Tree Ensemble approaches to a Regression Problem

Belloni Annalisa, Petrianni Valeria
Politecnico di Torino
Students id: [redacted], [redacted]
[redacted]@studenti.polito.it
[redacted]@studenti.polito.it

***Abstract*—In this report, we present a potential methodology to address the problem of predicting a pair of target numerical variables *(x, y)* that denote the coordinates where a particle of interest has traversed a sensor covered by 12 pads.**
Our proposed approaches, utilizing the two ensemble techniques *RandomForestRegressor* and *ExtraTreesRegressor*, demonstrate a significant improvement compared to the alternative methods considered.

I. Problem overview

The competition is centered around a regression problem in the field of particle physics. The goal is to predict the positions of particles (e.g., electrons) along their trajectories.
The input data for our problem were gathered using a Resistive Silicon Detector (RSD) sensor capable of detecting the passage of particles through it (a phenomenon referred to as an 'event'). Specifically, the sensor's two-dimensional surface is equipped with 12 *snowflake-shaped* metallic pads used to measure the signal.
The dataset is divided into two parts:

- A ***development*** set comprising 385,500 events. For each event, we have information on 5 properties related to the signals collected by each pad upon the passage of particles: *pmax*, *negpmax*, *tmax*, *area*, and *rms*. These represent, respectively, the magnitude of the positive peak of the signal (in mV), the magnitude of the negative peak of the signal (in mV), the delay (in ns) from a reference time when the positive peak occurs, the area under the signal, and the root mean square value of the signal.
  It's important to notice that for each event, there are 18 readings for each feature, despite the sensor having only 12 pads. This discrepancy arises due to hardware limitations during the data acquisition phase. Consequently, some of the 18 features include **noise** rather than actual readings.
  Clearly, the dataset also includes the pair of target numerical variables (x, y) (in μm) indicating the particle's position.
- An ***evaluation*** set containing 128,500 events, sharing the same structure as the *development* set, except for the two target variables that are missing, of course.

Examining the datasets, we can state that there are no missing values. Moreover, checking the domain constraints of the features, we noticed the existence of some records with invalid values in certain fields. During the preprocessing phase, we then proceeded to remove the corresponding events from the *development* dataset.
Additionally, in our preliminary examination of the relationships between variables, we determined that the targets *x* and *y* exhibit independence, as indicated by a correlation coefficient approximately equal to 0. So, the regression model can treat target variables independently during the learning process.

II. Proposed approach

A. Preprocessing

As mentioned earlier, we have removed 5 records from the *development* dataset that had incorrect values in the *negpmax* field, as they were positive.
Then, we considered it advisable to identify the noise variables among the inputs to work with more relevant information, reduce the dataset dimensions, and avoid overfitting. To accomplish this, we employed the PCA technique. For each pad $i = 0, ..., 17$ we grouped together the features *pmax[i]*, *negpmax[i]*, *area[i]*, *rms[i]* and *tmax[i]* and we extracted the first component from each group, which then became the feature representing the i-*th* pad. Subsequently, we computed the correlation between these 18 new features and the target variables *x* and *y*. The results can be seen in Figure 1.

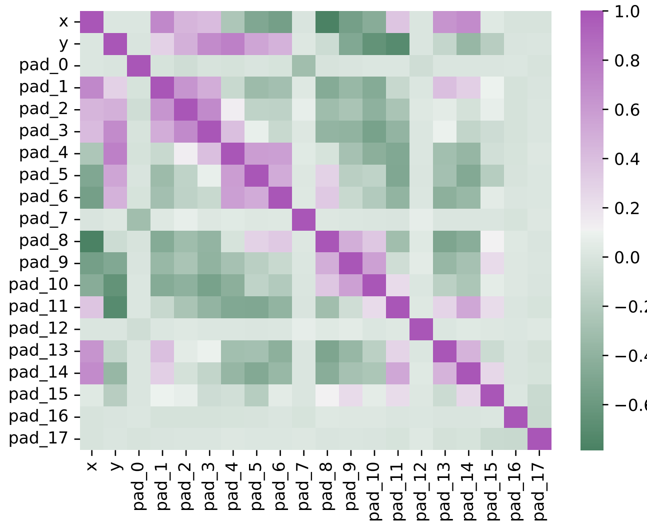

Fig. 1. Correlation between the target variables *(x,y)* and the pads features

Visually, it is quite clear that the features representing pads number 0, 7, 12, 15, 16, and 17 are not correlated with the pair of output variables *(x, y)*. Despite knowing that correlation identifies only a *linear* relationship between variables, and that our problem may not necessarily involve it, we have still embraced the previous observation, assuming that the discovered 6 are the noise features we've been looking for. This choice, like all the others, was backed by a tangible improvement in results, as will be explained in more detail shortly.

At this point, we have removed from the dataset all 30 noise features related to the 6 pads just identified, reducing the number of attributes from 90 to 60.

Following the removal of noise pads, we conducted a brief investigation to explore any correlation between the particles' coordinates and the features extracted from the signal captured by each remaining pad. In particular, we focused on the property *pmax*, plotting (for each pad) the trend of its values in relation to the distribution of data points in space. As depicted in Figure 2, we observed a strong relationship between this feature and the target variables. Notably, in each subplot we can see how the points associated with high values of pmax cluster in well-defined locations in space. This suggests that the magnitudes of the positive peak of the signals may play a significant role in predicting the particles' position.

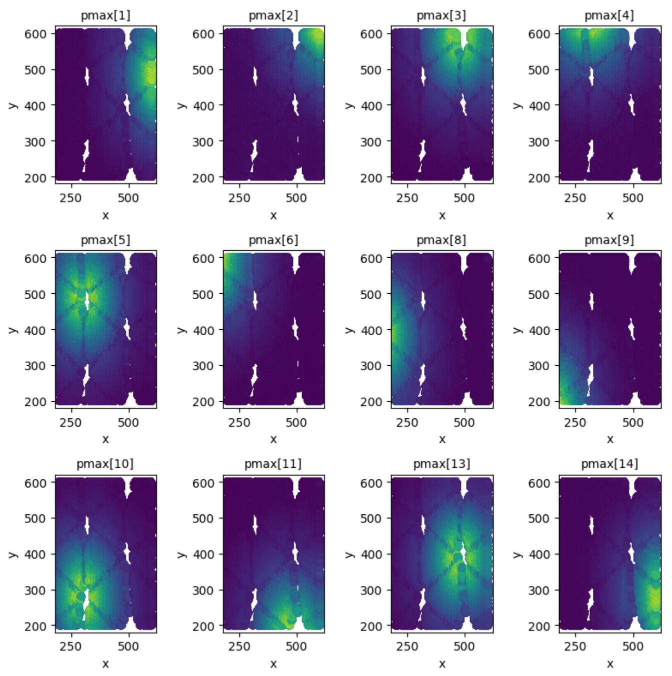

Fig. 2. Trend of *pmax[i]* in relation to the distribution of data points in space (for $i = 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 13, 14$). High values of the feature are represented by the colors green and yellow, whereas blue corresponds to lower values.

To obtain even more concise and meaningful input data, we considered leveraging the feature importance property, thus excluding another subset of features among the less relevant ones. To gain information about the importance of the remaining 60 attributes, we employed the *RandomForestRegressor* and the *ExtraTreesRegressor*, as will be discussed in the following subsection II-B, enabling us to ascertain this property across the different features. In doing so we observed that, for both models, the features were arranged in terms of importance grouped by category, broadly speaking. Specifically, as shown in Figure 3, in decreasing order there were: *pmax*, *negpmax*, *area*, *rms*, and *tmax*. For this reason, we decided to remove all the features from the last two categories, *rms* and *tmax*, thus keeping only 36 features. Moreover, our assumption on the importance of the *pmax* features proved to be correct.

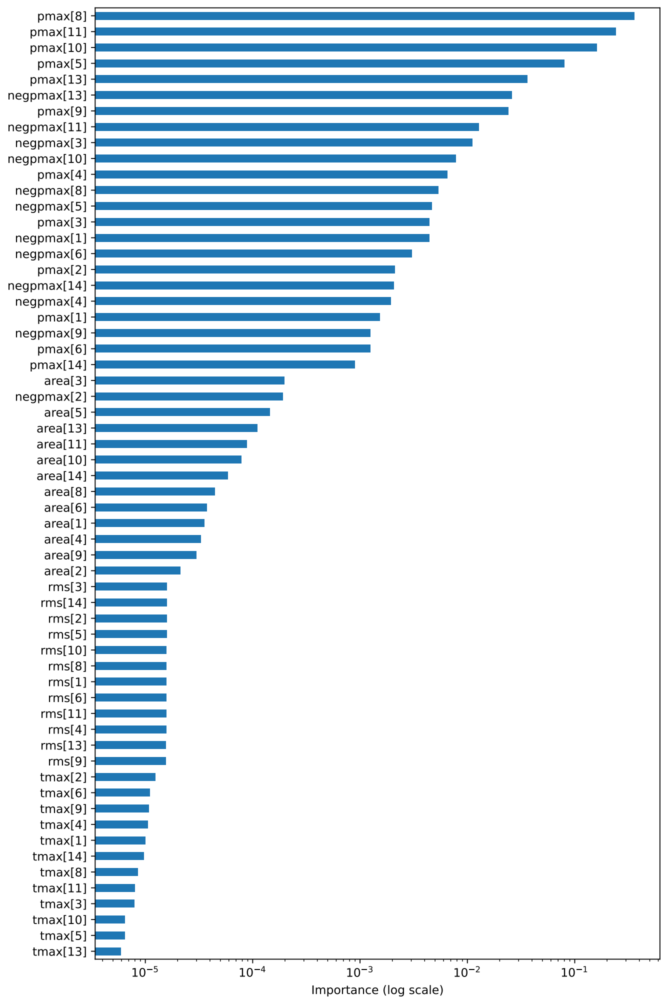

Fig. 3. Log-scaled feature importance from the *RandomForestRegressor* model.

All our choices in the preprocessing phase, before being implemented, underwent validation through experimental evidence. By progressively making these elimination decisions, in fact, we achieved increasingly satisfactory results in terms of performance when compared to that of the *RandomForestRegressor* fitted on the raw *development* dataset (without preprocessing), which served as our baseline.

However, a preprocessing action that was not embraced involved eliminating the correlation between independent variables. In fact, despite knowing from the theory that it is a good practice to address the correlation between a pair of variables by removing one of them, applying this feature selection technique led to poorer outcomes. Consequently, we abandoned this approach and retained all the remaining 36 variables. In the case of our dataset, specifically, we observed that the *area* features of all the pads were correlated with the corresponding *pmax* property. When attempting to remove all the *area* attributes, which were less important than *pmax*, we witnessed a decline in performance.

B. Model selection

We will now examine the models that have been considered and tested for the regression task.

1) *RandomForestRegressor*: we opted for this model because of its several favorable characteristics: it is especially effective in managing datasets with high dimensionality, mitigating the risk of overfitting. Furthermore, it demonstrates robustness in the presence of outliers and correlation among independent variables, due to the random selection of subsets of features at each split. Additionally, the algorithm automatically provides insights into the variables' importance, enabling us to eliminate less relevant features, reducing the complexity of the problem [1].
   The algorithm employs a series of decision trees, each trained on a different subset of the original training set, for making predictions. For each node in the various trees it selects the best split, among a subset of randomly chosen features, based on the purity of the division. This approach certainly reduces the risk of overfitting compared to a single Decision Tree, while still maintaining a certain degree of interpretability.
   However, the search for the *best* feature may still lead to overfitting in some cases. Furthermore, it's important to note that the use of various subsamples from the original learning set could potentially impact prediction quality negatively [2].
   To address these concerns, we incorporated an additional ensemble method, the *ExtraTreesRegressor*, which partially mitigates these issues by introducing more randomness into the tree-building process and working with the whole dataset.
2) *ExtraTreesRegressor*: as anticipated, this algorithm introduces greater randomness in the creation of trees. Specifically, it selects the feature and its corresponding cut-point for a given split fully at random, without conducting any search for the *best* split. Additionally, the individual trees are trained on the entire learning sample, rather than on a subset of it [2].
3) *KNeighborsRegressor*: this model has been tested on the data, but it did not produce satisfactory results. This is likely due to the high number of features in our dataset records. The *KNeighborsRegressor*, in fact, is particularly sensitive to the issue known as the "curse of dimensionality", as distance calculations between data points form the basis of the algorithm's operation.
   For this reason, we quickly abandoned this model, focusing instead on the first two, which proved to be much more promising.

C. Hyperparameters tuning

To tune our two chosen models, we initially divided the *development* dataset, allocating 80% and 20% of observations to the training and test sets, respectively. To find the most favorable combination of hyperparameters, we performed a grid search with 3-folds cross-validation for each of the two models. Since *RandomForestRegressor* and *ExtraTreesRegressor* operate very similarly, the hyperparameters that regulate their functioning are also the same. We, therefore, present a single table displaying the values of the various hyperparameters considered for both models (Table I).

TABLE I
HYPERPARAMETERS CONSIDERED FOR BOTH *RandomForestRegressor* AND *ExtraTreesRegressor*

Hyperparameters	**Values**
n_estimators	{100, 200, 300}
criterion	{"squared_error", "poisson"}
max_depth	{None, 10, 30, 50}
max_features	{"sqrt", 0.33, 0.50}

We have considered the hyperparameter *max_features* with values such as "*sqrt*", 0.33, and 0.50, to reduce overfitting risk and get results faster [3].

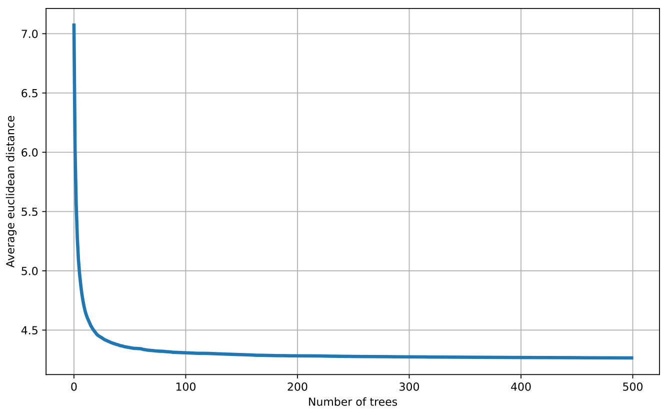

Fig. 4. The evolution of the average Euclidean distance between predictions and true values, in relation to the number of trees [4]

Regarding the hyperparameter *n_estimators*, a brief consideration is warranted. For both the *RandomForestRegressor* and the *ExtraTreesRegressor*, performance scales with the number of estimators up to a certain point, as illustrated in Figure 4. Even though the decrease in the average Euclidean distance becomes negligible beyond 100 estimators, we extended our consideration up to 300. This decision was made considering the acceptable trade-off between computational time and the marginal gain in performance.

III. RESULTS

Now let's examine and compare the results produced by the two models under consideration.

In table II the best configurations found are shown.

TABLE II
BEST CONFIGURATIONS

Model	**Hyperparameters**	**Values**
RandomForestRegressor	*n_estimators*	300
	criterion	”squared_error”
	max_depth	None
	max_features	0.33
ExtraTreesRegressor	*n_estimators*	300
	criterion	”squared_error”
	max_depth	None
	max_features	0.50

The average Euclidean distance obtained with *RandomForestRegressor* on our test set was approximately 4.03, while with *ExtraTreesRegressor*, the average Euclidean distance was about 3.92. On the public *evaluation* set, instead, we obtained scores of 4.762 for the *RandomForestRegressor* and of 4.621 for the *ExtraTreesRegressor*. Notably, the results are very similar, but the latter model performed slightly better, demonstrating a marginal improvement in predictive accuracy.

IV. Discussion

Both proposed models provide quite similar results. Although they are comparable in terms of prediction quality, it is worth noting that the *ExtraTreesRegressor*, in addition to producing a slightly better estimate, is also more efficient than the *RandomForestRegressor*. In table III below, the average execution times for the fit operation on the *development* set and the predict operation on the *evaluation* set are presented. These operations were performed by the two regressors under consideration, each configured with the optimal hyperparameters.

TABLE III
AVERAGE EXECUTION TIMES

Model	**Tranining time**	**Predicting time**
RandomForestRegressor	15 min and 11 sec	7 sec
ExtraTreesRegressor	3 min and 58 sec	20 sec

The inherent randomness in split selection of the *ExtraTreesRegressor* contributes to a more efficient training process, leading to time savings when compared to the process of searching for the *best* split, as in the case of the *RandomForestRegressor*.

References

[1] N. A. I. M. Jehad Ali, Rehanullah Khan, *Random Forests and Decision Trees*. IJCSI International Journal of Computer Science Issue, 2012.

[2] L. W. Pierre Geurts, Damien Ernst, *Extremely randomized trees*. Springer Science + Business Media, Inc., 2006.

[3] J. F. Trevor Haustie, Robert Tibshirani, *The Elements of Statistical Learning*. Springer, 2009.

[4] https://www.kaggle.com/code/ahmedabdulhamid/best-n-estimators-for-randomforest, 2022.