Final Project Report

Noam Moscovich 308353341
Yarden Rotem 305284515

August 15, 2020

Algorithm Name: Heterogeneous Genetic Algorithm based Random Forests

Reference: Bader-El-Den, Mohamed. "Self-adaptive heterogeneous random forest." 2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA). IEEE, 2014.

Motivation for the algorithm: The paper claims that random forest hyper-parameters have a significant impact on the performance of the algorithm. However, determining these hyperparameters may be a difficult task. Thus, the authors suggest a generic strategy, coined Heterogeneous Genetic Algorithm based Random Forests (HGARF) to optimize random forest in order to boost their performance.

Short Description: The algorithm starts by creating a large set of random forest of N decision trees each, generating a vector $\vec{RF}$. Unlike the classical Rnadom Forest which build the decision trees using the same M value (i.e., the number of features used at each split). Usually, M is set to square root of the number of features F. This paper claims that there is not defined way to select this value, and therefore it executes Random Forest with different values of M ranging from 2 to $F - 1$. Then, the algorithm randomly chooses decision trees from $\vec{RF}$. As a result, there are population of vectors $r\vec{f}_i$ which each vector (i.e., individual) contains set of decision trees in a size of the chromosome length. This initial population is then being manipulated through a number of generations, with the fitness function for each individual being its classification accuracy.

Pseudo-Codes:

Algorithm 1: HGARF Algorithm - Training

Input: $S\_train = (< x_1, y_1 >, \cdots, < x_m, y_m >)$ - Training Set
Input: $S\_val = (< x_1, y_1 >, \cdots, < x_m, y_m >)$ - Validation Set
Input: N - Number of decision trees
Input: S - Population's Size
Input: NG - Number of generations
Input: IS - Individual's size
Output: Best individual
$F \leftarrow number\_of\_features(S\_train)$
$\vec{RF} \leftarrow \emptyset$
for *i in range(2,F-1)* **do**
  $\vec{RF} \leftarrow \vec{RF} \cup random\_forest(\frac{N}{F-2}, i)$
end
for *i in range(1,S)* **do**
  $\vec{rf}_i \leftarrow random\_choose(\vec{RF}, IS)$
end
for *i in range(1,NG)* **do**
  $pop \leftarrow genetic\_algortihm(S\_val)$
  $\vec{rf}_b \leftarrow best\_individual(pop)$
end
return $\vec{rf}_b$

Algorithm 2: HGARF Algorithm - Inference

Input: X - A Sample
Input: $\vec{rf}$ - Set of Decision Trees
Output: C Class
$results \leftarrow []$
for *tree in* $\vec{rf}$ **do**
  $result \leftarrow tree.predict(X)$
  $results[result] \leftarrow results[result] + 1$
end
return $argmax(results)$

Algorithm Explanation: We divide the explanation into 3 different categories - Training, Inference and Genetic algorithm.

The training phase of the algorithm works as follows -

1. Extracting the number of features from the dataset.
2. Initializing empty set (coined RF).
3. Iterating over indexes 2 to $F-1$ which represent the number of features to execute random forest using $\frac{N}{F-2}$ decision trees, and adding the decision trees to RF.
4. Iterating over indexes 1 to S and building the relevant population by choosing randomly IS decision trees from RF.

5. Iteration over indexes 1 to NG which represent the number of generations to execute the genetic algorithm.
6. Extracting the best individual from the population of the last generation and return it.

The genetic algorithm properties are -

- Crossover – One-point Crossover.
  - Switch between the individuals in a given point.
  - In case there is a duplicate decision tree in the new individuals, remove it and choose randomly from RF.
- Mutation – Uniform Mutation.
  - Replace the genome of an individual with a given probability.
- Selection – Tournament.
  - Select the best individual among a given number of individuals.
- Fitness – Accuracy of the individual on the validation set.

In the inference phase, we return the class which most of the decision trees in the best individual classified to.

The training phase of the algorithm is presented in Algorithm 1 while the inference phase is presented in Algorithm 2.

Illustration: In order to illustrate the HGARF algorithm, we decided to evaluate and examines its performance on the breast cancer Wisconsin binary dataset. We execute HGARF with the following parameters -

Table 1: Illustration Parameters

Number of decision trees	40
Chromosome Length	5
Number of generations	50
Tournamet size	4
Crossover rate	0.5
Mutation rate	0.8

Because of the reason that our decision trees pool is calculated once (in the beginning of the algorithm), we can't illustrate the algorithm in the way presented in the example report. Therefore, we decided to illustrate the Algorithm using two different figures - (a) flowing chart (Figure 1) (b) traditional evolutionary algorithm presentation graph (Figure 2).

As can be noticed in Figure 2 -
(a) The best individual in the population has reached to 98% accuracy during the first generation.
(b) The average accuracy of the population has improved in the first generation, and then doesn't move.
(c) The worst individual in the population is not stable.

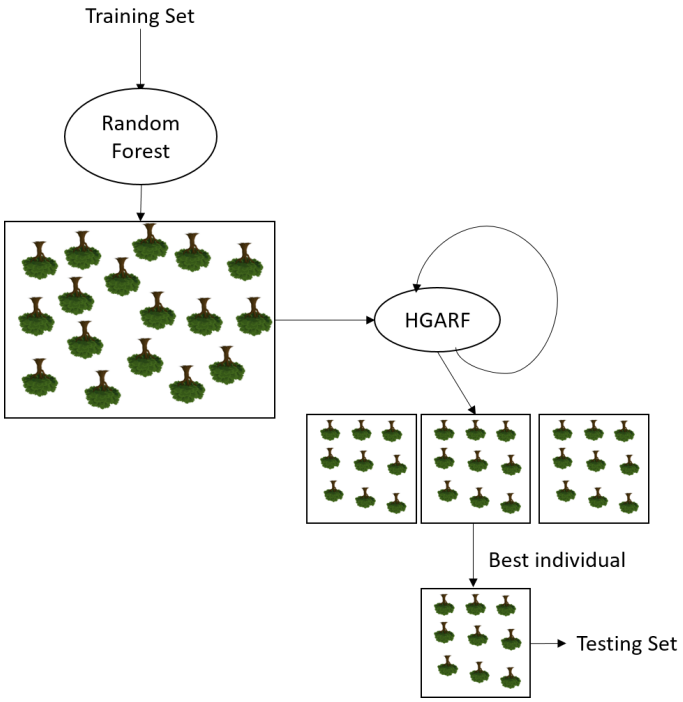

Figure 1: Illustration of HGARF

Strengths: In our opinion, the paper has multiple strengths -

1. Number of decision trees - The algorithm executes random forest for different number of features and trees. Thus, the set of decision trees is large which might lead to different individuals.
2. Straightforward - The algorithm is straightforward which combines basic concepts of bagging and evolution algorithms.

Drawbacks: In our opinion, the paper has multiple drawbacks -

1. Fitness function - The fitness function based on the accuracy of the forest, however, the diversity of the forest doesn't affect it. Adding the diversity to the fitness function might leads the convergence process to a accurate and diversity function.
2. Running time - In contrast to another bagging/boosting methods, the convergence process is relatively slow. Another examinations should be done in order to determine whether this issue based on implementation or algorithm issue.

Experimental Results: This section is divided into three subsections - Algorithm evaluation, Statistical analysis, and Meta-Learning. It should be mentioned that after discussing with Prof. Lior Rokach because of long running-times, he approved us to reduce (a) the number of datasets and (b) number of folds.

1. Algorithm evaluation - Attached in a CSV file.
2. Statistical analysis - To execute a Friedman test, we should handle at least three classifiers. In this project, we compare HGARF and AdaBoost; therefore, we can't perform

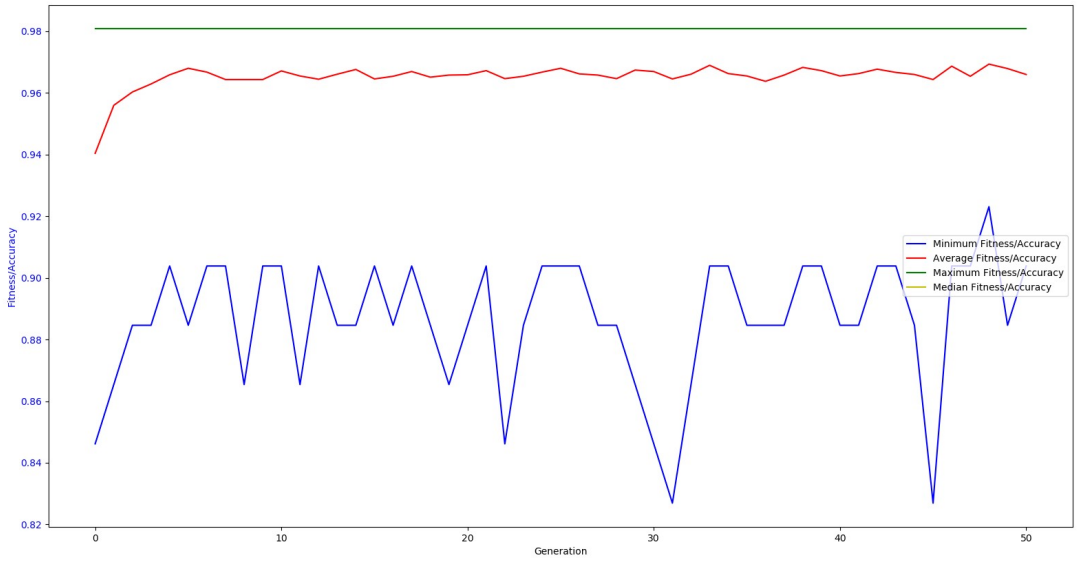

Figure 2: Toy dataset example

the Friedman test. To analyze the classifiers, we executed the Wilcoxon signed-rank test due to *Demšar, Janez. "Statistical comparisons of classifiers over multiple data sets." Journal of Machine learning research 7.Jan (2006): 1-30.* Our null hypothesis is that the related samples came from the same distribution (which means that the classifiers perform the same). The p-value of the test is 0.72, which means that we can't reject the null hypothesis, and therefore, we say that the classifiers' performance is equal.

Besides, to check the classifiers' training time, we calculate the same test with regards to the training time. In this test, the P-value is less than 0.05, which means that we can reject the null hypothesis (i.e., the classifiers' training times are equal) and claim that the AdaBoost algorithm is much faster than HGARF.

3. Meta-Learning - Attached in a CSV file, the shap values is presented in Figure 3.

Conclusions: In this paper, we introduce and implement HGARF, a decision-tree-based algorithm that improves itself by manipulating the population in an evolutionary algorithm manner. According to the results presented in this report, the accuracy of the algorithm is statistically equal to the accuracy of AdaBoost. However, the running time of the algorithm is much slower than AdaBoost. In future work, we think that implementing the algorithm in parallel (evaluating the population simultaneously) can dramatically improve the algorithm's training time. In addition, we believe that encapsulating a combination of algorithms than only decision trees (i.e., linear regression, SVM) can represent the data sets in different spaces, which may lead to better results. Besides, the connection between the hyper-parameters and the accuracy of the algorithm should be examined more in-depth.

Citations: The paper was cited multiple times; however, none of them reviewed its' advantages/disadvantages. Furthermore, in our opinion, most of the citations aimed at the regular random forest.

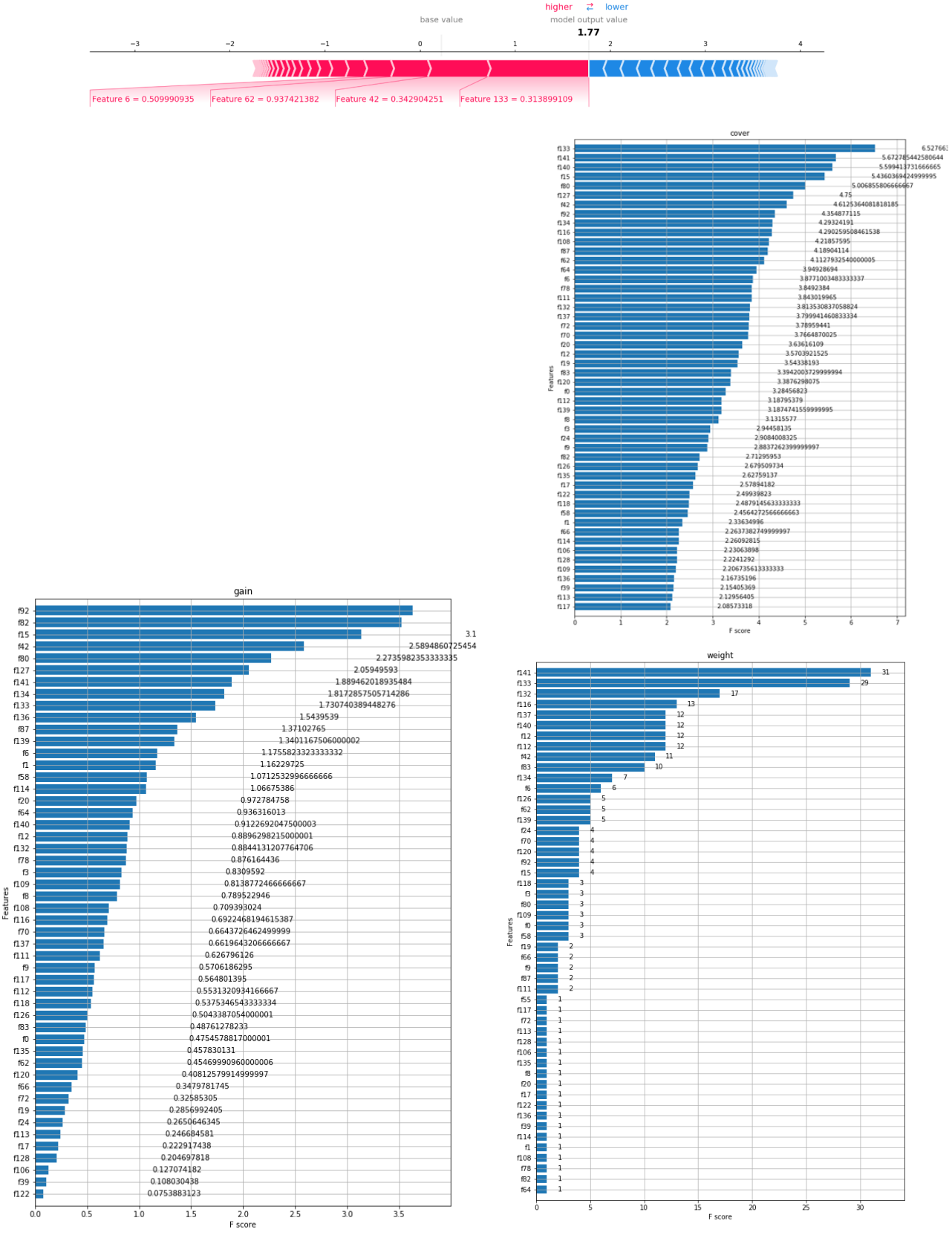

Figure 3: SHAP Values