Y}$, which are called irreducible and reducible errors. Since $\hat{f}$ is not always an ideal estimate of f, it introduces some error which is called reducible error. The error is reducible since by fitting a proper model, the error can be reduced. However, even the model is appropriate enough such that $\hat{Y} = f(X)$, some error is present which is called irreducible error. The reason for this error to be present is that Y is also a function of ε and even if the model performs well, the error that is introduced by ε cannot be reduced. This error could be present due to some reasons like not including some variables in the model or it is because of some variation in ε (James et al., 2013). If $\hat{f}$ is an estimate for f, the mean square error of that can be calculated as it is defined in Equation (4.2). It is also considered as a measure of performance:

$$E(\hat{Y} - Y)^2 = E(f(X) + \varepsilon - \hat{f}(X))^2 = E((f(X) - \hat{f}(X))^2) + Var(\varepsilon). \tag{4.2}$$

As James et al. (2013) explained, the first part is the reducible error which can be minimized by applying a proper model and the last part is irreducible error which is out of control. Actually, the most common criterion to measure the regression performance is Mean Square Error (MSE) and when the model is fitted to the sample data it is computed as depicted in Equation (4.3):

$$\hat{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2. \tag{4.3}$$

James et al. (2013) named the MSE in Equation (4.3) as the MSE for training data or training MSE. The training data is the one that was used to develop or train the model and it shows how much $\hat{f}(x_i)$ is close to $f(x_i)$. If the difference between these two functions is very small it implies that the model performed well on the training data and the prediction is close to the observed value. However, the main interest is not really on the performance of a model on training data. The goal is to see if this model performs

well on a new data which is called a test data and is independent from the training data.

James et al. (2013) pointed out to the fact that when a model fits well on a dataset, it is a positive point, but we are more interested to see if it has the same or close performance on a new similar dataset. For instance, if we have a number of patients with their clinical measurements and we want to predict the risk of diabetes, a model can be applied for this data and fits the data very well, but we want to use this model for future patients and predict the risk of diabetes based on their inputs.

James et al. (2013) described how the MSE for a new observation from a new dataset is computed. They explained when a model is developed by training data, the goal is to find if $\hat{f}(x_0)$ can produce an accurate estimate for $f(x_0)$ which is the prediction for a new observation x_0. To measure the performance of a model on a new observation, the MSE of that is computed by Equation (4.4) which is called a generalization error:

$$E(y_0 - f(x_0))^2 = var(\hat{f}(x_0)) + [bias(\hat{f}(x_0)]^2 + var(\varepsilon). \tag{4.4}$$

In Equation (4.4) the goal is to reduce the variance and bias as much as possible to have a better performance for the model. However, somtimes it has to be decided for a trade-off between these two factors. Generally, a test dataset is not available to measure the performance of a model, so when we have enough observations it is recommended to divide the dataset into two parts one for training and one for testing. If the performance of the model is assessed on the test set, most of the time its MSE is larger than the training MSE because some of the patterns that have been caught in the training set could not exist in a test set as they are purely due to noise, and this is called overfitting. Although it has to be noted that whether overfitting is present or not, the test MSE is larger than the training MSE because the model directly or indirectly tries to minimize the training MSE. In general, when the test MSE is much larger than the training MSE it is regarded as overfitting (James et al., 2013).

Friedman et al. (2001) also introduced another way of performing data partitioning. They suggested that we have two goals in modeling which are either model selection or model assessment. In model selection the performance of different models could be

compared to select the best model. In model assessment, the final model would be selected by estimating the generalization error on new data. If we want to address both needs, the best way is to divide the dataset into three parts which are training, validation, and test sets. The training set is used for developing the model, the validation set is used for model selection by estimating the prediction error, and the test set is used to evaluate the generalization error of the final model. There is no specific rule stating how to split the data into the three subsets, and it depends on the size of the dataset and some other factors, but one of the suggestions provided by Friedman et al. (2001) is to divide it to 50%, 25%, 25% of the data for training, validation, and test sets.

5 Out of Bag Error

As discussed in previous section, one of the ways to evaluate the performance of a model is to measure the MSE on a test set which most of the time is not available. In Random Forest this problem has been addressed by measuring the out of bag error which is the MSE of the test data that is available internally during the process of growing trees using bootstrap sampling.

When Bootstrap sampling procedure is applied to grow a tree, about one third of data is not included in the sample and they are regarded as out of bag observations. In fact, for each Bootstrap sample a tree is grown using two third of the sample and out of bag observations are the one third left out observations. These observations are used as a test set to measure the performance of the model (Cutler et al., 2012). The reason that one third of the data is dropped out of sampling as Louppe (2014) described is because after n draw with replacement the probability of not being selected can be calculated as expressed in Equation (5.1):

$$(1 - \frac{1}{n})^n \approx \frac{1}{e} \approx 0.368. \tag{5.1}$$

As Cutler et al. (2012) described these observations are used to assess the performance of the model by estimating the out of bag error. In fact, each of these observations are dropped from the tree, after ending up in a node, the prediction for them will be cal-

culated, so the mean square error can be computed. If $\mathbb{D}_n^j$ is the bootstrap sample and $\widehat{m_n(x_i;\Theta_j,\mathbb{D}_n^j)}$ is the prediction at x_i for tree j and the set of out of bag observations is defined as : $\upsilon_i = \{j : (x_i, y_i) \notin \mathbb{D}_n^j\}$ and the number of the members of the set is ι_i then out of bag estimation is defined as below:

$$\hat{f}_{oob}(x_i) = \frac{1}{\iota_i} \sum_{j \in \upsilon_i} \widehat{m_n(x_i;\Theta_j,\mathbb{D}_n^j)} \tag{5.2}$$

Where out of bag mean squared error is calculated:

$$\hat{MSE}_{oob}(reg) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}_{oob}(x_i))^2 \tag{5.3}$$

6 Cross Validation

There is another way to measure the prediction error or performance of a model which is very simple and popular, and it is called cross validation. This method estimates the expected error of the extra sample, the average generalization error, when the developed model on the training set is applied on a test set. Most of the times, this method is applied since the size of the dataset is not large enough to divide the dataset to different parts. The general method is called K-fold cross validation, and K could be different numbers but mostly it is recommended to set it to 5 or 10 (Friedman et al., 2001).

In this method the data is divided into K equal parts and the model is fitted on the $K-1$ sets and the error of the K^{th} part is measured as the test set error and recorded K times. Finally the average of these test set errors is computed as the prediction error of the model. Cross-validation works well on the expected error (Friedman et al., 2001).

7 Variable Importance

Despite the fact that a RF cannot be interpreted like a Decision Tree, it measures the variable importance, which is useful for interpretation and variable selection (Cutler et al., 2012). In this process, the variables which are more important are selected based on

two criteria: Mean Decrease Impurity (MDI) and Mean Decrease Accuracy (MDA). When the focus is on the effect of a variable in maximizing the CART-split criterion in Equation (1.1), averaging over all the trees, MDI would determine the important variables. On the other hand, MDA ranks the variables based on how much that variable minimizes the mean square error of a region. The MDI of a variable X^j for $ntree = M$ trees of an RF is calculated by the criterion in Equation (7.1):

$$\widehat{MDI}(X^j) = \frac{1}{M}\sum_{k=1}^{M}\sum_{\substack{t\in\tau_k \\ j^\star_{t,n}=j}} p_{t,n}L_{reg,n}(j^\star_{t,n}, z^\star_{t,n}) \tag{7.1}$$

In Equation (7.1) $p_{t,n}$ is the portion of the observations that fall in node t which includes intermediary and terminal nodes, $\{\tau_k\}_{1\leq k\leq M}$ is the group of trees in the RF, "n" represents the sample size, and $(j^\star_{t,n}, z^\star_{t,n})$ is the best cut that maximizes Equation (1.1) in node t. Equation (7.1) is the average of total improvement in CART-split by variable X^j that produces the best cut among number of variables in M_{try} for a node. In fact, variables that maximize Equation (1.1) are considered in this method and their weighted results would incorporate in producing the final resul (Biau & Scornet, 2016).

If prediction accuracy of a model (reducing the out of bag error) is considered, MDA would be selected (Biau & Scornet, 2016). If the goal is to measure the importance of variable X^j, the following steps will be taken. First, the out of bag observations are dropped from the tree and the predicted values for these observations are obtained. Second, the values of variable X^j in the out of bag observations are randomly permuted while other predictors are fixed. These reshaped out of bag observations are dropped from the tree and the predicted values for them is computed. This process will result in two sets of data: one from real values of variable X^j and other one from permuted values of this variable. The measure of variable importance is computed by taking the difference between the MSE of the predictions from the real data and MSE of the predictions from permuted data. The final variable importance is obtained by taking the average over all the observations. (Cutler et al., 2012).

As (Biau & Scornet, 2016) defined out of bag data of sample size n for variable X^j is

$\mathbb{D}_{l,n}$ and the data after permutation is $\mathbb{D}^{\urcorner}{}_{l,n}$ for l^{th} tree. Also, $m_n(\mathbf{x};\Theta_l,\mathbb{D}_{l,n})$ as defined before in section 2.3 is the estimate for l^{th} tree, the estimate of out of bag error MSE_n is computed by Equation (7.2)

$$MSE_n[m_n(\mathbf{x};\Theta_l,\mathbb{D})] = \frac{1}{|\mathbb{D}|}\sum_{i:(X_i,Y_i)\in\mathbb{D}}(Y_i - m_n(X_i;\Theta_l,\mathbb{D}))^2 \tag{7.2}$$

The estimate MSE_n is computed by Equation (7.2) by considering $\mathbb{D} = \mathbb{D}_{l,n}$ or $\mathbb{D} = \mathbb{D}^{\urcorner}{}_{l,n}$. MDA mathematically is obtained from Equation (7.3):

$$\widehat{MDA}(X_j) = \frac{1}{M}\sum_{l=1}^{M}[MSE_n[(\mathbf{x};\Theta_l,\mathbb{D}^{\urcorner}{}_{l,n})] - MSE_n[(\mathbf{x};\Theta_l,\mathbb{D}_{l,n}))]] \tag{7.3}$$

The first term produces the out of bag error for permuted observations of variable X^j and the second term produces the out of bag error for regular observations. The difference is the average over whole number of trees $ntree = M$. This measure shows how much the error will increase or decrease after permutation of a random variable. If the value for difference is high, it is due to getting high value for the out of bag error after permutation, which means that the variable is important. On the other hand, small values for the difference is due to the fact that, the variable is not important (Cutler et al., 2012).

8 Proximities

Two features of a RF is missing values imputation and outlier detection which is possible by using proximity measure between two observations. This measure is defined as the number of times two observations end up in the same terminal node. If two observations ended up in the same terminal node, their proximity is one; otherwise the proximity will be zero (Cutler et al., 2012). The proximity between two observations in a terminal node can be measured by Equation (8.1) (Louppe, 2014):

$$Proximity(x_1,x_2) = \frac{1}{M}\sum_{l=1}^{M}\sum_{t\in\tau_l'} I(x_1,x_2\in\chi_t) \tag{8.1}$$

In Equation (8.1), τ_l' is the set of terminal nodes for the l^{th} tree and χ_t is the t^{th} terminal node. Also, $ntree = M$ is the total number of trees in a RF (Louppe, 2014). This equation computes the number of times a pair of observations would settle in the same node among all the trees and divided that to total number of them. If this proximity is close to one they would be similar, if it is close to zero they would be regarded different and would be in separate nodes (Cutler et al., 2012).

9 Missing value imputation

The RF can be considered as a method that handles missing values really well. Missing values could be imputed using proximity measures in a RF itself or through the MissForest method.

9.1 RF imputation

If missing values are present just for predictors, they can be imputed by proximity measures of an RF, the procedure of which is as follows. Firstly, an RF is built with missing values that are replaced with the sample median of all the observations for considered random variables. In the next steps the sample median is updated with new imputed value by using the proximity measures. In the first RF it is evaluated to which terminal nodes the missing values (which replaced by sample median) were ended up and their proximity weights with respect to other observations in that node were computed for missing values. Having the proximity weights the second RF is produced and the imputed values of these missings are obtained by weighted average of those observations in that node. The weights of the observations are obtained by proximity matrices using Equation (8.1). The updated imputed values is computed and it will be used to produce the next RF. In the next RF, the missing values are updated by proximity measures and this process repeats several times by producing several Forests. It continues until the imputed values converge to a number and cannot be updated anymore. At this stage these final values are regarded as the final imputed missing values (Cutler et al., 2012).

9.2 MissForest

The second method of imputation is MissForest which can handle missing values for both categorical, continuous variables and for response variable Stekhoven and Bühlmann (2011). The authors stated that this imputation method was very efficient computationally and had a great performance when the number of predictors is very large. MissForest imputes missing values directly from the observed values. It divides the dataset into four parts regarding the observed and missing values of a variable $\mathbf{X}_s$. Suppose the observed and missing values of $\mathbf{X}_s$ are denoted by $\mathbf{Y}_{obs}^{(s)}$ and $\mathbf{Y}_{miss}^{(s)}$ respectively, $\mathbf{X}_s = [\mathbf{Y}_{obs}^{(s)}, \mathbf{Y}_{miss}^{(s)}]$. The components of the other variables $\mathbf{X}_r,\ r = \{1, 2, \cdots, p\}\ , r \neq s$, correspond to observed and missing values of variable X_s are denoted by $\mathbf{X}_{obs}^{s}$ and $\mathbf{X}_{miss}^{(s)}$, $\mathbf{X}_r = [\mathbf{X}_{obs}^{(s)}, \mathbf{X}_{miss}^{(s)}]$

The imputation starts with replacing missing values by mean of the observations for variables $\mathbf{X}_s, s = \{1, \ldots, p\}$. Then the variables are sorted from smallest to the largest amount of missing values. The missing values imputation for each variable $\mathbf{X}_s$ continues by fitting Random forest on the predictors $\mathbf{X}_{obs}^{(s)}$ and the response $\mathbf{Y}_{obs}^{(s)}$ and applying it to predict $\mathbf{Y}_{miss}^{(s)}$ from $\mathbf{X}_{miss}^{(s)}$. This procedure repeats until the stopping point criterion holds, which is when the difference between the new and old imputed values increases for the first time with respect to both types of variables if applicable. Stekhoven and Bühlmann (2011) defined the difference for continuous variables as expressed in Equation (9.1):

$$\triangle_N = \frac{\sum_{j \in N} (\mathbf{X}_{new}^{imp} - \mathbf{X}_{old}^{imp})}{\sum_{j \in N} (\mathbf{X}_{new}^{imp})^2} \tag{9.1}$$

and their criterion for the difference between new and old imputed values for set of categorical variables F is defined in Equation (9.2):

$$\triangle_F = \frac{\sum_{j \in F} \sum_{i=1}^{n} I_{\mathbf{X}_{new}^{imp} \neq \mathbf{X}_{old}^{imp}}}{\#NA} \tag{9.2}$$

Stekhoven and Bühlmann (2011) also defined a criteria to measure the performance of the model after imputation. There are two measures: "Normalized Root Mean square error" (NRMSE) for continuous variables and "Proportion of Falsely Classified entries"

(PFC) for categorical variables. The NRMSE was defined in Equation (9.3):

$$NRMSE = \sqrt{\frac{mean((X^{true} - X^{imp})^2)}{VAR(X^{true})}} \tag{9.3}$$

Where X^{true} is the complete dataset and X^{imp} is the imputed one. The notations VAR and mean are the sample mean and variance which are computed for missing continuous values only. The PFC is the percentage of entries that are miss-classified over the number of missing values for categorical variables and that is defined as above $\triangle_F$. If values for these two items are close to zero, it will be an indication of good performance and if it is close to one it represents a bad performance for imputation. (Stekhoven & Bühlmann, 2011).

Chapter 3

Genetic Algorithm

Haupt and Haupt (2004) explained the process of natural selection and evolution with the purpose of getting an insight into the process of a Genetic Algorithm mechanism. It can be supposed that the new generation of organisms are produced through a process of optimization, in which the goal is to maximize their survival. The most fit cases are the ones that can last longer. In the process of reproduction and creating a new generation, genetics and evolution play an important role.

The basic unit of inheritance is a gene which is a component of a chromosome, and the process of reproduction occurs at this level. The group of chromosomes that can match and reproduce is called the population. The reproduction starts with the cell division such that chromosomes with the same size and shape are selected, and half genes from mother and half genes from father's chromosomes join. The match is such that the left part of the chromosomes of the mother are joined with the right part of the chromosomes of the father. This process is called crossover and leads to observing some variety in the species. The other process that brings more diversity in the next generation is mutation which could be a random change of a gene due to an external force or could happen internally (Haupt & Haupt, 2004).

As Haupt and Haupt (2004)explained, each time a new generation is produced through the process of selection, crossover, and mutation and evolves. The evolution of these generation as Darwin explained includes four components:

- Many characteristics of the parents would pass to the offspring.
- Individuals have various characteristics that could be passed to the new generation.
- A small number of the offspring could survive.
- The survival of the offspring depends on their characteristics.

Therefore, as it can be observed the process of selection, crossover, mutation and their evolution leads to creating each new generation. The same approach holds for the GA which will be explained in next section (Haupt & Haupt, 2004).

1 Definitions

Since GA mechanism is similar to the biological process of reproduction, some of its related terminology is borrowed from biology. However, the terminology in GA is much simpler than the one in the biological process. Some of the definitions that are more common between these two process are explained as follows (Haupt & Haupt, 2004):

- Chromosome: A group of parameters or genes that are plugged into the fitness function.
- Fitness function: is a function that has to be optimized by providing a fitted value.
- Generation: An iteration in the process of implementing the Genetic Algorithm
- Population: A number of chromosomes (solutions) that mix together to produce the next generation of the solutions.
- Population size: The size of initial and evolved population during the process of optimization.
- Search Space: Possible values of the parameters (solutions) of the function
- Selection: It is a process in which parents are selected for the reproduction.

- Crossover: It is a process in which new offspring (solution) is reproduced by exchanging part of information from parents.
- Mutation: It is a random change in the genes (parameter) of a parent (current solution).
- Probability of crossover: It is the percentage of population that are selected to do the crossover.
- Probability of mutation: The percentage of population that are selected for the mutation.
- Elitism: The percentage of best fitted solutions that survive to the next generation.

2 Types of GA

Carr (2014) explained that there are three types of Genetic Algorithms with regard to the solutions:

- Binary: A string of zero's and one's
- Permutation: combinations of the items
- Real-valued: continuous values

Carr (2014) illustrated that the fitness function in Binary GA could be optimized by converting the solutions to the binary strings. The initial population of solutions to the problem are encoded to an array of binary values like a set of genes in a chromosome of an organism. An example of the solutions (chromosomes) could be these two members of the population [11010111001000] and [01011101010010]. These solutions are plugged in the fitness function, and the fitness value is calculated.

The next step is the selection process in which the fittest solutions are selected to produce the new solutions or offspring. There are different ways of selecting the parents which will be explained later. In this step the function searches solution over a search space

and finds the fittest ones. When parents are selected they produce the new generation through crossover and mutation.

Carr (2014) defined crossover and mutation through an example. For instance if two binary solutions from the initial population are [1101***0111001000***] and [0101**1101010010**], and they crossover after the fourth digit, their new offspring are [1101**1101010010**] and [0101***0111001000***]. Mutation brings more diversity in the population and does not let the algorithm converge very fast. Mutation could happen before selection and crossover or after that. If mutation occurs after crossover, one of the bits in new solution or offspring would be flipped from zero to one or vice versa. For instance in the first offspring above, the mutation could occur on the 11^{th} point and change the zero to one as follows [1101**1101011010**] and the final new solution is reported.

This process of selection, crossover, and mutation repeats for next generations and each time the best solution with the best fitness value is recorded. The process of evolution is continued until the convergence criteria is met. This could be for an algorithm to reach a maximum number of iterations. The process of all the iterations which leads to a value as a solution is called a ***Run***. The second type of convergence occurs when a quite large number of generation passed with almost the same fitness value or without any improvement in it. The third type of convergence is when there is a predefined bound for the population statistic (scrucca2013ga). Performance of a GA depends on the fitness function and parameters like probability of crossover, probability of mutation, size of the population, and the number of iterations which can be adapted after some primary runs (Carr, 2014).

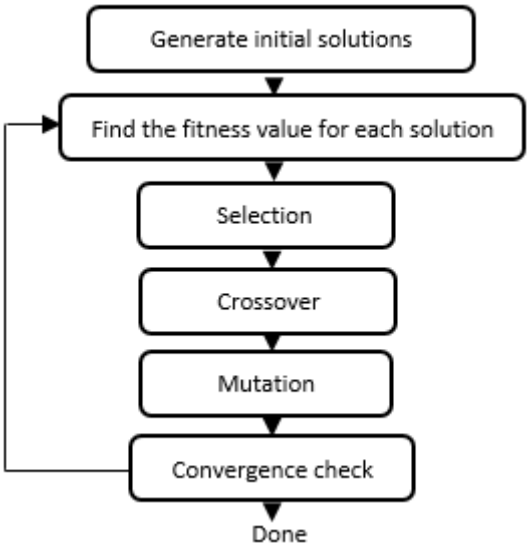
As Carr (2014) explained, Permutation type of Genetic Algorithm is an optimization problem for combinations which means the function is optimized by selecting the best permutation or order of the numbers or items. The very famous problem of permutation type of a GA is "The Traveling Salesman Problem" (TSP).

2.1 Real-valued GA

When we are interested in optimizing a function, the variables of which are continuous, we are required to apply Real-valued GA. This type of GA is less time consuming since the solutions do not need to be converted to binary values as in the Binary type of GA (Haupt & Haupt, 2004). If the fitness function is bi-variate, $f(x, y)$, with continuous variables x and y, it will report two solutions to the problem. The domain of the variables has to be determined for the function to help GA to limit the search space to a reasonable size (Haupt & Haupt, 2004).

Firstly, the process of optimization of a real-valued GA starts with an initial population of continuous solutions. Secondly, the fitness values of these initial solutions are computed using the fitness function. Then, it will be investigated if they are optimal numbers and the algorithm converges or if they have to be evolved using GA operators through the process of selection, crossover, and mutation. If the initial solutions are not optimal, the second generation of solutions will be produced using the GA operators and the convergence will be checked. This process could continue for several generations until the optimal numbers will be obtained and the GA converges. The whole process of GA which will lead to producing the optimal solutions is depicted in Figure 3.1(see (Haupt & Haupt, 2004)).

Figure 3.1: The process of GA



There are different types of GA operators, and various combinations of them could be

used to produce the optimal solutions. Some of the GA operators will be explained in the following sections.

2.2 Selection

The selection process for reproduction is a very crucial step in the optimization of a GA. The selection mechanism determines the individuals that need to be selected to produce the new generation. Also, there is an important parameter in the selection process called selection pressure which aims for for the selection of the fittest solutions or individuals for producing the next generation. If selection pressure is too low then it slows down the process of convergence to find optimum solutions. If it is too high, the algorithm converges too fast to a local optimum since it does let less fitted solutions to be selected which leads to less diversity in the population. Therefore, it is important to know which selection method to choose that help the algorithm converges to the global optimum solution (Oladele & Sadiku, 2013).

- Roulette Wheel Selection (RWS): Razali, Geraghty, et al. (2011) explained that in this method each individual solution has a probability of being selected proportional to its fitness value. If a solution is denoted by $f_i,\ \ i = 1, 2, \cdots, n$, the probability for a solution to be selected is $p_i = \frac{f_i}{\sum_{i=1}^{n} f_i}$. In this approach the solution with the highest probability have higher chance of being selected. The process of selection in roulette wheel method can be summarized in these steps:

  - $S = \sum_{i=1}^{n} f_i$ is the sum of all the fitness values
  - $\alpha \in (0, S)$ is a random number generated
  - go through the solutions and calculate the sum of fitness values $S' = \sum_{i=1}^{j} f_i$. If $\alpha < S'$ then solution j is selected.
  - repeat the last two steps to produce enough pairs that could mate and produce new generation or solutions.

  For instance, if the current solutions are $f_i = \{1, 5, 6, 7, 11, 20\}$ and the sum is $S = 50$ and α is selected randomly from this range $\alpha \in (0, 50)$ such as $\alpha = 23$.

Also, the $S' = 1 + 5 + 6 + 7 + 11 = 30$ is computed. since $\alpha < S'$ then 11 is the one that is selected. This method gives chance to every individual to be selected but the opportunity is not the same for every individual. The ones with the largest probability have more chance to be selected since the selection pressure is too high. They dominate the population and do not let the other individuals to be involved which leads to less diversity in the next generation. Therefore, the algorithm could produce the local optimum rather than global.

- Rank Selection: This method was introduced to bring more diversity to the solutions that cannot be obtained in the previous method due to fittest solutions dominance. In fact this method gives every solution the same chance of being selected. In this method the individuals are ranked according to their fitness values. The individual with the lowest fitness value ranks 1 and the one with the highest fitness value ranks n (Kumar, 2012). Rank selection method prevents the algorithm to converge too fast and will not let the algorithm get stuck in the local optimum. The process of selection in Rank Selection method by considering the rank of a solution as r_i can be summarized in these steps (Kumar, 2012):

  - $S = \sum_{i=1}^{n} r_i$ is the sum of all the fitness values
  - $\alpha \in (0, S)$ is a random number generated
  - go through the solutions and calculate the sum of fitness values $S' = \sum_{i=1}^{j} r_i$. If $\alpha < S'$ then the solution j is selected.
  - repeat the last two steps to produce enough pairs that could mate and produce new generation or solutions.

  For instance, if the current solutions are $f_i = \{1, 5, 6, 7, 11, 20\}$ and the sum is $S = 21$ and α is selected randomly from this range $\alpha \in (0, 21)$ such as $\alpha = 14$. Also, the $S' = 1 + 2 + 3 + 4 + 5 = 15$ is computed. since $\alpha < S'$ then 11 is the one that is selected. The problem with this method is that it converges very slowly because it is based on the ranks of the solutions, but it is more robust comparing to the other method of selections. Rank selection method includes two types of methods:

– Linear Rank Selection: This method was proposed by Blickle and Thiele (1995). In this method the probability of selection is linearly proportional to their rank

$$p_i = \frac{1}{n}(\eta^- + (\eta^+ - \eta^-)\frac{i-1}{n-1}), \quad i \in \{1, 2, \cdots, n\}$$
$$\eta^+ = 2 - \eta^-, \quad \eta^- > 0 \quad (2.1)$$

In Equation (2.1) the probability of the worst solution to be selected is $\frac{\eta^-}{n}$ and the probability of the best solution to be selected is $\frac{\eta^+}{n}$. Also, the constraints $\eta^+ = 2-\eta^-, \quad \eta^- > 0$ must be achieved as the population size is held invariant. In this method even the individual solutions with the same fitness value get different probability of being selected . In this equation if the fitness value of the fittest individual is set to 2 or larger than that, then the worst individual does not have the chance to be selected for the reproduction. By setting $\eta^+ \geqslant 2$ the worst solutions fitness values will be negative which means they cannot produce any new solutions (Blickle & Thiele, 1995). Another method that gives a chance to the worst members of a population to be selected is the Non-linear Rank selection.

– Non-linear Rank Selection: In this method the probability of selection is not proportional to the rank. it is proportional to a non-linear function of the rank. In this method the selective pressure is higher than linear rank selection, so the algorithm converges slowly to find the optimal solution (Kumar, 2012).

- Linear Scaling Selection(Goldberg & Holland, 1988): This process of selection is defined to overcome the problem of the fittest solutions domination in Roulette wheel selection method. In this method, a linear function f_{scaled} of the current fitness function f_{raw} is defined as $f_{scaled} = af_{raw} + b$. The coefficients a and b are selected such that $f_{raw}^{avg} = f_{scaled}^{avg}$. Also, $f_{scaled}^{max} = Cf_{raw}^{avg}$ in which $C \in (1.1, 2)$ is the number of expected desired copies of best solutions that can be selected to produce the next generation of solutions. In fact, C controls the number of fittest solutions to be selected for the reproduction. The choice of C pulls the fitness function out

remarkably when the algorithm reaches to the end of the run. This situation causes some problem in using linear scaling. There are some bad solutions with fitness values that are very far from the average and the maximum which are almost close in a mature run. When scaling is applied, since the maximum and average fitness values are so close, it affects the scaled bad fitness values and they become negative. In this case one of the solutions is to map these negative values to zero for the bad fitness values and keep the rest stay the same as they are.

- Sigma Truncation Selection: The previous method works well and prevents the domination of powerful solutions. However, some weak solutions produce negative amounts for the scaled fitness function f_{scaled} due to scaling. In this case, it is suggested using the variance of fitness values of the population before scaling. In this process, before scaling a constant is deducted from the raw fitness value and the new fitness function f' is computed. The new fitness function is $f' = f_{raw} - (\bar{f} - c \times \sigma)$ where $\bar{f} = \sum_{i=1}^{n} f_i$ and σ is the standard deviation of the fitness values of the population. Also, $c \in (1, 3)$ is selected as a multiple of σ. After sigma truncation is incorporated, and new fitness values f' are computed, the linear scaling procedure which explained in the previous part can be applied. Sigma truncation prevents from having negative values for scaled fitness values (Goldberg & Holland, 1988).

- Tournament Selection (Jebari & Madiafi, 2013): In this process a subset of the population $k < n$ is selected and from this subset the fittest individual will be the one to be chosen for reproduction. This process repeats n times and for the whole population. Since this method does not involve sorting, it works really well for large populations.

2.3 Crossover

When the parents are selected by one of the methods that described above, then we need to decide how they are going to mate and produce the next generation which leads us to the topic in this section, crossover. There are various methods for crossover which

was explained by Adewuya (1996):

- Single point crossover: In this method, parents that are selected from the initial population exchange some part of the information to produce the new generation. For instance consider there are k variables involved to the problem that have to be solved. The solution is a k-dimension vector, and the two parents as an example could be $F_1 = [f_1, f_2, f_3, f_4, \cdots, f_k]$ and $F_2 = [f'_1, f'_2, f'_3, f'_4, \cdots, f'_k]$. The new offspring can be produced by choosing randomly a single point such as point three in the parents and exchange the items between them as follows to produce new solutions: $F_1^{new} = [f'_1, f'_2, f'_3, f_4, \cdots, f_k]$ and $F_2^{new} = [f_1, f_2, f_3, f'_4, \cdots, f'_k]$. This method is rarely used, and if it is the case, mutation rate has to be high to bring more diversity to the solutions by introducing new members.

- Uniform Crossover: In this method the crossover occurs on multiple points randomly. If two parents are $F_1 = [f_1, f_2, f_3, f_4, \cdots, f_k]$ and $F_2 = [f'_1, f'_2, f'_3, f'_4, \cdots, f'_k]$, The new offspring could be produced by choosing randomly a single or multiple points in the parents chromosomes and exchange the items between them as follows $F_1^{new} = [f'_1, f_2, f'_3, f_4, \cdots, f_k]$ and $F_2^{new} = [f_1, f'_2, f_3, f'_4, \cdots, f'_k]$.

- Local Arithmetic crossover: If F_1 and F_2 are the parents, the offspring is produced from these three linear combinations

$$F_1^{new} = 0.5F_1 + 0.5F_2 \qquad F_2^{new} = 1.5F_1 - 0.5F_2 \qquad F_3^{new} = -0.5F_1 + 1.5F_2$$

  The combinations occurs gene by gene in the parents, and the best two out of these three are selected as the new offspring. This method brings more diversity to the solutions.

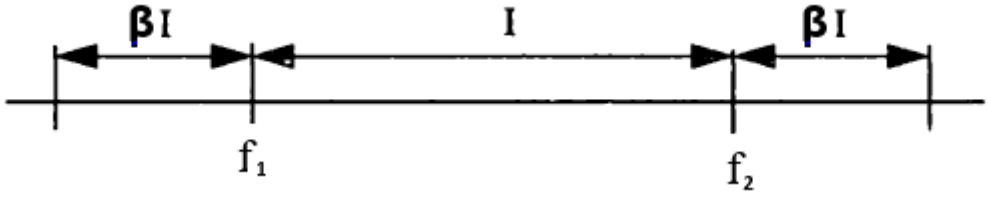
- Whole Arithmetic crossover: If F_1 and F_2 are the parents and the offspring is created through this combination:

$$F_1^{new} = aF_2 + (1-a)F_1 \qquad F_2^{new} = aF_1 + (1-a)F_2$$

where $a \in [0, 1]$ is randomly selected, it is called whole arithmetic crossover.

- Blend Crossover (**BLX**$-\beta$): In this method a parameter β is selected that identifies a bound outside the range of values between two parents to produce the new offspring. Figure 3.2 depicts how this type of crossover works:

Figure 3.2: Blend crossover (**BLX** $- \beta$) method



If $\beta = 0$ then, it is called flat crossover, and the values are selected from the outside the range of parents values. Parameter β stretches the range of the new generation but not too far.

2.4 Mutation

When the methods for selection and crossover are selected, the next step is to introduce mutation to the algorithm. Mutation brings more diversity to the solutions and also prevents the algorithm from converging too quickly on a local optimum value. There are different methods for mutation as well (Adewuya, 1996):

- Uniform random mutation: If the solution vector for the fitness function and for generation t is $F^t = (f_1, f_2, \cdots, f_j, \cdots, f_k)$ and every item has the same chance to be selected, the new solution could be $F^{t+1} = (f_1, f_2, \cdots, f'_j, \cdots, f_k)$ where f'_j, $1 \leq j \leq k$ could be any number between lower and upper bound of the f_k. This method produce solutions close to the original solutions.

- Non-uniform random mutation: If the solution vector for the fitness function is $F^t = (f_1, f_2, \cdots, f_j, \cdots, f_k)$ and every item has the same chance to be selected, the

new solution could be $F^{t+1} = (f_1, f_2, \cdots, f'_j, \cdots, f_k)$ where:

$$f'_j = f_j + y \times r \left(1 - \frac{t}{T}\right)^b$$

where t is the number of current generation, T is the maximum number for generation, r is a random number that is generated and it is between 0 and 1,b determines a non-uniformity is the system parameter, and y takes values between -1 and 1 with probability of 0.5.

- Random mutation around the solution: If the solution vector for the fitness function is $F^t = (f_1, f_2, \cdots, f_j, \cdots, f_k)$ and every item has the same chance to be selected, the new solution could be $F^{t+1} = (f_1, f_2, \cdots, f'_j, \cdots, f_k)$ where f'_j could be the lower bound L_j or the upper bound U_j of the f_j, $1 \leq j \leq k$. This method creates solutions with less diversity since the new solution can take values either as the upper or lower bound.

All these methods are the ones that are defined for GA package in R. There are other parameters that have effect on the performance of GA such as probability of crossover, probability of mutation, and elitism. In next section it is discussed how these parameters could have impact on the performance of GA.

2.5 GA control parameters

All the methods that are described above are part of the GA operators: selection, crossover, and mutation. The choice of these methods has effect on the performance of GA. There are also some other parameters which are called control parameters in the GA function that have effect on its performance like population size, probability of crossover, and probability of mutation. Most of the time, the best values for these control parameters are determined by trial and error. There is no single optimal number for these parameters and it changes among various problems or even at different levels of the GA process. He presented some of the studies that investigated the optimal numbers for probability of crossover and probability of mutation (Patil & Pawar, 2015).

DeJong (1975) explained that high values for the mutation rate decrease the performance of GA regardless of what values to be assigned to the other parameters. One of their suggestions for these parameters is a population size of (50-100), a single point probability of crossover (0.6) and a probability of mutation of (0.001) which have been used in many Genetic Algorithms.

Grefenstette (1986) illustrated that for a low population size between $(20, 40)$ the GA performs well either with high/low probability of crossover with low/high mutation error. One of his suggestions is a population size of 30 with mutation probability of 0.01 and with crossover probability of 0.95. Schaffer et al. (1989) also studied and concluded that the GA performance is more sensitive to the crossover probability than mutation probability. He suggested some parameter setting very close to what Grefenstette (1986) proposed. He proposed a population size of (20-30) with probability of mutation between (0.005-0.01) and crossover rate between (0.75-0.95). Also, there are some research showing that changing the mutation rate has a positive effect on GA and it has to decrease over the time of process.

As it can be noticed there are different possibilities for control parameters, and it has to be decided based on the time of the process, accuracy of the results, and the type of a problem which one to select.

As mentioned before, the GA package in R could be used for the analysis. In the R package the GA maximizes the fitness function, so for minimizing a function, the negative of that function has to be maximized. The function needs an initial population which could be generated from a random (uniform) population of real values in the range of the variables.

Chapter 4

GA optimization for RF based Methods

As discussed before the number of trees, $ntree$, and number of features at each split, M_{try}, are mostly regarded as tuning parameters for a RF, and they are considered as the most important ones that have impact on performance of the RF (Elyan & Gaber, 2017). These two parameters are also inputs for MissForest. Since MissForest is a RF based method, these parameters could also have an impact on its performance. The goal is to apply GA to find the optimized values for these parameters that minimize the MSE of the RF and imputation error of MissForest. In this section, it will be elaborated how these optimized methods work, and they are compared with classic models.

1 MissForest Optimization (MissRFGA)

Since a RF cannot be implemented when missing values are present, first the MissForest approach and its optimization process was explained. In this study, if datasets include missing values, they are imputed by MissForest and by an optimized version of that function. If they did not have missing values, 10% of the datasets was randomly converted to missing just to compare the performance of both classic and optimized models.

In the optimization process, the GA was applied to produce the solutions for the optimized MissForest. The solutions are optimized values for $ntree$ and M_{try} that minimize the imputation error of the MissForest which is a fitness function for GA. At the beginning, an initial population of solutions was required to start this process. The optimization

started with this population, a population of 50 solutions, and evolved over a number of generations to find the optimal values to improve the performance of the model. The component of the population were values for this vector $V = [ntree, M_{try}]$. The number of trees was between, $ntree \in [50, 500]$, and number of features was set between these values $M_{try} \in [1, p]$ where p is the number of parameters. The range for $ntree$ was selected to have enough number of trees to reduce the variance. Also, the number of features includes all the choices to see if the boundaries will be selected as solutions. The default setting for MissForest is $ntree = 100$ and $M_{try} = \sqrt{p}$.

In this process, at first a fitness function was defined which was called later by GA. The fitness function is the imputation error of the model and its computation algorithm is depicted in Table 4.1 (Elyan & Gaber, 2017):

Table 4.1: Compute fitness function for MissForest

begin

$A \leftarrow Dataset$

$ntree \leftarrow ntree0$

$M_{try} \leftarrow Mtry0$

$(ntree, M_{try}) \leftarrow (ntree0, Mtry0)$

$model \leftarrow fit\ missRF(A, ntree, M_{try})$

$imputation\_error \leftarrow evaluation\ of\ the\ model$

return(imputation_error)

$(ntree, M_{try}) \leftarrow Solutions$

repeat the process from step 5

end

In this algorithm as it can be observed, at first a primary number was assigned to both number of trees, $ntree0$, and number of features $Mtry0$. Secondly, the fitness function (imputation error) was computed by applying these numbers, then it was evaluated for other solutions during the optimization process. The fitness function above was called in GA function to be optimized and produce the optimum results. The algorithm for GA is

presented in Table 4.2 (Elyan & Gaber, 2017):

Table 4.2: Genetic Algorithm Process

begin

$iter \leftarrow 0$

$j \leftarrow 0$

$Generation_j \leftarrow K\ random\ solution$

$fitness \leftarrow compute\ fitness(k)\ \forall k \in Generation_j$

While $fitness\ not\ optimized\ and\ i \leq iter$ **do**

$Generation_{j+1} \leftarrow evolve(Generation_j)$

$fitness \leftarrow compute\ fitness(k)\ \forall k \in Generation_{j+1}$

$iter \leftarrow iter + 1$

end

return(fittest solution)

end

This algorithm called the fitness function from Table 4.1, then it searched over *ntree* and M_{try} values and reported the optimized numbers and their corresponding fitness values (imputation errors). The fittest solution was the minimum value for imputation errors which include both NRMSE (for continuous variables) and PFC (for categorical variables). As explained before there are different options for GA control parameters like population size, probability of crossover, and probability of mutation. Also, there are other parameters like elitism, maximum iteration, and run which can be set to default or some initial values to investigate the performance of GA. The parameters selected for this study is depicted in Table 4.3:

Table 4.3: GA parameters for optimizing MissForest

Parameters	**value**
Population Size	50
Pcrossover	0.8
Pmutation	0.1
Elitism	0.05
Maxiteration	20
Run	20

The package MissForest in R was applied to produce the results. In the process of optimization, GA was applied to search over the range of values $ntree$ and M_{try} to minimize the imputation error of MissForest and the same time impute the missing values. Then, the minimized imputation error was compared to the imputation error of the classic MissForest with default values $ntree$ and M_{try}. If the optimized MissForest had minimized the imputation error, the $ntree$ and M_{try} of the minimized error were reported as the optimal solutions. Since there were packages for both RF and GA in R, the optimization functions were defined and implemented in this environment.

2 Random Forest Optimization

One of the objective of this study was using the GA to optimize the performance of a RF. The goal was to find the optimal values for $ntree$ and M_{try} that minimize the MSE of the RF. The range of the initial population is $ntree \in [50, 1000]$ and $M_{try} \in [1, p]$ where p is the number of parameters. The range for number of trees is such that to make sure to have enough trees. Some studies have shown that number of trees beyond 500 to 1000 does not have a significant effect on improving the error of estimation, so the maximum number of tress was set to 1000 (Elyan & Gaber, 2017). Also, for a classic regression RF $ntree = 500$ and $M_{try} = p/3$. The GA control parameters to optimize the RF is presented in Table 4.4.

Table 4.4: GA parameters for optimizing RF

Parameters	**value**
Population Size	500
Pcrossover	0.8
Pmutation	0.1
Elitism	0.05
Maxiteration	500
Run	300

The fitness function algorithm for a RF optimization is the same as MissForest except for the fitness function that changes from imputation error to MSE. The GA function minimizes this fitness function and reports the optimized $ntree$ and M_{try}.

In the optimization process of a RF, the function GA was applied to minimize the MSE of the RF to produce the optimal solutions for $ntree$ and M_{try}. In order to optimize a Random Forest, the datasets were divided into train, test, and validation sets by 50%, 25%, 25%. The training set was used to apply GA to optimize the RF during the optimization process. The validation set was used to evaluate the optimized model while selecting the GA parameters. The test set was used for assessment of the final model. All these procedures were defined and implemented in a function in R using RF and GA packages.

3 Genetic Algorithm

In this study, the Real-valued type of Genetic Algorithm was applied to optimize the performance of the RF and MissRF. The GA was applied to minimize the validation MSE of a RF and imputation errors (NRMSE and PFC) of a MissForest and return their corresponding optimal integer value of $ntree$ and M_{try}. The GA function in R is applied for maximization of the fitness functions. In this study the goal is to minimize the estimation error, so a minus sign was added to the fitness function for minimization. As mentioned in section 3.2 there are several options for GA operators like selection, crossover, and mutation, but for this study the default settings were applied for both models which are

depicted in Table 4.5.

Table 4.5: GA arguments for RF and MissRF

GA Arguments	**default options**
Population	range [min,max]
Selection	gareal_lsSelection
Crossover	gareal_laCrossover
Mutation	gareal_raMutation

The population refers to the initial population that is generated randomly from the range of the solutions. For instance for $ntree \in [50, 1000]$ in a RF, the initial population is selected randomly from this range. The size of the population is determined in Table 4.4 (500). The default options for selection, crossover, and mutation are linear scaling, local arithmetic crossover, and uniform random mutation respectively. In order to evaluate the results for other options, one dataset (Automobile) was selected and the results were obtained and compared with default values which will be reported in next section.

Chapter 5

Experiments

This section presents the experiments that were conducted to compare the performance of optimized models with classic types. The optimized regression RF is called RRFGA and the optimized version of MissForest is MissRFGA. The goal of the experiments was to show that the optimized version of the models outperforms the classic types. In order to have a stable and valid conclusion, the experiments were replicated 10 times for all the datasets by using different seed numbers for RF and MissForest functions.

1 Datasets

There are five datasets that were selected from the UCI repository for the analysis. Two datasets contained missing values, and for the remaining three datasets, 10% of them were converted to missing values, which are considered missing completely at random. All the datasets and their descriptions are presented in Table 5.1.

Table 5.1: Description of the datasets used in the experiments

Dataset	Size	Variables No	Continuous variables No	Categorical variables No	Missing values
Concrete Compressive Strength	1030	9	9	0	No
Automobile Data	205	26	15	11	YES
Auto MPG Data Set	398	9	5	4	YES
Student Performance	649	31	4	27	No
Computer Hardware Data Set	209	8	7	1	No

For the first dataset "Concrete compressive strength dataset", the response, "Concrete strength", is predicted by 8 predictors, and this dataset does not include any missing values. The second dataset is the Automobile data where the price of the automobile is predicted by 25 predictors. This dataset includes missing values, so at first they were imputed by MissForets and then the RF was used for the prediction. The third dataset measures mile per gallon gas consumption of the automobiles based on eight variables. In the fourth dataset, students' final grades were predicted by 30 variables as predictors. The grades range was between 0 and 20 and it was a continuous variable. For the final dataset the performance of a computer hardware was measured by seven variables. This dataset included nine variables, but since one of them (model name) had more than 51 categories and since a RF could not handle this type of variable, it was removed from the dataset.

2 Results

In this section the results of the optimized and classic MissForest and RF are presented. The MissForest and RF were optimized by GA. There was a comparison between optimized MissForest and classic MissForest to evaluate their performance by comparing their imputation errors. Also, the performance of optimized RF and the classic RF were compared to investigate which model produce the minimum MSE.

These models were applied on five datasets from the UCI repository website. Firstly,

during the optimization process the final model which is the optimized one was selected, and the corresponding optimal solutions for $ntree$ and M_{try} were obtained. It has to be considered that one optimal number was obtained from the optimization process. After receiving optimal numbers, they were plugged in the MissForest and RF functions as initial values and the imputation error and MSE were computed respectively. Then, the imputation error and MSE of the optimal numbers were compared to the imputation error and MSE of the classic MissForest and RF using default values for $ntree$ and M_{try}. The comparison of the MSE of classic RF and optimized RF and classic MissForest and optimized MissForest were replicated 10 times with different seed numbers for both RF and MissForest.

The 10 replications for comparing the estimation error of the RF and MissForest have been conducted to evaluated if the optimal numbers for $ntree$ and M_{try} from optimization process produce minimum error comparing to the default values of these parameters in RF and MissForest. When 10 replications were conducted, each time the MSE of the RF and imputation error of the MissForest for both optimized and default $ntree$ and M_{try} were recorded and the average and standard deviation of 10 repetitions were computed for each dataset. The processing time to run the experiments for optimization of MissForest and RF was between 7 to 9 hours.

2.1 MissForest

The results of the average and standard error of the imputation errors for the 10 replicates of both optimized and classic MissForests are depicted in Table 5.2.

Table 5.2: Comparing imputation error of optimized and classic MissForest

Datasets	$ntree$	M_{try}	$MissRF_{avg}$	$MissRFGA_{avg}$	$MissRF_{std}$	$MissRFGA_{std}$
Concrete (NRMSE)	88	6	0.066	0.063	0.0004	0.0005
Automobile (Price/NRMSE)	194	10	0.131	0.124	0.0025	0.0013
Automobile (MPG/NRMSE)	285	5	0.086	0.085	0.0013	0.0006
Automobile (MPG/PFC)	285	5	0.277	0.288	0.0060	0.0079
Students performance(NRMSE)	172	17	0.317	0.305	0.0024	0.0023
Students performance(PFC)	172	17	0.437	0.444	0.0028	0.0018
Cmputer Hardware (NRMSE)	251	5	0.377	0.354	0.0061	0.0053

The values for $ntree$ and M_{try} in Table 5.2 are the values that were obtained from the optimization process. The column $missRF_{avg}$ represents the average imputation error of the classic MissForest using the default values for $ntree$ and M_{try} which are 100 and $\sqrt{p}$ respectively, where p is the number of predictors. The column $missRFGA_{avg}$ represents the average imputation error of the optimized MissForest using the optimal numbers for $ntree$ and M_{try} respectively. The last two columns represent the standard deviation of the imputation error of the classic and optimized MissForest. This table represents the impact of applying GA as an optimization method on the performance of MissForest. It shows that especially when missing values are present for continuous variables, the corresponding imputation error (NRMSE) produces better results than the classic type. The last column in Table 5.3 shows the percentage of improvement of the optimized MissForest over the classic one.

Table 5.3: Experimental Results of imputation error for MissForest

Datasets	ntree	M_{try}	$MissRF_{avg}$	$MissRFGA_{avg}$	percentage of improvement for the optimized model
Concrete	88	6	0.066	0.063	5%
Automobile (Price)	194	10	0.131	0.124	5%
Automobile (MPG/NRMSE)	285	5	0.086	0.085	1%
Automobile (MPG/PFC)	285	5	0.277	0.288	-4%
Students performance(NRMSE)	172	17	0.317	0.306	4%
Students performance(PFC)	172	17	0.437	0.444	-2%
Cmputer Hardware	251	5	0.377	0.354	6%

The optimal *ntree* and M_{try} are presented in Table 5.3. The columns $MissRF_{avg}$ and $MissRFGA_{avg}$ represent the average of imputation error (NRMSE). The last column provides the percentage of improvement in minimizing the (NRMSE) using optimized MissForest by computing the percentage in improvement $(\frac{MissRF_{avg}}{MissRFGA_{avg}} - 1)\%$. As it can be noticed by looking at Table 5.3 the optimized MissForest had a better performance than the classic model especially for three datasets that contain missing values just for continuous variables. However, there are some variations in its performance when missing values are present for both continuous and categorical variables. In this case, the imputation error of the MissForest function includes both NRMSE and PFC, the imputation error for continuous and categorical variables respectively.

When missing values were present for both categorical and continuous variables, the GA that was applied to optimize MissForest did not minimize both imputation errors at the same time and as a result did not produce one optimal solutions for *ntree* and M_{try}. Therefore, the GA optimizes the MissForest and produces two optimal solutions for them: one that minimizes NRMSE but not the corresponding PFC error. The other solution minimizes PFC but not the corresponding NRMSE. In this case the average of the both NRMSE and PFC is computed and the solutions (*ntree* and M_{try}) are selected whose average is smaller. The Table 5.4 shows the two sets of solutions that was obtained from the optimized MissForest for Automobile MPG dataset. The last column shows the average of the imputation error two solutions.

Table 5.4: Two sets of solutions for optimized MissForest

ntree	M_{try}	NRMSE	PFC	average of imputation errors
285	5	0.075	0.225	0.15
350	6	0.098	0.275	0.187

As it can be noticed from Table 5.4 by comparing the average in the last column, the first solution was selected to produce the 10 times replications of the results for MissForest. Figure 5.1 represents the box plot of the 10 replicates of the datasets for MissRF and its corresponding optimized versions. This Figure represents the 10 replications of optimized

and classic MissForest for NRMSE (imputation error for continuous variables) in the boxplots for five datasets.

Figure 5.1: Boxplot for 10 replicates of five datasets for MissForest

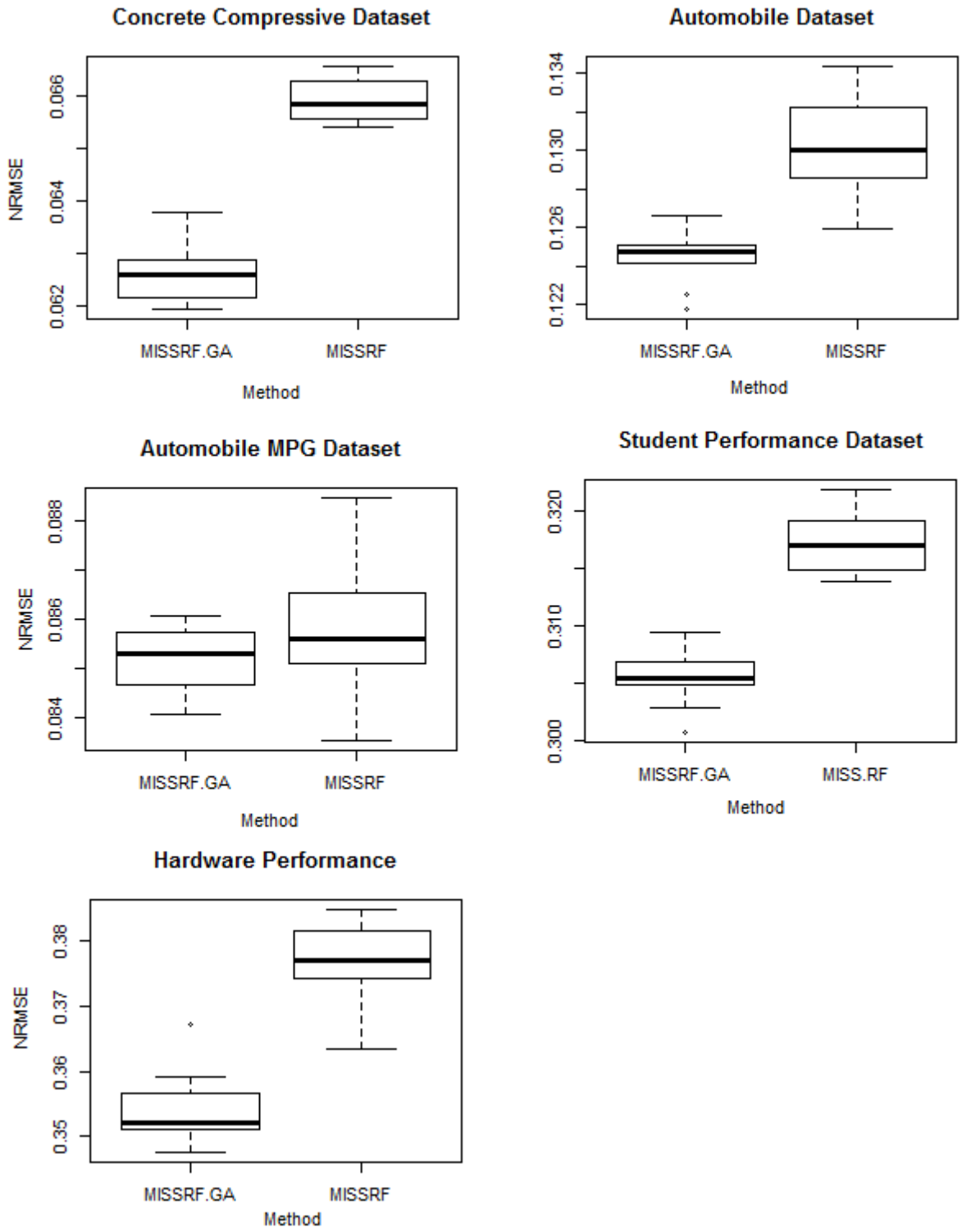


Figure 5.1 depicts that the performance of MissRFGA is better than the classic Miss-

Forest for almost all datasets. Three datasets including concrete, student performance, and computer hardware performance which showed a better performance did not include missing values, so 10% of them were converted to missing values completely at random. (Misztal, 2013) showed that MissForest has a better performance for MAR and MCAR. Therefore, it could be concluded that better results for these datasets was due to having MCAR missing values. However, Automobile and Automobile MPG datasets included missing values and still showed a better performance for MissRFGA.

2.2 RF

Table 5.5 represents the average and standard deviation of MSE of 10 replications of running the RF function for both optimized and classic RF. The solutions of the optimized RF were *ntree* and M_{try} which were depicted in Table 5.5. The default values for classic RF were $ntree = 500$ and $M_{try} = p/3$. The optimized and default values of these parameters are plugged in the RF function and the results of 10 replications were obtained and depicted below.

Table 5.5: Comparing Performance of Optimized and classic RF

Datasets	*ntree*	M_{try}	RF_{avg}	$RRFGA_{avg}$	RF_{std}	$RRFGA_{std}$
Concrete	616	6	25.789	20.961	0.223	0.268
Automobile (Price)	388	4	3,569,261	3,530,232	49250	48340
Automibile (MPG)	285	5	8.317	8.088	0.085	0.072
Students performance	485	16	15.849	15.577	0.038	0.095
Cmputer Hardware	452	4	3292.497	3013.604	130	132

The values for *ntree* and M_{try} in Table 5.5 are the values that were obtained from the optimization process. The column RF_{avg} represents the average MSE of the classic RF using the default values for *ntree* and M_{try} which are 500 and $\frac{p}{3}$ respectively, where p is the number of predictors. The column $RFGA_{avg}$ represents the average MSE of the optimized RF using the optimal numbers for *ntree* and M_{try} respectively. The last two

columns represent the standard deviation of the MSE of the classic and optimized RF. Table 5.5 shows that optimized RF had a great impact on first and last datasets while it also performed relatively well on other datasets. Table 5.6 represents how much in percentage the optimized RF outperformed a classic RF.

Table 5.6: Experimental Results for Random Forest

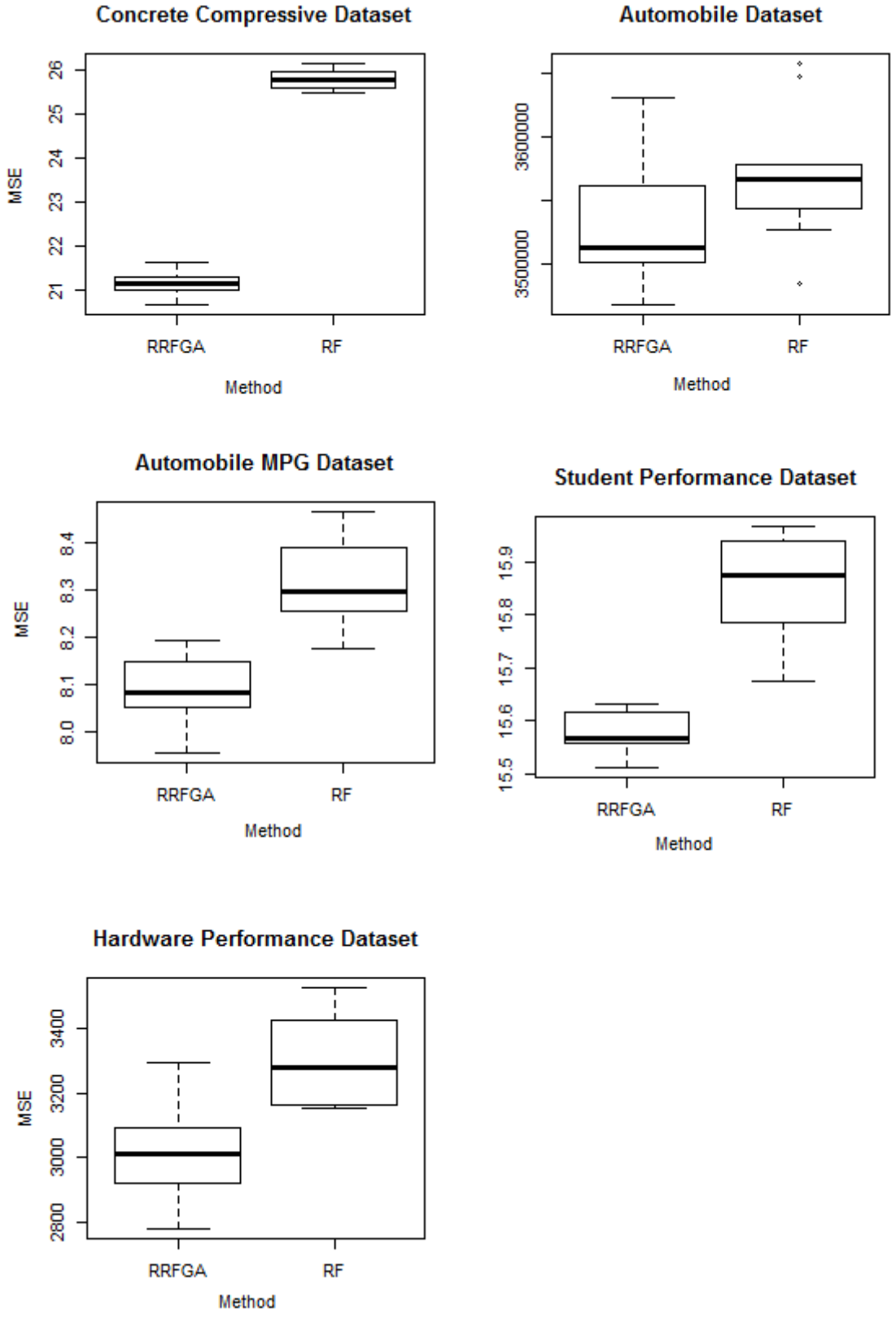
Datasets	ntree	M_{try}	RF_{avg}	$RFGA_{avg}$	percentage of improvement for optimized RF
Concrete	616	6	25.789	20.961	23%
Automobile (Price)	388	4	3,569,261	3,530,232	1%
Automobile(MPG)	285	5	8.317	8.088	3%
Students performance	485	16	15.849	15.577	2%
Cmputer Hardware	452	4	3293	3013	9%

The optimal *ntree* and M_{try} are presented in Table 5.6. The columns RF_{avg} and $RFGA_{avg}$ represent the average of MSE of the estimation. The last column provides the percentage of improvement in minimizing the MSE using optimized RF by computing the percentage in improvement $(\frac{RF_{avg}}{RFGA_{avg}} - 1)\%$.

The percentages in Table 5.6 show a great performance for the Concrete dataset and for computer hardware dataset. By looking at these two datasets it appears that the former did not include any categorical variable and the number of observations is large comparing to the other datasets. The hardware dataset also includes only one discrete variable as a predictor which shows a relatively good performance.

Figure 5.2 shows the distribution of the 10 replicates of the results for optimized and classic RF. As it can be observed the performance of the optimized model is better than classic RF especially for the Concrete dataset where the MSE for RRFGA is much lower than for RF. Also, in most cases the distribution for RRFGA is skewed to the left which represents the lower amounts for the MSE of the model. The boxplot for the 10 replicates of the MSE of the RF is represented in Figure 5.2.

Figure 5.2: Boxplot for 10 replicates of five datasets for Random Forest



The results in Figure 5.2 for the automobile dataset show some outliers for the classic RF on the two extremes, while it represents relatively higher values of the MSE comparing to the results of RRFGA. Also the results for MSE of RRFGA for this dataset is pretty close to the RF. The dataset student performance contains both categorical and continuous variables, and even if on average it does not show a great performance, it shows a better performance for the replicates which could be due to the larger number of observations after Concrete dataset.

Since there are different options for GA operators including selection, crossover, and mutation, some of these options were applied for the analysis of Automobile dataset, and the result of 10 replications was reported and compared to the default setting (Table 4.5). The results of these options are depicted in Table 5.7.

Table 5.7: Different GA operators for selection, crossover, mutation

GA Operators:slection/crossover/mutation	RRFGAavg	increase in average MSE compare to default(%)
lsSelection,laCrossover,raMutation (default)	3,530,232	
nlrSelection,laCrossover,raMutation	3,879,156	10%
lrSelection,waCrossover,nraMutation	3,863,449	9%
nlrSelection,spCrossover,raMutation	3,858,709	9%
nlrSelection,blxCrossoverraMutation	3,854,262	9%
rwSelection,spCrossover,raMutation	3,822,033	8%
nlrSelection,laCrossover,raMutation	3,798,243	8%
nlrSelection,waCrossover,raMutation	3,710,612	5%
rwSelection,laCrossover,raMutation	3,621,036	3%
lrSelection,blxCrossover,nraMutation	3,590,870	2%
lrSelection,spCrossover,nraMutation	3,586,734	2%
lrSelection,spCrossover,nraMutation	3,586,734	2%
rwSelection,blxCrossover,raMutation	3,576,262	1%
rwSelection,waCrossover,raMutation	3,559,247	1%

The first result on the top of the table was the default setting. This method produced the lowest average and standard error among all available options. The average MSE of the optimized RF using other operators in GA was compared to the default operators on the top in Table 4.4. It was evaluated to what extend (in terms of percentage) using other operators of GA will increase the average MSE of the RF $(\frac{equivalent\ operator}{default\ operator} - 1)\%$. This occurrence could be random, so this process could be repeated for other datasets to see if it holds for other types of data. Different GA operators could be examined for RF optimization and the one that produces the lowest minimum MSE could be selected.

In last part of section 3, it was discussed that the performance of GA depends on the GA operators: selection, crossover, and mutation. Also, some studies showed the effect of GA control parameters like the probability of crossover and mutation on its performance. These studies suggested some numbers especially for population size, probability of crossover, and probability of mutation. Some values were selected from the suggested ranges and plugged in the GA function which was applied to optimize the RF function.

These parameters for GA were applied for RF optimization, and the solutions which were $ntree$ and M_{try} obtained. Then, 10 replications were conducted for the automobile dataset, and the average of the results was reported in Table 5.8. The first column depicted the parameters respectively: $ntree$, M_{try}, population size, probability of crossover, and probability of mutation. The first record on the top represents the control parameter suggested in table 4.4 that was used for all the datasets.

Table 5.8: Comparing different GA control parameters of RRFGA on Automobile dataset

RRFGA parameters	MSE_{avg}	increase in average MSE comparing to default (%)
388-4-500-0.9-0.1	3530232	
498-6-50-0.6-0.001	3634526	3%
557-10-100-0.6-0.001	3531811	0.04%
498-4-30-0.95-0.01	3931844	11%
498-4-30-0.95-0.005	3931844	11%
496-13-30-0.75-0.01	3575685	1%
496-13-30-0.75-0.005	3575685	1%
513-12-20-0.95-0.005	3553703	0.7%
513-12-20-0.95-0.01	3553703	0.7%
530-12-20-0.75-0.005	3532588	0.07%
530-12-20-0.75-0.01	3532588	0.07%

Table 5.8 represents the average MSE of RF using different GA control parameters. The first row is the proposed numbers in Table 4.4 to run the GA. The average MSE of the optimized RF using other control parameters in GA was compared to the suggested ones in Table 4.4. It was evaluated to what extend (in terms of percentage) using other control parameters of GA will increase the average MSE of the RF $(\frac{equivalent\ option}{suggested\ option} - 1)\%$.

The results show that the GA operators suggested before in Table 4.4, which is the top row in Table 5.8, outperformed the other numbers depicted in this table. However, as it can be observed, in most cases the increase in the MSE of the RF is not noticeable, except for two cases which show a growth of 11% comparing to the default GA parameters represented on the first row. Also, some of the suggested numbers for population size, probability of crossover and mutation produced the same numbers for *ntree* and M_{try} which can be seen on the 4^{th} record and later. All these cases have to be examined on the other datasets to evaluate if different GA parameters on other datasets produce a higher MSE comparing to the default numbers. In addition, since for some of the cases

the difference between the average MSE for default and suggested numbers is very small especially when the population size is involved, it could be suggested using the proposed options with smaller population size to reduce the processing time.

3 Conclusions and Future Work

In this study the main objective was to optimize regression RF and MissForest by applying GA. The goal was to minimize the MSE of RF and imputation errors of MissForest by obtaining the optimal numbers for $ntree$ and M_{try}. Since the search space was very large, the fitness function was complex, and the goal was to look for optimal numbers, GA was applied. As it was observed the proposed models outperformed the classic types. Although the impact of optimization on RF was more remarkable than MissForest.

This study coordinates with two other studies that had been conducted by Bader-El-Den and Gaber (2012) and Elyan and Gaber (2017) where they used GA to optimize the performance of Classification RF. Bader-El-Den and Gaber (2012) applied single point crossover with uniform mutation as operators. The population size was either one of these choices, 100, 200, 400, 500 and the probability of crossover and mutation were set to 0.9 and 0.1 respectively. The results showed the optimized model GARF had a better performance on 8 out of 15 datasets. Elyan and Gaber (2017) applied GA to optimize the classification RF by obtaining the optimized values for sub-classes of the response, $ntree$ and M_{try}. The parameter setting for GA in their study was the same as in Table 4.4 for this study. They obtained 3% improvement in the performance of RF using GA. According to my knowledge since there are not many studies available related to improving the performance of regression RF, in this study it was evaluated if GA could improve its performance. The study was conducted on five datasets, and in all of them the optimized methods, RRFGA and MissRFGA, outperformed RF and MissForest. The optimized MissForest also had a better performance especially when just continuous variables were present.

In spite of noticing some improvement in the proposed methods specially for RF, it has to be considered that among all the combinations for GA control parameters, one

approach (Table 4.4) was studied. Also, for a RF some other parameters such as node size, size of the tree, or size of the bootstrap samples could be evaluated to be regarded as parameters.

The results of various GA operators were presented in Table 5.8, and it was observed that the default operators had a better performance for Automobile dataset. Also, the results of GA control parameters presented in Table 5.8 showed a better performance for the options depicted in Table 4.4. Also, one important issue in this study is the time of the analysis. The average time for the optimization process was between 7 to 9 hours, however, for some of the GA control parameters in Table 5.8, it was reduced on average to 30 minutes.

Regarding all the issues discussed above, it is suggested that for a future study, the effect of all the available GA control parameters on various types of datasets be evaluated to find the one that could produce optimal solutions. Also, the effect of GA operators on different datasets could be investigated to find one operator that produce the optimal results. It also can be studied how to have a trade-off between time of the process and producing an accurate result. In addition, since proposed method produce two sets of solutions for MissForest, it could be assessed if there is a possibility to minimize both errors at the same time to produce one set of solution.

R codes

```
library(randomForest)
library(MASS)
library(ROCR)
library(party)
library(GA)
library(missForest)


/*RRFGA and Classic RF*/

setwd("D:/courses/Thesis/for data set/other data set/datasets")
ds=read.csv("Concrete.csv")

set.seed(23568)
n = nrow(ds)
trainIndex = sample(1:n, size = round(0.8*n), replace=FALSE)
train = ds[trainIndex ,]
test = ds[-trainIndex ,]
n1=nrow(train)
vindex=sample(1:n1, size = round(0.75*n1), replace=FALSE)
train1=train[vindex,]
validation=train[-vindex,]
```

```
cgarf<-function(ntrees,mtrys){
ntrees<-50
mtrys<-4
set.seed(23568)
n = nrow(ds)
trainIndex = sample(1:n, size = round(0.8*n), replace=FALSE)
train = ds[trainIndex ,]
test = ds[-trainIndex ,]
n1=nrow(train)
vindex=sample(1:n1, size = round(0.75*n1), replace=FALSE)
subtrain=train[vindex,]
subtest=train[-vindex,]
rf<-randomForest(subtrain[,-9],subtrain[,9],xtest=subtest[,-9],
ytest=subtest[,9],ntree = ntrees,mtry=mtrys,proximity=TRUE,importance = TRU
err.g<-rf$test$mse[ntrees]
pred<-rf$test$predicted
return(err.g)
}

RRFGA<-function(ds,mintrees=100,maxtrees=1000,minfeatures=1/(ncol(ds)-1),
maxfeatures=1,popSize=50,maxIter=10,runs=10){



set.seed(23568)
n = nrow(ds)
trainIndex = sample(1:n, size = round(0.8*n), replace=FALSE)
train = ds[trainIndex ,]
test = ds[-trainIndex ,]
```

```
n1=nrow(train)
vindex=sample(1:n1, size = round(0.75*n1), replace=FALSE)
subtrain=train[vindex,]
subtest=train[-vindex,]

ntrees1<-500
nfeatures<-ncol(ds)-1

set.seed(4578)
rf<-randomForest(subtrain[,-9],subtrain[,9],xtest=test[,-9],
ytest=test[,9],ntree = ntrees1,proximity=TRUE,importance = TRUE)
err.r<-rf$test$mse[ntrees1]

minc<-c(mintrees,(minfeatures*nfeatures))
maxc<-c(maxtrees,(maxfeatures*nfeatures))

GA<-ga(type="real-valued",fitness=cgarf,
min=minc,max=maxc,keepBest=TRUE,popSize=popSize,maxiter=maxIter,run=runs,se
names=c("ntrees","mtry"))

testResults<-data.frame()
for(i in 1:nrow(GA@solution)){
ntrees<-as.integer(round(GA@solution[i,1]))
mtrys<-as.integer(round(GA@solution[i,2]))


rfg<-randomForest(subtrain[,-9],subtrain[,9],xtest=subtest[,-9],
ytest=subtest[,9],ntree = ntrees,mtry=mtrys,proximity=TRUE,importance = TRU
err.g<-rfg$test$mse[ntrees]
```

```
#pred<-rf$test$predicted
write.csv(pred,"prd.csv")
testResults<-rbind(testResults,c(ntrees,mtrys,err.g,err.r))
fp<-paste0("results/",names(ds[ncol(ds)]),"preddf.csv")
write.csv(pred,file=fp)
}
f1<-paste0 ("results/",names(ds[ncol(ds)]),".csv")
f2<-paste0 ("results/",names(ds[ncol(ds)]),"_testResults","_",".csv")
write.csv(GA@solution,file=f1)
names(testResults)<-c("ntrees","mtrys","err.g","err.r")
write.csv(testResults,file=f2)

return(GA)

}
start.time <- Sys.time()
RRFGA(ds,100,1000,1/(ncol(ds)-1),1,500,500,300)
end.time <- Sys.time()
rf<-randomForest(ds[,-9],ds[,9],importance = TRUE,ntree=616,mtry=6,do.trace
rf1<-randomForest(ds[,-9],ds[,9],importance = TRUE,do.trace = 100)



\newpage
/*MissRFGA and MissForest*/
setwd("D:/courses/Thesis/for data set/other data set/datasets")
ds=read.csv("Concrete.csv")

concrete.mis <- prodNA(ds1, noNA = 0.1)
write.csv(concrete.mis, file = "conmis.csv")
```

```
ds1=read.csv("conmis.csv")


gamiss<-function(ntrees,mtrys){
ntrees<-50
mtrys<-4
ga.imp <- missForest(ds,ntree=ntrees,mtry=mtrys)
err.g<-ga.imp$OOBerror
pred<-ga.imp$ximp
return(err.g)
}


MissRFGA<-function(concrete.mis,mintrees=50,maxtrees=200,minfeatures=1/(nco
maxfeatures=1,popSize=30,maxIter=20,runs=10){


ntrees1<-500
nfeatures<-ncol(ds1)-1

concrete.imp <- missForest(ds1)
err.r<-concrete.imp$OOBerror
obs.d<-concrete.imp$ximp


minc<-c(mintrees,(minfeatures*nfeatures))
maxc<-c(maxtrees,(maxfeatures*nfeatures))

GA<-ga(type="real-valued",fitness=gamiss,
```

```
min=minc,max=maxc,keepBest=TRUE,popSize=popSize,maxiter=maxIter,run=runs,
names=c("ntrees","mtry"))

testResults<-data.frame()
for(i in 1:nrow(GA@solution)){
ntrees<-as.integer(round(GA@solution[i,1]))
mtrys<-as.integer(round(GA@solution[i,2]))

ga.imp <- missForest(ds1,ntree=ntrees,mtry=mtrys)
err.g<-ga.imp$OOBerror
pred<-ga.imp$ximp

testResults<-rbind(testResults,c(ntrees,mtrys,err.g,err.d))
fp<-paste0("missing/",names(ds1[ncol(ds1)]),"predmiss.csv")
write.csv(pred,file=fp)
fp1<-paste0("missing/",names(ds1[ncol(ds1)]),"defmiss.csv")
write.csv(obs.d,file=fp1)

}
f1<-paste0 ("missing/",names(ds1[ncol(ds1)]),".csv")
f2<-paste0 ("missing/",names(ds1[ncol(ds1)]),"_testResults","_",".csv")
write.csv(GA@solution,file=f1)
names(testResults)<-c("ntrees","mtrys","err.g","err.r")
write.csv(testResults,file=f2)

return(GA)

}
start.time <- Sys.time()
```

```
MissRFGA(con.miss,50,500,1/(ncol(ds1)-1),1,50,20,20)
end.time <- Sys.time()

con.imp <- missForest(ds1,ntree=88,mtry=6)
err<-con.imp$OOBerror

con.impp <- missForest(ds1)
err<-con.impp$OOBerror
```

References

Adewuya, A. A. (1996). *New methods in genetic search with real-valued chromosomes* (Unpublished doctoral dissertation). Massachusetts Institute of Technology.

Bader-El-Den, M., & Gaber, M. (2012). Garf: towards self-optimised random forests. In *International conference on neural information processing* (pp. 506–515).

Bernard, S., Heutte, L., & Adam, S. (2008). Forest-rk: A new random forest induction method. In *International conference on intelligent computing.*

Biau, G., & Scornet, E. (2016). A random forest guided tour. *Test*, *25*(2), 197–227.

Blickle, T., & Thiele, L. (1995). A comparison of selection schemes used in genetic algorithms. *The Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich*(11).

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Breiman, L., et al. (2001). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, *16*(3), 199–231.

Carr, J. (2014). An introduction to genetic algorithms. *Senior Project*, *1*, 40.

Cutler, A., Cutler, D. R., & Stevens, J. R. (2012). Random forests. In *Ensemble machine learning* (pp. 157–175). Springer.

Efron, B. (1979). Bootstrap methods: another look at the jackknife. *The Annals of Statistics*, *7*(1), 1–26.

Elyan, E., & Gaber, M. M. (2017). A genetic algorithm approach to optimising random forests applied to class engineered data. *Information sciences*, *384*, 220–234.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning*

(Vol. 1). Springer series in statistics New York.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine learning*, *3*(2), 95–99.

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on systems, man, and cybernetics*, *16*(1), 122–128.

Haupt, R. L., & Haupt, S. E. (2004). *Practical genetic algorithms*. John Wiley & Sons.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112). Springer.

Jebari, K., & Madiafi, M. (2013). Selection methods for genetic algorithms. *International Journal of Emerging Sciences*, *3*(4), 333–344.

Kumar, R. (2012, 08). Blending roulette wheel selection and rank selection in genetic algorithms. *International Journal of Machine Learning and Computing*, *2*(4), 365-370.

Latinne, P., Debeir, O., & Decaestecker, C. (2001). Limiting the number of trees in random forests. In *International workshop on multiple classifier systems* (pp. 178–187).

Louppe, G. (2014). Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*.

Misztal, M. (2013). Some remarks on the data imputation using "missforest" method. *Folia Oeconomica*, *285*, 169–179.

Oladele, R. O., & Sadiku, J. S. (2013, May). Article: Genetic algorithm performance with different selection methods in solving multi-objective network design problem. *International Journal of Computer Applications*, *70*(12), 5–9.

Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012). How many trees in a random forest? In *Machine learning and data mining in pattern recognition* (pp. 178–187).

Patil, V., & Pawar, D. (2015). The optimal crossover or mutation rates in genetic algorithm: A review. *International Journal of Applied Engineering and Technology*, *5*(3), 38–41.

Razali, N. M., Geraghty, J., et al. (2011). Genetic algorithm performance with different selection strategies in solving tsp. , *2*, 1134–1139.

Samuel, A. (1959). some studies in machine learning using the game of checkers. *journal of research and development*, *3*(3), 210–299.

Schaffer, J., Caruana, R., Eshelman, L., & Das, R. (1989, 01). A study of control parameters affecting online performance of genetic algorithms for function optimization. , 51-60.

Stekhoven, D. J., & Bühlmann, P. (2011). Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, *28*(1), 112–118.

Tang, F. (2017). *Random forest missing data approaches* (Unpublished doctoral dissertation). University of Miami.

Young, J. (2017). Imputation for random forests. *All Graduate Plan B and other Reports*(994). `https://digitalcommons.usu.edu/gradreports/994`.