Resistive Silicon Detector Regression Problem

Alessio Giuffrida
Francesco Giannuzzo
Politecnico di Torino
Student id: s328964, s328830
s328964@studenti.polito.it, s328830@studenti.polito.it

***Abstract*—In this report we propose a possible solution to the RSD (Resistive Silicon Detector) regression problem. The proposed approach consists in building a regression model to predict for each event the target (x,y) coordinates where the particle of interest passed. In particular, we try to identify the noisy readings and remove them, along with other features that are deemed not important through some statistical analysis. In this way we obtain a model that gives satisfactory results, outperforming a naive baseline.**

I. Problem Overview

The dataset proposed describes the readings of a RSD composed by 12 metallic pads capable of detecting the passage of a particle, called an event, and measuring a signal: the properties of this signal, which are correlated with the position where the particle passes, are extracted and stored. Since we are dealing with a 2-dimensional surface, the position of the particle, which is the target of the regression task, is defined as a pair of (x,y) coordinates, in μm. The dataset is divided into two parts:

- the development set, composed by 385,500 events
- the evaluation set, composed by 128,500 events.

We will use the development set to build a regression model to correctly predict the (x,y) coordinates of the readings in the evaluation set. We can visualize the positions of the events that comprise the development set through a scatter plot to better understand how they are distributed. As we can see in Figure 1, the distribution of the events highlights the "snowflake" shape of the pads, as we expected. Therefore, at the end of our prediction task we expect to obtain a similar result.

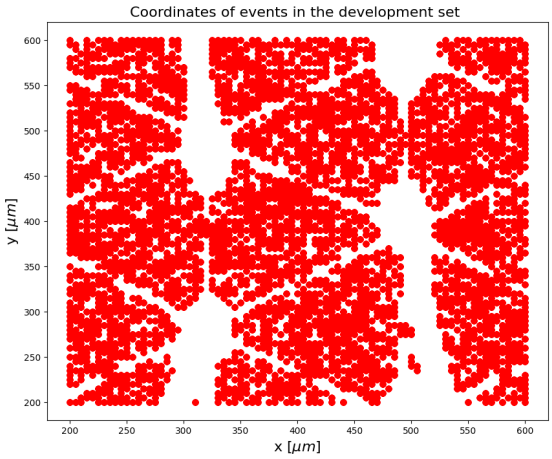


Fig. 1: Events distribution of positions in development set

Each event is associated with some features, which constitute the dataset:

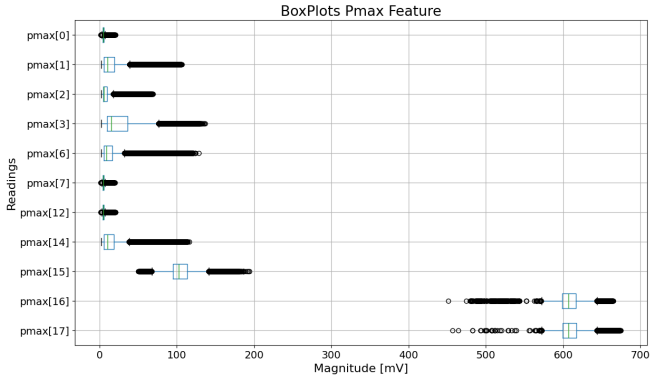
- *pmax*: the magnitude [mV] of the positive peak of the signal
- *negpmax*: the magnitude [mV] of the negative peak of the signal
- *area*: the area [mV· ns] under the signal of an event
- *tmax*: the delay [ns] from a reference time when the positive peak of the signal occurs
- *rms*: the Root Mean Square [mV] value of the signal.

We can see that the dataset contains only numerical features and includes 18 readings for each event. However, we know that there are only 12 pads: this means that some of these readings are affected by noise. By inspecting the development set we can notice that there are no null values, so the data are complete. Moreover, there are no duplicated readings. The main goal of the preprocessing phase will therefore be selecting the most relevant features considering the presence of noise.

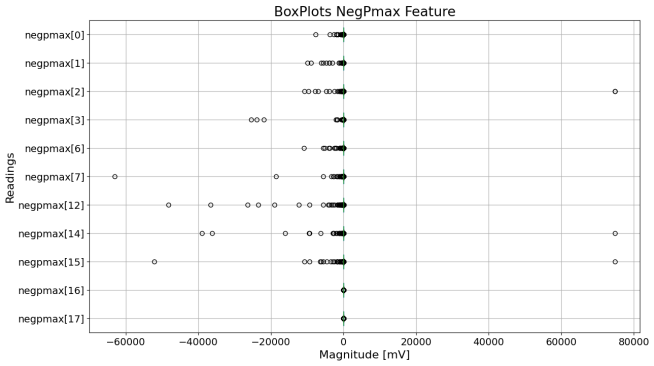
II. Proposed Approach

A. Preprocessing

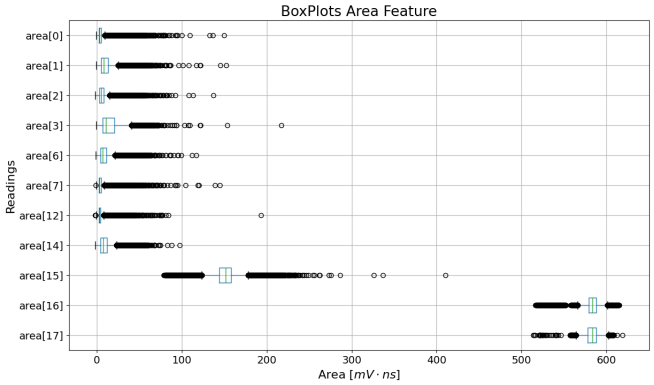
At first, to analyse the values assumed by the features we compute some statistics, namely the mean and the standard deviation. Then we examine the distribution of the features, which can be summarized by the boxplots in Figures from 2a to 2e, in which we omit some of the readings of the pads because they have a similar behaviour to others.



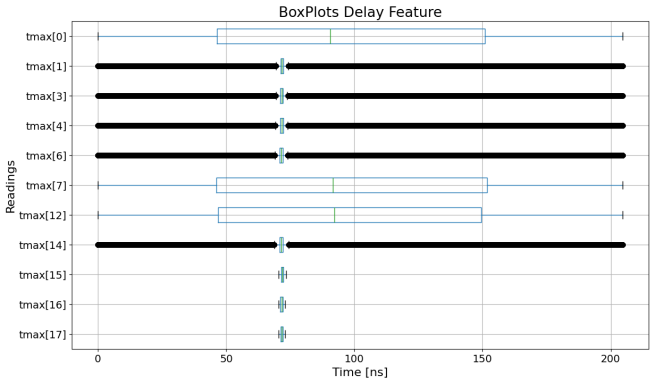
(a) Pmax Readings



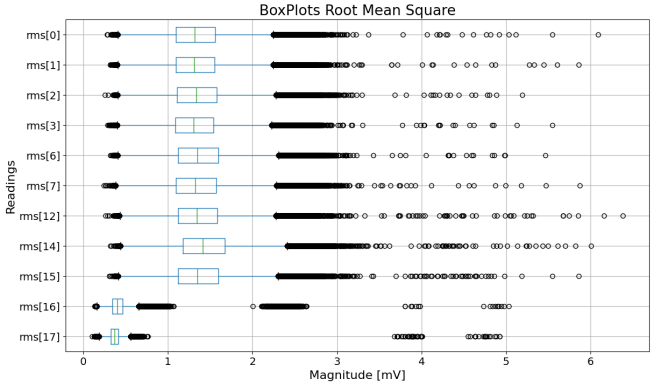
(b) NegPmax Readings



(c) Area Readings



(d) Delays Readings



(e) Rms Readings

Fig. 2 : BoxPlots of the distribution of the features

We can immediately make some observations:

- it is noticeable that the distribution of the values of pmax is very similar to the distribution of the area
- the rms gives more general information because it is an aggregated feature, so we believe it could be less relevant, considering we have more specific features like pmax and negpmax
- tmax is a temporal feature, in contrast with the others which concern the intensity of the signal, so we do not expect it to be really that relevant.

A relevant result is the substantial present of outliers in the readings: these are justified by the physical nature of the problem, so they may be valid measurements. For this reason we decide to keep them in the dataset with the exception of some positive outliers in the negpmax feature that have no physical meaning as this feature should include only negative numbers.

In order to extract the most relevant features we analyse the means and standard deviations of pmax, since we presume that this feature will have high importance in the regression task: we can observe that pmax[16] and pmax[17] have mean values much higher than the others, whereas pmax[0], pmax[7] and pmax[12] have standard deviations much lower than the others. We suspect that these are noisy readings.

From the previous observations we decide to exclude the readings 0, 7, 12, 16 and 17 and the features rms and tmax from our analysis. Furthermore, we can try to construct new features that better characterize the signals: since pmax is very important, we choose to add as new feature the peak-to-peak amplitude, given by the sum between pmax and the absolute value of negpmax. Another approach to reduce the number of features that can give satisfactory results is the PCA: this is a linear algebra for continuous attributes that finds new attributes (called principal components) that are linear combinations of the originals and capture the maximum amount of variation in the data [1]. Figure 3 shows that 14 principal components are able to explain at least 90% of the total variance.

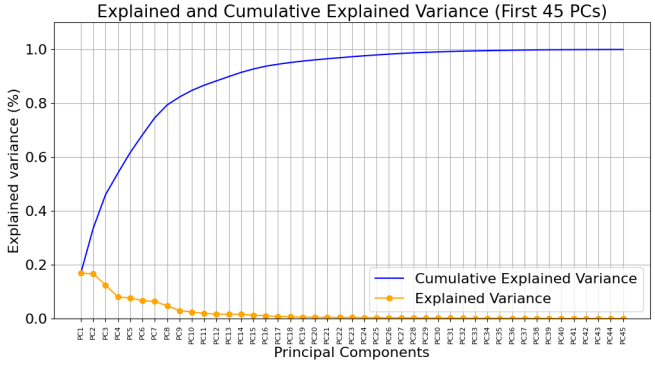


Fig. 3: Cumulative explained variance and explained variance of each component

During the training of our models we therefore use two different configurations of features: one with the PCA and the other removing the features through statistical analysis.

B. Model selection

The following algorithms have been tested:

- *K-Neighbors Regressor*: this model is based on k-nearest neighbors, which finds the k nearest data points to a given input and predicts the numerical value of the output variable by computing the average of the target values of the neighbors.
- *Random Forest Regressor*: this algorithm is an ensemble technique that builds multiple decision trees, which are trained on subsets of data and features to make predictions. In this way it is possible to avoid overfitting while maintaining some degree of interpretability. Random forests, like decision trees, work on one feature at a time, so normalization is not necessary.

The best configuration of hyperparameters for our models has been identified through a grid search, as explained in the following section.

C. Hyperparameters tuning

Firstly we want to verify whether the PCA is an effective approach for this problem, so we perform a 80/20 train/test split on the development set and then we train a random forest regressor and a k-neighbors regressor with default hyperparameters, obtaining the following results:

- Random Forest Regressor: score = 13.301
- K-Neighbors Regressor: score = 23.786

It is clear that the introduction of PCA gives unsatisfying results, so we decide to discard this solution and focus on the feature reduction made by our statistical analysis.

Considering the development set, we decide to run a grid search with 5-fold cross validation on both random forest and k-neighbors, based on the hyperparameters defined in Table I. The scoring metric used is the average Euclidean distance.

Model	Parameter	Values
Random Forest Regressor	*n_estimators*	{50, 100, 200}
	criterion	{squared_error}
	max_features	{sqrt, log2}
	n_jobs	{-1}
	random_state	{42}
K-Neighbors Regressor	*n_neighbors*	{5, 7, 9}
	weights	{uniform, distance}
	algorithm	{auto}

TABLE I: Hyperparameters considered

III. Results

The best configuration found for the random forest is {*criterion=squared_error*, *max_features=sqrt*, *n_estimators=200*, *n_jobs=-1*, *random_state=42*} with score = 3.921, whereas the best configuration for k-neighbors is {*algorithm=auto*, *n_neighbors=9*, *weights=distance*}, with score = 4.073. The two models achieve satisfactory and comparable results. Furthermore, thanks to the interpretability of the random forest we can check the importance given by the model to each feature, verifying our initial assumptions about pmax being the most relevant feature.

We trained the best performing random forest Regressor and k-neighbors regressor on the whole development set and then we predicted the x and y values for each event in the evaluation set. The public score obtained is 4.670 for the random forest regressor and 5.698 for the k-neighbors regressor. We can see that on the evaluation set the k-neighbors performs much worse than the random forest. For comparison we also defined a naive solution by selecting all the features present in the development set and training a default linear regression model on them: the public score obtained is 17.35. To further check the accuracy of our predictions we can plot the distribution of the (x,y) coordinates predicted using the random forest regressor: Figure 4 shows that the model is able to identify the regions where pads are present, but the zones occupied by wires are not so well distinguishable.

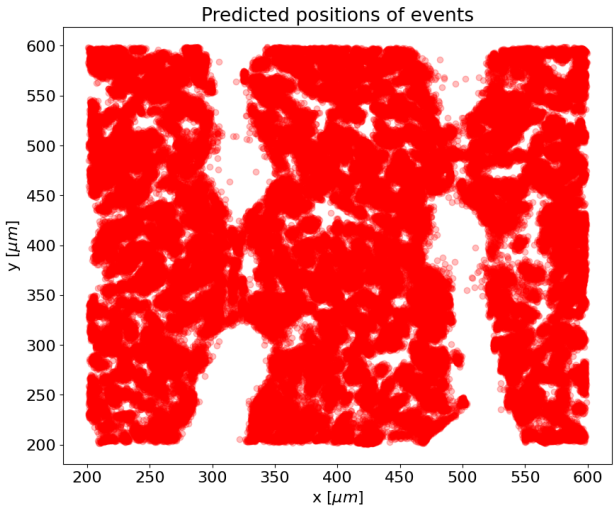


Fig. 4 : Distribution of predicted coordinates

IV. Discussion

The results obtained with the random forest are far better than the naive solution and k-neighbors regressor. However, there are different aspects to consider that may help to improve the obtained results:

- Try different models, like MLPRegressor (Multi-layer Perceptron Regressor): it is a supervised learning algorithm based on a feedforward neural network. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers [2]
- Construct new features that are able to better describe the problem from a physical point of view, for example taking advantage of knowledge pertaining the nature of the particles
- Run a more exhaustive grid search. In our solution we have analysed only a limited set of hyperparameters, but

exploring different combinations could result in a better configuration.

Nevertheless, the results obtained with our approach are already very satisfying and represent a good starting point to solve the proposed problem.

REFERENCES

[1] P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining, Second Edition*. Pearson, 2019.

[2] *Scikit-learn library: https://scikit-learn.org/stable/modules/neural_networks_supervised.html*.