RSD Sensor Regression Problem

Francesca Geusa, Carlo Marra
Politecnico di Torino
Student id: 329174, 249863
s329174@studenti.polito.it, s249863@studenti.polito.it

***Abstract*—In this report we propose a possible solution to predict the position of a passing particle through an RSD (Resistive Silicon Detector) sensor. The suggested method integrates stages for both feature selection and feature engineering, enhancing the characterization of the data. Two Ensemble Methods, Random Forest and HistGradient Boosting Tree, are compared, with Random Forest being selected for further refinement due to its superior performance.**

I. Problem overview

The objective of this regression problem is to predict the position, expressed in (*x*,*y*) coordinates, where the particle of interest passed through an RSD sensor. The sensor in question has a 2-dimensional surface on which 12 pads are used to detect the passage of a particle. Each pad is able to estimate five measurements, more specifically:

- *pmax*: the magnitude of the positive peak of the signal, in mV
- *negpmax*: the magnitude of the negative peak of the signal, in mV
- *tmax*: the delay (in ns) from a reference time when the positive peak of the signal occurs
- *area*: the area under the signal
- *rms*: the root mean square (RMS) value of the signal.

The prediction of the position is based on the information above.

The dataset provided is composed by:

- A **development set** of 385,500 rows, each representing the passage of a particle through the sensor (*event*). For each row there are 18 readings of each feature for a total of 92 columns including the target features (*x*,*y*).
- An **evaluation set** of 128,500 events with the same structure of the development set, except for the target values.

The existence of 18 readings per feature is justified by hardware constraints in the data acquisition phase. This indicates the probable presence of noise in the dataset, since a subset of the total predictors does not represent actual readings. To address this problem, it might be useful to consider a feature reduction process. A possible solution to find noisy features can be a search for low variance attributes or correlation-based method.

Nevertheless, the dataset does not contain any missing values.

II. Proposed approach

A. Preprocessing

The first problem to address is identifying and eliminating the 30 features added by the hardware, since they are to be considered pure noise and, consequently, irrelevant features in the dataset. To do so, we compute the **correlation matrices** for all five measurements. Looking for near-zero values in the correlation matrix is a strategy used to identify features that have little or no linear correlation with other features in the dataset, this can be a good indicator for spotting noisy attributes. We use the **Pearson's correlation coefficient** (commonly denoted as ρ), which is able to measure linear correlation between two sets of data (X, Y) [1] and the **Spearman's rank correlation coefficient** (commonly denoted as r_S), which is more robust to noise and can spot non-linear relationships. The first one is defined as follows:

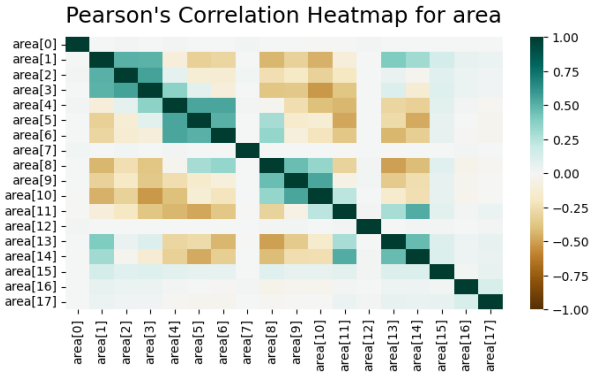
$$\begin{aligned} \rho_{XY} &= \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} \\ \text{where} \quad \text{cov} &= \text{Covariance} \\ \sigma_X &= \text{SD of X} \\ \sigma_Y &= \text{SD of Y} \end{aligned}$$

The Spearman's correlation is, given two sets of data (X, Y), defined as follows [2]:

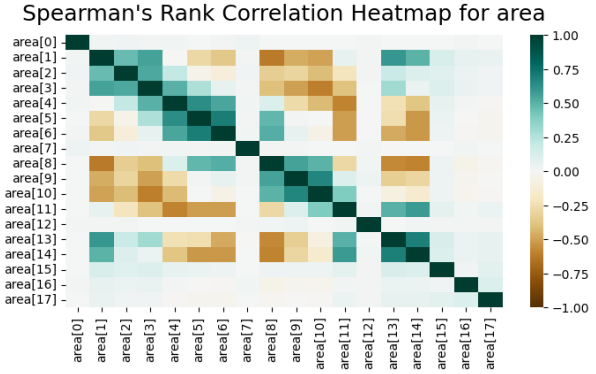
$$\begin{aligned} r_s = \rho_{R(X),R(Y)} &= \frac{\text{cov}(R(X), R(Y))}{\sigma_{R(X)} \sigma_{R(Y)}} \\ \text{where} \quad R(X), R(Y) &= X, Y \text{ converted to ranks} \\ \text{cov} &= \text{Covariance} \\ \sigma_X &= \text{SD of the rank variable X} \\ \sigma_Y &= \text{SD of the rank variable Y} \end{aligned}$$

Correlation matrices can be visually represented through a heatmap, making it easier to identify high and low correlation coefficients. Specifically, by examining the Pearson's and Spearman's rank correlation matrices for the *area* and *pmax* measurements (respectively observable in figures 1 and 2), it is possible to observe exactly 6 pads that have nearly zero correlation with the other 12, namely pads 0, 7, 12, 15, 16, and 17. These particular pads will be considered by us as noise pads and consequently discarded.

We can now construct training and testing sets excluding the noisy features mentioned above. In this context, the optimal choice for data partitioning is the **holdout technique**, particularly suitable for large datasets or datasets with high

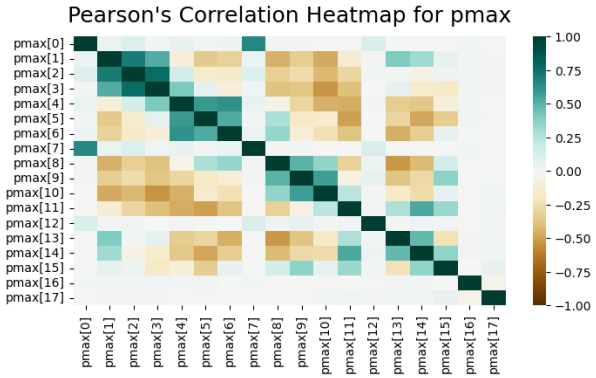


(a) *area* Pearson's Correlation Matrix heatmap

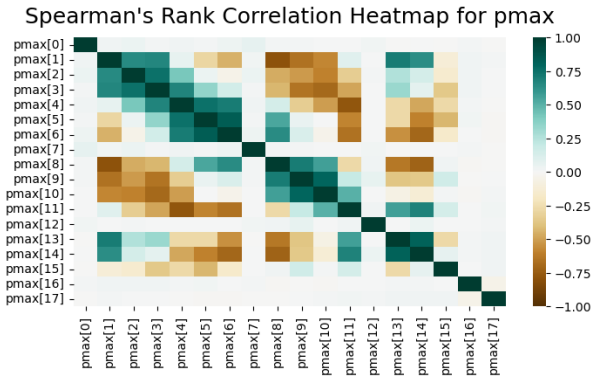


(b) *area* Spearman's Rank Correlation Matrix heatmap

Fig. 1. *area*'s Correlation Matrices



(a) *pmax* Pearson's Correlation Matrix heatmap



(b) *pmax* Spearman's Rank Correlation Matrix heatmap

Fig. 2. *pmax*'s Correlation Matrices

cardinality. From here onwards, for the initialization, training, and evaluation of the models, we rely on the scikit-learn API [3]. Scikit-learn is an open-source machine learning library for the Python programming language, offering straightforward and efficient tools for data mining and analysis. It is built on top of other widely-used scientific computing libraries, including NumPy, SciPy, and Matplotlib.

We employ a Random Forest regressor, incorporating the Multi-Output Regressor extension, to establish an initial baseline evaluated using the average Euclidean distance metric defined as:

$$d = \frac{1}{n} \sum_{i} \sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2}$$

where (x_i, y_i) are the real coordinates and $(\hat{x}_i, \hat{y}_i)$ are the coordinates predicted. The regressor serves the purpose of extracting feature importances, elucidating the utility of each feature within the model. Analysis reveals that the two measurements *tmax* and *rms* are deemed as irrelevant features (as it is evident in table), leading us to exclude them.

Feature	**Importances**
tmax	8.926e-05
rms	1.768e-04
area	6.728e-04
negpmax	0.020
pmax	0.9787

This reduction in feature dimensionality enhance the baseline regressor's performance up to 2%, as shown in the comparison table below, as well as an overall better computational performance.

Model	**Avg Euclidean Distance**
Baseline RF	4.697
Baseline RF without noise	4.690
Baseline RF without noise and unimportant features	4.593

B. Model selection

Model selection is a central topic to the process of building good (supervised) machine learning models and it plays a crucial role in obtaining satisfactory results.

As for the choice of the appropriate model, we have considered an Ensemble Method to be a suitable option. Ensemble methods, such as Random Forests and Gradient Boosted Trees, can be good options for several reasons, especially when dealing with regression tasks in datasets with high cardinality like the one we are dealing with. Some advantages are:

- **Stability**: combining multiple base models (trees in the case of Random Forests or Gradient Boosted Trees) can lead to more robust and stable predictions that are less sensitive to noise and outliers.

- **Good handling of high dimensional data**: individual trees in the ensemble can focus on different subsets of features, making them efficient for high-dimensional problems.
- **Ability to capturing relationships non-linear**: the relationships between the features and the target coordinates may be complex and non-linear and ensemble methods can handle such complexities.
- **Feature importance**: Ensemble methods provide a measure of feature importance, which can be valuable in understanding which features contribute more to the prediction. This can aid in feature selection and interpretation of the model.
- **Less chance of over-fitting**: By aggregating predictions from multiple trees, over-fitting to the training data is mitigated, leading to better generalization performance on unseen data.
- **Good handling of missing data**: In spite of the absence of missing data within the development and evaluation dataset, it is still advisable to prepare for the possibility that, in the real world, one of the 12 pads may malfunction and be unable to record an event.
- **Easy to tune and parallelization**: Ensemble methods often perform well "out of the box," and tuning can further enhance their performance. Moreover, they can be parallelized, allowing for faster training on large datasets.

In general, HGBT (Histogram Gradient Boosting Trees) manages to achieve better results in terms of both speed and accuracy compared to a Random Forest, whether using the default hyperparameters or considering the cost of hyperparameter tuning. However, as visible in the table below, the more favorable outcomes are achieved by Random Forest. This results are likely attributed to the fact that we are dealing with a MultiOutput Model. By fitting one tree per class internally during each boosting iteration, HGBT follows a different approach. However, the naturally multiclass trees employed by Random Forest models enhance the speed-accuracy trade-off [4], as demonstrated in this specific scenario. Therefore, our exclusive focus moving forward will be on fine-tuning the Random Forest Regressor model.

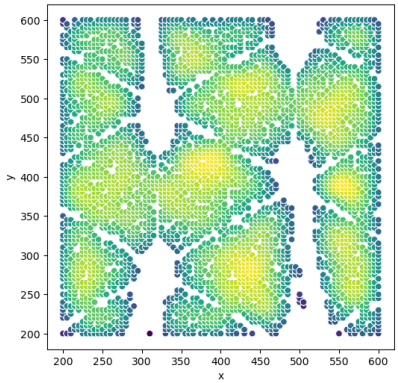
Baseline Model	Avg Euclidean Distance
Random Forest Regressor	4.593
Histogram-based Gradient Boosting Regression Tree	7.155

C. Hyperparameters tuning

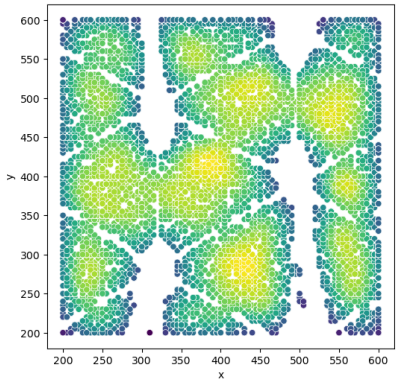
The high cardinality of the development dataset (almost 400,000 rows) brings both advantages and disadvantages in the hyperparameter tuning phase. Among the advantages, there is the opportunity to obtain a robust model that can capture nuanced information. Another advantage is certainly the possibility of using **cross-validation** method to get a reliable estimate of the model's performance. By dividing the available dataset into different portions (*folds*) and subsequently training and evaluating the model on various combinations of these portions, it helps mitigate the risk of obtaining performance evaluations that significantly depend on the specific data split into training and test sets and helps prevent over-fitting to the training data.

The biggest drawback, however, turns out to be the enormous computational burden of approaching hyperparameter tuning with a standard grid search on the entire available dataset. One possible solution to the problem is sampling a fraction of the data that is as representative as possible of the original dataset.

To compute a sample of an arbitrary percentage of the cardinality of the development dataset, we decide to perform a **stratified sampling**, a sampling technique that involves dividing the total population into homogeneous subsets (*stratum*) and then sampling from each of these subsets. Since all the coordinates go from a minimum of 200 to a maximum of 600 units, by dividing the dataset into intervals of 50 units in length, it is possible to obtain a total of 64 subsets and take 20% of the data from each specific quadrant. The result, as visible from the two scatter plots in Fig. 3, is satisfactory and the sample appears to be quite representative of the total dataset.



(a) Entire Dataset



(b) 20% Sample

Fig. 3. Scatterplots

In order to perform as comprehensive Hyperparameter Tuning as possible, we rely on Optuna [5]. Optuna is an open-source hyperparameter optimization framework designed to

automate hyperparameter search. It is capable of efficiently exploring extensive parameter spaces and pruning unpromising trials to accelerate results. Additionally, it can parallelize hyperparameter searches across multiple threads or processes. Initially, we execute a tuning round employing a K-fold cross-validation strategy with $K = 3$. After that Optuna concludes the search by providing the parameters associated with the best cross-validation score, it is possible to initiate a new cycle of exploration with increasingly specific parameters until the optimal result is achieved.

III. Results

The hyperparameters for the Random Forest Regressor that achieved the best score among the 200 executed trials can be found in the table below. With this model configuration, we

Parameter	Value
n_estimator	968
max_features	'sqrt'
max_depth	65
min_samples_split	3
min_samples_leaf	2

obtain an average (Euclidean) distance of the predictions from the targets of 4.371 through a cross-validation with 3 folds and 4.801 on the public evaluation set.

IV. Discussion

At the conclusion of the study, satisfactory results were achieved with the Random Forest Regressor. Unfortunately, a notable discrepancy exists between the outcomes obtained on the development set and those on the evaluation set. It can be confidently ruled out that overfitting is the cause, as model decisions were only influenced by cross-validation and on a validation set derived from the development set.

While it is conceivable that a distributional mismatch between the two sets may contribute to the observed differences, certainty in this regard is lacking. Nevertheless, the constructed model demonstrates significant stability.

Surprisingly, attempts to enhance performance by eliminating data points beyond the 1st and 99th percentiles, as a potential outlier cleansing strategy, consistently result in a deterioration of the model's performance. This highlights the nuanced relationship between the model and outliers, challenging conventional outlier removal practices.

The same can be said for feature scaling practices, as Random Forests are generally robust to the scale of features in the dataset, and they are not as sensitive to the scale of input variables as some other machine learning algorithms like Support Vector Machines or k-Nearest Neighbors.

The results highlight the complexities of model adaptation in practical scenarios, underscoring the necessity for a more profound comprehension of the interaction among data distribution, model robustness, and the impact of outliers. Further investigations are recommended to unravel the complexities of the observed performance variations and refine the model's predictive capabilities.

References

[1] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Pearson Education Limited, 2014.

[2] Wikipedia, "Spearman's rank correlation coefficient," 2005. [Online; checked 01-26-2024].

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] A. Amor, "Comparing Random Forests and Histogram Gradient Boosting models." https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_hist_grad_boosting_comparison.html, 2007. [Online; January 24, 2024].

[5] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.