Particles trajectory prediction problem

Tommaso Mazzarini, Leonardo Merelli
Politecnico di Torino
Student id: s332078, s332464
s332078@studenti.polito.it, s332464@studenti.polito.it

***Abstract*—This report explores predicting particle positions detected by the Resistive Silicon Detector (RSD). We start by cleaning the data to remove noise and irrelevant information, ensuring our dataset is well-prepared for model training. Then, we develop and compare two forecasting models, selecting the one with superior performance based on specific parameters. This method enables us to achieve satisfactory results efficiently.**

I. Problem overview

Our study focuses on a regression problem in order to identify the positions through which particles, called events, travel their trajectories. The dataset in question includes 514,000 records from a Resistive Silicon Detector (RSD) sensor designed to detect the passage of such particles and determine their positions. With 12 pads, the sensor measures the properties of a signal directly related to the position of the passing particles. The dataset is divided into two subsets:

- a *development set*, containing 385500 events with labels;
- an *evaluation set*, containing 128500 events.

The goal of our work is to develop a supervised regression model using the *development set* that can accurately predict the positions of events in the *evaluation set*.

Each event in the two subsets has 18 instances for each of the following properties:

- *pmax*: the magnitude of the positive peak of the signal;
- *negpmax*: the magnitude of the negative peak of the signal;
- *tmax*: the delay from a reference time when the positive peak of the signal occurs;
- *area*: the area under the signal;
- *rms*: the root mean square value of the signal.

Among the 18 instances for each feature, 12 come directly from the pads, while the remaining 6 are introduced as a consequence of hardware constraints and are therefore treated as noise.

Moreover, after examining both datasets, we discovered that there are no missing values for any feature

Additionally, as previously mentioned, only the development set contains labels (denoted as x and y) indicating the particle's passage position on the sensor's two-dimensional surface.

When visualizing the data (Figure 1), we notice certain areas of the sensor where no particles are detected. This occurs because those coordinates correspond to locations covered by pads or wires, which reflect the signals.

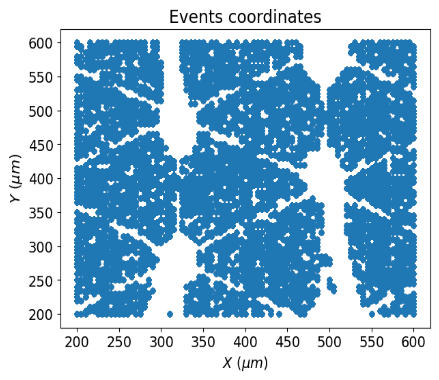

Fig. 1. Distribution of particles on the sensor surface

II. Proposed approach

A. Preprocessing

In order to prepare the data for model training, we conducted a thorough examination of each feature's behavior, aiming to identify pads that might serve as sources of noise. In particular, our focus was on the *pmax* feature, where we observed the highest values. For each instance, ranging from *pmax[0]* to *pmax[18]*, we specifically considered values exceeding the mean plus twice the standard deviation. Subsequently, we extracted the corresponding positions of events for visual representation, as illustrated in Figure 2.

In this visualization, orange dots highlight events with elevated *pmax* linked to specific pads, while blue dots depict how events are distributed over the sensor surface

Examining Figure 2, it is observed that the noise appears to come from the alleged "pads" numbered 0, 7, 12, 15, 16, and 17. This inference follows from the fact that the orange dots cover the entire surface of the sensor rather than being confined to specific areas associated with a particular pad.

As for "pad" 15, it was difficult to determine whether it contributes to the noise. However, looking at Figure 2, it can be seen that the alleged "pad" 15 is able to detect high voltage spikes in distant areas of the sensor. This suggests that it is indeed a source of noise.

Therefore, we eliminated all the columns of features related to noise. As a result, the *development set* is reduced from 90 to 60 columns.

Once the noise was removed from the dataset, we focused on analyzing outliers. For this, we used boxplots, a graphical

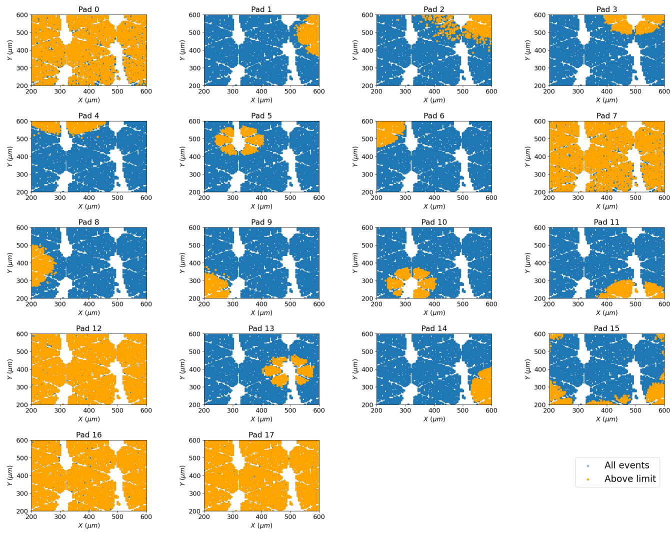

Fig. 2. Distribution of particles with high *pmax* values in relation to pads

representation that provides an overview of data distribution. Notably, we observed a substantial number of outliers.

However, we chose to remove as few of them as possible. This decision was made because these outliers are related to other values present in several features of the *development set*. Removing those values would not only result in the loss of useful information, but could also reduce the size of the dataset to a level not appropriate for training the predictive model.

Specifically, by looking at the boxplot for the negpmax feature (Figure 3), we noticed that some pads showed positive values, which is not feasible since this feature measures negative voltage spikes of the signal. Consequently, we eliminated those values.

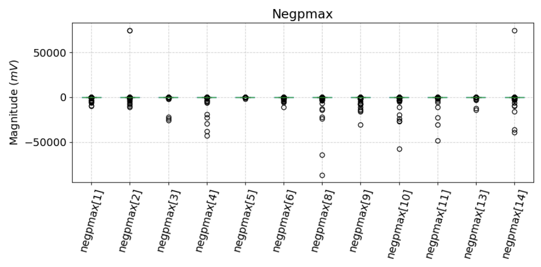

Fig. 3. Boxplots of *negpmax* features

To identify other outliers, we exploited two approaches based on:

- *Z-score*: a statistical measure that quantify the extent to which a value deviates from the mean in terms of standard deviation [1]. Its equation is:

$$z = \frac{x - \mu}{\sigma}$$

  Where:
  - x: is the variable
  - μ: is the mean
  - σ: is the standard deviation

  For this first approach, we calculated the z-score for each feature within the dataset, considering individual pads (e.g., *pmax[1]*, *negpmax[1]*, ..., *rms[1]*). Following that, we computed the mean of the z-scores for each event. Ultimately, for each pad, we created a list of these mean z-scores corresponding to the number of events in the dataset. We then focused on a subset of events with a z-score mean below a specific threshold for each pad, excluding those that were common across all pads.
- *Mahalanobis distance*: a metric that, unlike Euclidean distance, takes into account not only the distance between variables, but also their correlation [2]. Its equation is:

$$D(X, \mu) = \sqrt{(X - \mu)^T \Sigma^{-1} (X - \mu)}$$

  Where:
  - X: is the variable
  - μ: is the mean of the distribution
  - Σ: is the covariance matrix

  In particular, for this second approach, we compute the Mahalanobis distance for each event. Subsequently, we proceeded by eliminating all observations that had a distance greater than a certain limit selected by us. This choice of limit was carefully made to exclude only a limited number of outliers, aiming to maintain a balance between removing anomalous data and preserving meaningful information.

After removing the outliers, we proceeded with the features analysis, with the goal of identifying and eliminating the less significant ones. To this end, we trained a regression model based on Random Forest that extracted the feature importances, as is shown in Figure 4.

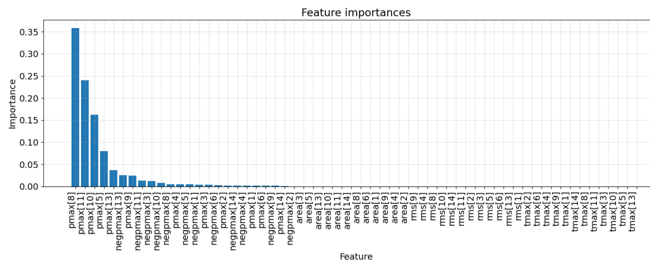

Fig. 4. Feature importances

From the Figure 4 it is possible to see that the most significant features are *pmax* and *negpmax* followed by *area*, *tmax*, and *rms*. In this regard, we decided to remove *tmax* and *rms* since they are the least important.

B. Model selection

The following algorithms have been tested:

- *Random forest*: is an algorithm that is based on using several randomized decision trees and combines their predictions by averaging [3]. During the training process, the algorithm builds a set of decision trees by dividing the

dataset into subsets and training each tree on a specific set of features for each subset.
The strengths of this algorithm lie in its ability to avoid overfitting, improve accuracy, and increase robustness to noise and outliers. Additionally, its efficiency is notable, as it enables parallel execution. For these reasons, we chose to use random forest regressor for our prediction.

- *k-Nearest Neighbours* (kNN): this algorithm is based on the assumption of locality in data space. In local neighborhoods of a given value, patterns are expected to have similar output value to the corresponding function output [4]. To make a prediction using this algorithm, we need to choose the number *k*, of the nearest neighbors. The algorithm calculates the distance between the new input data and all other data points in the dataset, identifies the K nearest points, and the predicted value is determined by the average of the values of these neighboring points.

For both algorithms, the best-working configuration of hyperparameters has been identified through a grid search, as explained in the following section.

C. Hyperparameters tuning

We identified two distinct sets of hyperparameters to be tune:

- random forest parameters;
- kNN parameters.

We conducted a grid search for both regression models, using the same split ratio between training and test data (80/20) and the parameters defined in Table I.

Model	Parameter	Values
Random forest	n_estimators	100 $\rightarrow$ 1000, step 100
	criterion	[squared_error, absolute_error, friedman_mse, poisson]
	max_features	[sqrt, log2, None]
	max_depth	[10, 20, 30, 40, 50, None]
	min_samples_split	[2, 4, 5, 8, 10]
	min_samples_leaf	[2, 4, 6, 8, 10]
kNN	n_neighbors	5 $\rightarrow$ 50, step 5
	weights	[uniform, distance]
	algorithm	[auto, ball_tree, kd_tree]

TABLE I
HYPERPARAMETER TUNING

In particular, for the random forest, to mitigate high computational costs, we implemented two distinct grid searches. The first one aimed to determine the optimal number of estimators, and as expected, the performance scales with the number of estimators. Nevertheless, we used 300 estimators to calculate the results because we achieve reasonable performance while keeping the computational cost at a manageable level. Meanwhile, the second grid search was employed to identify the best values for the remaining parameters.

III. Results

The best configuration for random forest, with 300 estimators, is as follows:

- $criterion$: $squared\_error$
- max_depth : 40
- $max\_features$: $sqrt$
- $min\_samples\_leaf$: 2
- $min\_samples\_split$: 2

While the best configuration for kNN is as follows:

- $n\_neighbors$: 10
- $weights$: $uniform$
- $algorithm$: $auto$

The two regressors yield satisfactory results, that are presented in Table II; however, the random forest achieves a better outcome, and indeed, it is the one employed for prediction. This behavior is a consequence of disparities in characteristics between the two models. The Random Forest demonstrates increased robustness to noise and better scalability compared to the kNN. The latter, when faced with a large training set, may experience a degradation in its performance.

After training both regression models on the *development set*, we used them on the *evaluation set* to predict the position of the signals on the sensor surface.

We employed different metrics to measure the improvement of the models, like:

- R^2 score;
- Mean Absolute Error (MAE);
- Mean Squared Error (MSE).

But to effectively evaluate the final accuracy, we used the average Euclidean distances (reported in Table II as "Public score") between the predicted positions and the actual ones.

Model	R^2 score	MAE	MSE	Public score
Random forest	0.9991	2.6732	12.1583	4.750
kNN	0.9985	2.8061	20.1154	5.683

TABLE II
RESULTS

IV. Discussion

The proposed approach obtains satisfactory results with reduced computational time, achieved by eliminating noise and redundant values. However, there are some aspects that could enhance the achieved outcome, such as:

- run multiple grid search in order to obtain even more refined hyperparameter configurations;
- use of Artificial Neural Networks (ANNs): due to their generalization ability, which allows them to "see through" the noise and distortion of weak relationships, offering acceptable levels of accuracy in prediction [5].

References

[1] H. Abdi, "Z-scores," *Encyclopedia of measurement and statistics*, vol. 3, pp. 1055–1058, 2007.
[2] H. Ghorbani, "Mahalanobis distance and its application for detecting multivariate outliers," *Facta Universitatis, Series: Mathematics and Informatics*, pp. 583–595, 2019.
[3] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, pp. 197–227, 2016.
[4] O. Kramer, "Unsupervised k-nearest neighbor regression," *arXiv preprint arXiv:1107.3600*, 2011.

[5] L. Marquez, T. Hill, R. Worthley, and W. Remus, “Neural network models as an alternative to regression,” in *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*, vol. 4, pp. 129–135, IEEE, 1991.