Predicting Coordinates with Multioutput Regression in Particle Events in HEP

Pouria Mohammadalipourahari
Politecnico di Torino
Student id: s327015
s327015@studenti.polito.it

Reihaneh Kharazmi
Politecnico di Torino
Student id: s328816
s328816@studenti.polito.it

***Abstract*—This project utilizes advanced data science techniques to predict particle positions in high-energy physics, specifically using a Resistive Silicon Detector (RSD). The dataset comprises 514,000 events, each containing signal features from 12 detector pads, including magnitude, delay, and area. The primary objective is to develop a finely-tuned regression system that can predict 2D particle coordinates with high precision. The performance of the system is evaluated by measuring the average Euclidean distance between predicted and actual particle positions. This work represents the integration of data science and particle physics to enhance our understanding of particle behavior and tracking accuracy.**

I. Problem overview

This project addresses a unique regression challenge, necessitating dual-column predictions for each record, focusing on X and Y coordinates. These coordinates, restricted within the 400-600 integer range and uniformly distributed as illustrated in figures 1 and 2, are exclusively multiples of five. This characteristic implies that rounding predictions to the nearest multiple of five may refine model precision. The feasibility of a classification strategy, prompted by this pattern, is limited due to the simultaneous prediction requirement and the extensive potential classes, rendering it beyond this report's scope.

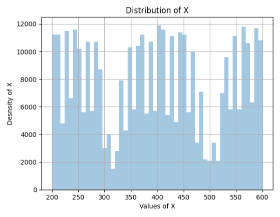
Fig. 1. Distribution of X

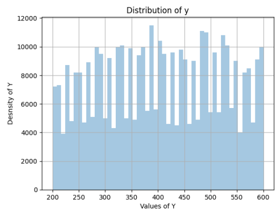
Fig. 2. Distribution of Y

Associative relationships among the five columns forming a pad need a specific criterion for noise identification. Analyzing feature distribution per pad, especially for pads 13, 14, and 15 illustrated in Figure 3 with 'pmax', is crucial. Pad 15 stands out with a unique distribution. Extending this analysis to all features and pads is essential to identify potential noisy pads. In post-noise-feature-reduction, the challenge persists, prompting careful examination of the substantial remaining features. The proposed approach suggests training a model on the 12 non-noisy pads and assessing each feature's significance in the model.

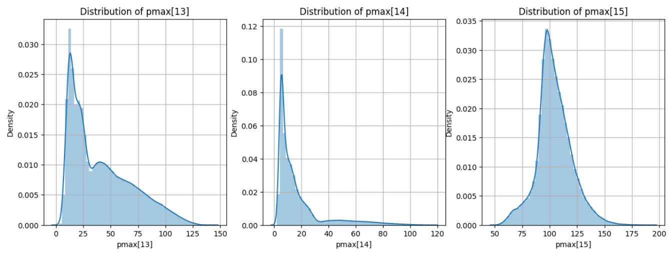
Fig. 3. Distirbution of pmax for pad number 13, 14, 15

The presence of outlier data in certain columns necessitates the use of efficient and effective outlier detection methods to maintain data quality. Due to the large dataset scale, the chosen method should strike a balance between thorough outlier removal and data preservation.

Furthermore, the unique characteristics of 'negpmax' feature columns with negative values emphasize the need for standardization to facilitate accurate model training and error mitigation. Additionally, the significance of 'rms' feature columns, representing a specific distance metric, suggests that a feature selection process driven by algorithms should be favored over the preemptive exclusion of these columns.

II. Proposed approach

A. Preprocessing

1) Noise Pads Elimination: Noise pad elimination begins with the application of a criterion designed to assess discrepancies among individual pads. To achieve this, each pad is aggregated into one representative metric based on its multiple features, a process that is replicated across all pads. An analysis of the compiled data suggests that the computation and storage of median values for each pad, thereby compressing the original 90 feature columns into 18, might play a significant role in distinguishing noise. A visual representation for these aggregated metrics would allow informed decision-making regarding the exclusion of noisy pads from the dataset. Figure 4, Which is the Box plot of medians for all 18 Pads, provides a clear visualization of the noise distribution among the pads. Pads 0, 7, 12 show noise due to median values confined to 0-20, notably lower than non-noisy pads. Conversely, pads 15, 16, 17 exhibit densely clustered median values, significantly deviating from the observed range in other pads.

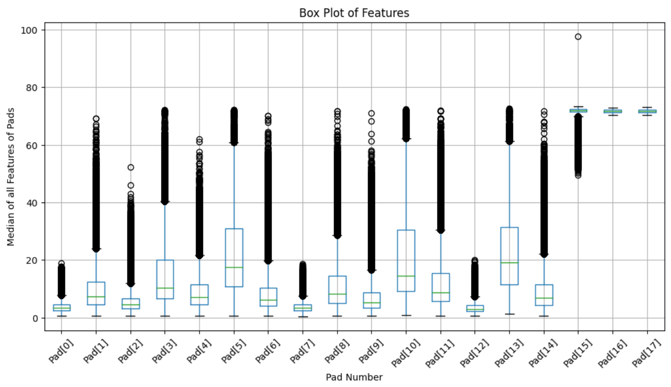

Fig. 4. Box plot of Medians for each pad

Another approach that can be utilized to ensure the correct selection of noise pads, in addition to the previous method, is to plot the distribution of each pad. This distribution is derived from the values obtained by median calculation from each record. In Figure 5, a comparison can be observed between the median values (which stored in each pad column)for pads 13, 14, and 15 based on their distributions. As was observed in Figure 4, pad number 15 exhibits a completely different behavior compared to pads 13 and 14.

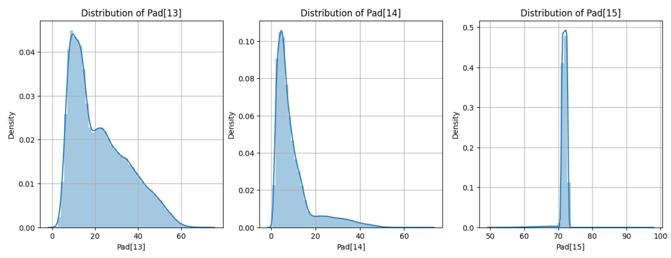

Fig. 5. Distribution of pad number 13,14,15

2) Outliers Removal: The outlier detection process involved three key methods: Z-Score analysis, Local Outlier Factor (LOF) [1], and the Isolation Forest [2] algorithm. Z-Score analysis measured deviations from the mean, but its application resulted in a significant dataset reduction. LOF and Isolation Forest, on the other hand, treated the problem as a clustering one by excluding target column data, making them suitable for these analyses. Both LOF and Isolation Forest effectively detected outliers, with the latter demonstrating superior computational speed.

Ultimately, the Isolation Forest method was chosen due to its ability to remove at least 3,000 records in each iteration, efficiently eliminating a substantial percentage of outliers. This approach outperformed the others in terms of computational efficiency, making it the preferred choice for processing large datasets for anomaly detection

3) Feature Selection: In the pursuit of optimal feature selection, traditional techniques like PCA, Variance Threshold, and Correlation Matrix analysis proved suboptimal due to limitations in handling target columns and high-dimensional datasets. Consequently, the 'Model-based Feature Selection' approach was chosen, relying on importance weights obtained from a Random Forest Regressor model. This choice was deliberate, given the algorithm's robustness and effectiveness in complex datasets.

The Model-based Feature Selection process involved training the Random Forest Regressor on the dataset after excluding noisy pad data. By leveraging the model's ability to assess feature importance, this approach allows for a nuanced analysis, essential when feature-target relationships are significant. Detailed parameter selection for the model will be discussed in subsequent sections of this report.

Feature importance analysis for X and Y coordinates led to a threshold criterion of 0.001, guided by graphical representations. Features below this threshold were marked for removal. Comparative analysis revealed overlapping low-importance features in X and Y predictions, leading to their strategic exclusion. Evaluations with various thresholds identified 0.0015 for X and 0.0036 for Y as the most effective. This approach resulted in retaining 24 columns and optimizing predictive accuracy.
Figures 6 and 7 show the significance of 'tmax' attribute columns in predicting X and Y. Red-highlighted columns in both predictions will be removed, except for those that are blue in Figure 6 and red in Figure 7, based on the current analysis stage.

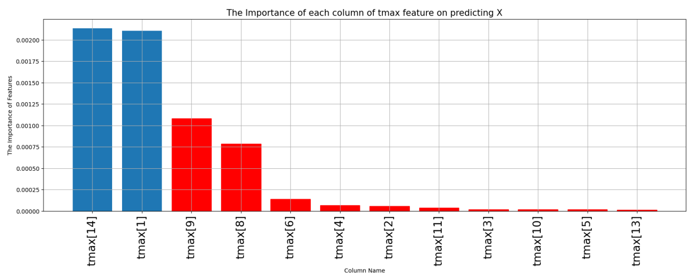

Fig. 6. Bar chart Importance of each column of tmax feature for predicting X

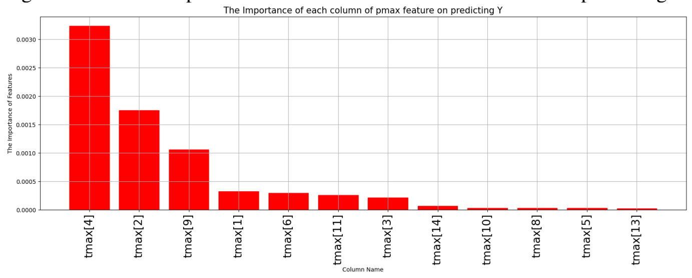

Fig. 7. Bar chart of Importance of each column of tmax feature for predicting Y

4) Scaling: In the context of feature scaling, three common techniques are discussed: Min-Max Scaling, Decimal Scaling, and Standard Scaling.

Min-Max Scaling: Min-Max Scaling normalizes data to a specific range, typically [0, 1], preserving relationships between data points. However, it can be sensitive to outliers and might not handle negative values effectively.

Decimal Scaling: Decimal Scaling involves moving the decimal point to scale data, preserving data distribution. It requires the selection of a suitable 'n' value, which determines the scale.

Standard Scaling (Z-score Normalization): Standard Scaling [3] transforms data to have a mean of 0 and a standard deviation of 1, making it suitable for normally distributed data. It is robust against outliers and can handle negative values effectively. This method is recommended for datasets with more than 380,000 rows and 36 columns, as it aligns well with various machine learning algorithms. This method was employed throughout the project's lifecycle to train each model using a pipeline.

B. Model Selection

1) K-Nearest Neighbors Regressor: The KNN Regressor [4], a non-parametric model, excels in handling datasets with discrete and limited value ranges. Operating on the nearest neighbor algorithm, it predicts outputs based on the similarity of data points in feature space. This simplicity in concept and effectiveness in execution make it an ideal choice, especially for datasets with clear patterns or clusters. Its suitability for scenarios with bounded and discrete values in target columns further enhances its appeal in machine learning applications, prioritizing interpretability and straightforward implementation.

2) RandomForest Regressor: The RandomForestRegressor [5] is renowned for processing large and high-dimensional datasets through ensemble learning. It builds multiple decision trees, combining their predictions for a comprehensive data analysis. Notably, its feature extraction capability identifies the most informative features, crucial for large datasets, ensuring only relevant features contribute to predictions. The model's robustness, versatility, and adeptness with diverse datasets make it a trusted choice in predictive modeling. It underwent two training phases in the project: one for determining feature importance and another for predicting target columns.

3) MLP Regressor: The MLPRegressor (Multi-Layer Perceptron Regressor) [6] is a model based on neural network architecture. It excels in managing intricate, non-linear data patterns, which is a common challenge in large-scale datasets. The model's neural network structure allows for the handling of complex relationships within the data, providing the capability to model intricate patterns that simpler models might miss. Additionally, the MLPRegressor incorporates advanced optimization techniques like L2 regularization to prevent overfitting, ensuring that the model remains generalizable to new, unseen data. Its capacity to manage high-dimensional data and complex relationships, along with built-in mechanisms to maintain model robustness, make the MLPRegressor an essential tool for advanced data analysis tasks.

C. Hyperparameters tuning

Since tuning operations for datasets of this scale can be challenging and require significant computational power, we initially conducted this process on a smaller subset of the data. The parameters obtained from this preliminary tuning did not yield satisfactory results. Therefore, the parameter selection was carried out empirically, guided by existing knowledge about each parameter's impact on the model. This approach involved a trial-and-error method.

1) K-Nearest Neighbors Regressor: In the KNN Regressor algorithm, the parameter selection process was critical for optimizing performance on a dataset with with more that 380000 rows and 36 columns.

1) **Number of Neighbors (n_neighbors)**: The number of neighbors was set to 200, chosen after testing various values. This higher number helps to incorporate a broader range of data points, enhancing prediction accuracy for the large dataset. The number of neighbors was set to 200, chosen after testing various values, and it was observed that larger values led to overfitting.
2) **Weights (weights)**: The 'distance' parameter was selected over 'uniform', enabling the model to give more weight to nearer neighbors, thus capturing more nuanced data variations.
3) **Algorithm (algorithm)**: The 'brute' method was chosen, suitable for large datasets due to its brute-force search approach, albeit computationally demanding.
4) **Distance Metric (p)**: The distance metric was set to 2, corresponding to the Euclidean distance.

These parameters were carefully selected to strike a balance between computational efficiency and the ability to handle the complexity and size of the dataset.

2) Random Forest Regression Parameters: The Random Forest Regression algorithm, implemented through the RandomForestRegressor model, was carefully configured to optimize its performance in predictive modeling.

1) **number of esimators (n_estimators):** Then n_estimators parameter determines the number of trees in the forest. For this model, a value of 100 was selected based on experimentation. Lower values led to reduced accuracy, while values higher than 100 only marginally improved model performance.
2) **Maximum number of Features (max_features):** The max_features parameter controls the number of features considered when searching for the best split in each tree. The chosen value is the square root of the number of columns ('sqrt'), ensuring effective model training.
3) **max_depth, min_samples_split, min_samples_leaf:** These parameters govern the depth and behavior of individual trees in the forest.
    a) **maximum depth for each tree (max_depth):** It sets the maximum depth of each decision tree. A value of 40 was chosen to avoid overfitting observed with higher values.

b) **minimum sample split (min_samples_split):** This parameter determines the minimum number of samples required to split an internal node. A value of 10 was selected for a balance between tree complexity and accuracy.

c) **minimum sample leaf (min_samples_leaf):** It specifies the minimum number of samples required to be at a leaf node. A value of 5 was chosen to prevent over-pruning and maintain predictive accuracy.

These parameter configurations were established through rigorous testing, aiming to strike a balance between model complexity and accuracy which is lower distance. The selected values are expected to provide optimal predictive performance for the given dataset.

3) MLPRegressor Model Parameters: the MLPRegressor model's effectiveness is highly dependent on its parameter configuration. The following parameters were meticulously evaluated:

1) **Hidden Layer Sizes (hidden_layer_sizes):** This parameter defines the number of neurons in each hidden layer of the model. Three configurations were explored:
   - **(64, 32):** This configuration led to a simplistic training regime, indicating an insufficient level of complexity.
   - **(128, 64):** Demonstrated optimal performance, effectively managing the requirements of both target columns. Consequently, this configuration was selected.
   - **(256, 128):** While tested, this setting resulted in excessive complexity, causing the model to overfit.
2) **Learning Rate (learning_rate:)** Due to variability in iteration counts, selecting a specific numerical value is impractical. Thus, the optimal choice for the learning rate parameter is 'adaptive.' This is preferred because it adjusts the learning rate after stages where the model struggles to minimize the loss.
3) **Activation Function (activation):** The activation function in hidden layers ([7]) was initially expected to favor ReLU but was found to be more effective with the `tanh` function. The choice of `tanh` is driven by its effectiveness in handling instances where records acquire negative values after standardization. Its zero-centered nature and saturation characteristics potentially act as a regularizer, contributing to superior performance.
4) **Maximum Iterations (max_iter):** This parameter indicates the maximum number of iterations for the solver. The standard value is set to 100; however, given the extensive dataset:
   - Adjusted to **200**, an increase was necessary as it was observed that the loss for the target columns did not consistently reach its minimum within the initial 100 iterations.

III. Results

In the results phase, we employed the tuned parameters discussed in the tuning section to evaluate the performance of our machine learning models. The objective was to assess their predictive accuracy and select the most suitable model for our dataset. Here are the outcomes of our analysis.

Model	**Euclidean Distance**
KNNRegressor	7.79
RandomForestRegressor	4.98
MLPRegressor	4.32

TABLE I
PERFORMANCE OF MODELS ON THE EVALUATION SET.

From Table I, it is evident that the KNNRegressor model did not yield satisfactory results. However, the other two models, particularly the MLPRegressor, performed well on the evaluation dataset. Despite this, the RandomForestRegressor model exhibited the shortest training and prediction times among the models. The MLPRegressor model undergoes a varying number of iterations in each stage for both X and Y, resulting in different loss values at the end. However, the average loss for the X column is approximately 4.7, while for the Y column, it is around 6.5, and The recorded Euclidean distance in the table indicate that the MLPRegressor model has achieved its best performance on the dataset.

IV. Discussion

In a broader context, achieving the lowest possible Euclidean Distance relies on the successful implementation of methods with high accuracy and discerning capabilities for extracting valuable insights from noisy and outlier-laden datasets. A relevant example is the pursuit of an approach that maximizes data quality while simultaneously removing a substantial quantity of outlier data in high-dimensional datasets. These efforts contribute to optimizing overall performance metrics. Additionally, investigating methodologies that align model predictions with specific coefficients (in this project, both target columns have coefficients of 5) shows potential for speeding up execution times and enhancing problem-solving effectiveness.

References

[1] E. Schubert, A. Zimek, and H.-P. Kriegel, "Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection," *Data mining and knowledge discovery*, vol. 28, pp. 190–237, 2014.

[2] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008.

[3] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*. Springer, 2015.

[4] Y. Song, J. Liang, J. Lu, and X. Zhao, "An efficient instance selection algorithm for k nearest neighbor regression," *Neurocomputing*, vol. 251, pp. 26–34, 2017.

[5] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[6] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *IEEE International Conference on Neural Networks*, pp. 586–591, 1993.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.