DSL Winter Project Report 2024

Minal Jamshed & Ehsan Dashti
Politecnico di Torino
Student id: s329091 & s316511
s329091@studenti.polito.it
s316511@studenti.polito.it

***Abstract*—In this project, we aimed to predict the coordinates of the particles using signals captured by a sensor equipped with 12 pads. Our first step involved identifying and eliminating noisy columns from the dataset through a combination of statistical analysis and cross-validation, resulting in the removal of thirty columns deemed as noise. We then selected the Random Forest Regression algorithm due to its suitability with our problem overview and thoroughly fine-tuned its hyper-parameters. Lastly, we validated our model on a separate evaluation set, which provided promising results on the leader board, showing the effectiveness of our methods in solving the particle coordinate prediction problem, landing us a score of 4.736.**

I. Problem overview

Statement: In the field of particle physics detecting the exact coordinates of the particles is a major challenge which can be overcome by employing certain sensors like Resistive Silicon Detector (RSD). These sensors are equipped with 12 pads that generate certain signals when a particle passes over them and by the evaluation of those signals, the sensors are able to detect the exact coordinates x and y, of the particles. In this project we are given a dataset that is comprised of 514,000 events (passage of a particle through the sensor) out of which 385,500 events are in the development set and 128,500 in the evaluation set. A total of 18 readings of every feature are provided for each event due to hardware constraints, whereas only 12 pads generate corresponding signals in the sensor.

Considerations: It should be noted that the features extracted from the sensors are peaks, time delay, RMS, and area, each having values coming from 18 sources. However, it is known that a subset of the 18 values of every feature contains noise. Thus, identifying and removing the noisy columns from the dataset requires careful consideration in this project.

Employed Techniques: Our aim is to build a data science pipeline which would take these extracted features from the signals: **pmax, negpmax, tmax, area and rms** as inputs and predict the x and y coordinates(in µm) of the particles. Various statistical techniques, followed by feature engineering and selection of the most accurate algorithm to train the model are essential points to be considered for the final accuracy of the result. Moreover, we are required to use a Multi Output Regressor as for each input we are predicting two output coordinates x and y.

II. Proposed approach

A. Preprocessing

Here, we define the preprocessing steps undertaken to prepare the dataset for effective model training and evaluation by conducting a thorough analysis to understand its structure, content, and any peculiar anomalies. The focus was to ensure that the dataset is correctly processed to maintain its quality by implementing the following steps in phases:

The initial phase involved loading the dataset and understanding the features and their relationships with each other and with the target values.
• **Missing Values** - we began by checking for any missing values in the entire dataset to proceed with the necessary actions. However, there were no missing values found thus, no such changes were required in this step.
• **Nature of Values** - we checked if all the columns contained numeric values in order to process them accordingly using Pandas, and found all of them to be appropriate float values for the Pandas library; besides the header columns.
• **Cross Validation** - by transforming the entire dataset into a data frame we were able to visually examine if Pandas correctly read the entire development set's values.

After the basic initial checks were completed we proceeded to carry out the statistical analysis of the entire development set to better understand the values and make inferences from it. As, it was evident from the problem statement that six of the values for all features were noise due to the hardware constraints of the sensor, the second phase involved cleaning the dataset of noise.
Identifying the six noisy columns for each feature required a thorough statistical analysis. We began by finding the statistical values: mean, median, standard deviation and variance, for every feature given by each pad. From the statistical summary of the features, it was easy to distinguish that certain columns for every feature had anomalies compared to the other columns, thus indicating them to be noisy columns. However, to predict the noisy columns with assurance, we found out the correlation of every feature with the target coordinates given. The correlation values clearly demonstrated that the promising columns identified earlier by the statistical values were indeed noisy columns. Thus, we concluded that the columns 0, 7, 12, 15, 16 and 17 of

every feature, were indeed noise and not values from the sensor pads. We had also validated our decision later by training the model with the entire development set (noisy columns included) and then by removing the noisy columns to compare the results, which were positively indicating that the selected columns are indeed noise. From the five features, RMS did not show much correlation with the target values thus showing least contribution in predicting the values. Also, as evident from the detailed statistical values calculated in python, column 2 depicted to be noisy for the feature negpmax whereas pmax, tmax and area showed relatively higher correlations and better statistical values for column 2 to be considered noisy. However, we did further evaluate the columns by training the model on different combinations of noisy columns including and excluding column 2, which further strengthened our claim of proving that the values received by features for columns 0, 7, 12, 15, 16 and 17 are indeed noise and not from any of the twelve pads.

The third phase included feature engineering to better allow the model to learn new insights from the dataset and enhance its predictive capability. We researched and analysed that various techniques could help the model to acquire deeper insights, thus we tried and tested numerous methods that involved aggregating existing features, creating their statistical combinations and making polynomials, to help the model extract meaningful information from their distributions and influence its accuracy in predictions. Therefore, we tried different feature combinations in training the model and checked the performance of it in every iteration and recorded the score on the leaderboard. The nature of our model is very robust and efficient as it provides feedback on the features that have majorly contributed to the accurate prediction of values, and by studying the feedback given by our model we further tested by adding and removing more statistical combinations until we found the best ones. We examined that the model performed better with squared and cubed features suggesting that it captures useful information from the quadratic and cubic relationships in the dataset. This iterative approach demonstrated the importance of vigilant data quality assessment in the preprocessing framework.

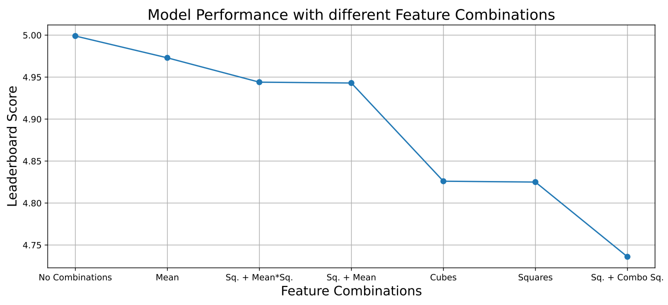

Fig. 1. Comparison among different Feature Combinations

As, we had already identified the noisy columns and significantly reduced the number of feature columns we further did not reduce the dimensionality of the features so that the model gets complete insight from all the pad values to better predict the coordinates in the evaluation set. Also, our model was carefully selected such that no complexity is incorporated into training it, therefore no normalization or standardization was required of the features.

B. Model selection

After carefully studying the problem statement and the nature of our development set, we rigorously evaluated various predictive regression models to select the most suitable model for our project. From our analysis we concluded the selection criterion to be based on the model's capability to handle complex datasets, its interpretability, behaviour with the types of datasets, training duration, tuning capabilities, and computational efficiency. The models we assessed along with their key features are:

TABLE I
COMPARISON OF MACHINE LEARNING MODELS FEATURES

Features	**Random Forest**	**Neural Networks**	**Linear Regression**	**Support Vector Machines**
Complexity	Efficient	High	Low	High
Interpretability	High	Very Low	High	Moderate
Data Type	Efficient with complex datasets	Needs larger datasets	Efficient for linear relationships	Efficient when features $>$ values
Training Time	Relatively fast	Relatively low	Fast	Moderate
Tuning Complexity	Relatively easy. Fewer hyperparameters	Very complex Many hyperparameters	Low	Moderate to high
Resource Usage	Moderate	High	Low	Moderate to high

Thus, by thoroughly studying each model's capability in handling such datasets we contemplated between choosing Neural networks and Random Forest as the final model for our project and by comparing both the regressors in different domains we concluded by selecting **Random Forest Regressor** as our final model. Since, it is more interpretable than the other models, as its hyperparameters can provide insights regarding its structure. It is efficient in handling complex datasets and is robust to over-fitting. With regards to other features, it's hyperparameters are relatively easy to tune and it takes comparatively less time to train. However, in weighing its pros and cons, Random Forest emerged out to be a more efficient model for our problem than the other models in handling the dataset with reasonable computational power, which is further backed up by the results of our experiments.

C. Hyperparameters tuning

The hyperparameters tuning stage was very crucial in optimizing the model's efficiency and accuracy in predicting the target coordinates. We experimented with various parameters and tuned them iteratively, noting down the performance of our model in every iteration and how

accurately it predicted the results.
We tried to balance between the exhaustiveness of the search and its computational feasibility. We first trained our dataset by using the efficient technique of **GridSearchCV** to explore the range of values that generated favourable results. Since gridsearch is a comprehensive algorithm where it checks combinations of parameters and cross validates every combination, it is computationally very exhaustive and thus limited its usage on a wider range of hyperparameters. However, after getting a baseline from gridsearch, we further experimented those ranges using a lesser computationally exhaustive algorithm called **RandomizedSearchCV** and further explored wider ranges of parameters. After we evaluated the performance of our model in different iterations, we then refined the model by setting the parameters manually and recorded its performance on the leaderboard.

For our particular random forest regressor the parameters that we chose to train our model upon included:
1. **n_estimators** - Number of decision trees to be made
2. **max_features** - Number of features selected for the best split during the tree creation
3. **max_depth** - How deep a tree should go
4. **min_samples_split** - Minimum number of features required to split an internal node
5. **min_samples_leaf** - Minimum number of features required to be at a leaf node
6. **max_leaf_nodes** – Maximum number of terminal nodes in a tree

Out of these six hyperparameters, we carefully analyzed how a certain change in each parameter affected the quality of our model and thus by iterating different values and re-training our model multiple times we found the favourable parameters.

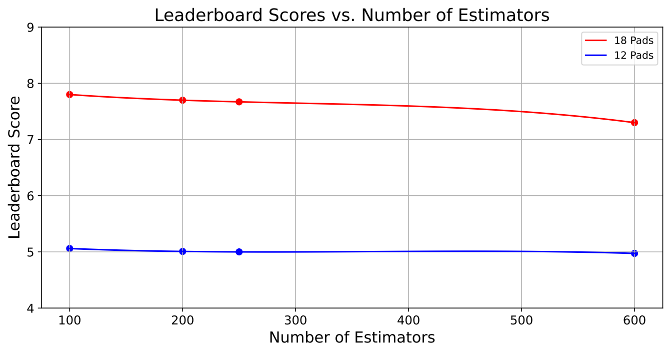

Fig. 2. Impact of number of Estimators

Initially, we noted down the performance of the model on the development set by splitting it into training and test sets and calculating the performance metrics like Mean Squared Error (MSE) and R-squared (R^2) score. These metrics provided quantitative measures to analyse and compare the performance of our model with different hyperparameter combinations. Once we became confident with a certain range of hyperparameters, we stopped splitting our development set and trained our model on the entire dataset to get the complete insights from the values to better predict the coordinates in the evaluation set.

Summarizing the above preprocessing steps; the chosen methods that includes data preparation, choosing the most efficient model, and fine-tuning its hyperparameters, was thoroughly executed with extensive research and analysis ranging over a number of experiments to meet the desired results of the project. The cautious steps taken in every part of the process helped us design an efficient model which is capable of predicting the target coordinates quite efficiently.

III. Results

After the thorough evaluation and working on all of the above-mentioned processes for the final evaluation of our machine learning project; we methodically measured the performance by using a robust iterative process and a systematic approach.

Hyperparameters Evaluation: We evaluated the model's performance on multiple hyperparameters and got the desired results with the following values:
- n_estimators = **700 trees**
- max_features = used **square root** of all the features
- max_depth = **50 depth** of each tree
- min_samples_split = **2 features**
- min_samples_leaf = **1 feature**
- max_leaf_nodes = **NONE** (used all of the feature combinations)

Since we had earlier explored the desired ranges for the above hyperparameters using optimized algorithms, in the final solution we manually deployed these parameters to train the model in efficient time. We observed that the best results given by the model was when the number of trees increased up to 700, however further increasing the range required more intensive computational resources and thus we could not experiment ahead. The depth of the trees showed an increase in the performance up to the depth of 50, by increasing the depth further it showed no improvement thus indicating that saturation was achieved. Concerning the best features used in creating a decision tree was the method of taking square root of all the features, since all other methods provided a higher score on the leaderboard given the same estimators and depth of the trees.
We observed that the minimum number of features required to split an internal node and to be at a leaf node should be kept as low as possible, by increasing the values of the features, the performance of the model degraded, thus the best numbers we selected were 2 features to split an internal node and 1 feature to be at the leaf node.
We had refined our model through hyperparameter tuning by leveraging evaluation metrics such as **Mean Squared Error** (MSE) and the **R-squared** (R^2) score. Since we were

using the entire development set for optimized training of our model, the metrics of MSE and R² were predicted using the performance of our model on the same training set, which however does not provide the true image of how the model would have performed on the test set. But a good score of MSE and R² shows that our model had been trained quite accurately and this insight has helped us in optimizing our model to achieve better accuracy. By evaluating the final performance of our model on the evaluation set and comparing with the results on the leaderboard we further analysed our performance.

Training Duration: The model's training duration was bench-marked to ensure the efficiency of our algorithm within the computational constraints provided. It took 55 minutes for the model to get trained and evaluate the test set.

Final Evaluation on the Leaderboard: We believe that our model is more accurate than the model used to achieve the baseline score in the leaderboard by comparing the distance score, since it can be seen that a lower score reflects better prediction of the coordinates by the model. Our model has achieved a Euclidean distance score of **4.736** on the leaderboard, surpassing the baseline requirement of 6.629.

Baseline Score = **6.629**
Our Score = **4.736**
Improvement Percentage = $\left(\frac{6.629-4.736}{6.629}\right) \times 100 \approx$ **28.55%**

Remarks: This shows a significant improvement in prediction, reducing the average distance error by approximately 28.5% relative to the baseline model. The scores achieved by others in the leaderboard range from 3.77 to 200s, which we believe depicts that our model has performed quite well in predicting the coordinates of the particles.

IV. Discussion

Upon the conclusion of our project, we can reflect on several key points that contributed to the model's performance. The application of our solution to the problem has yielded insightful conclusions and highlighted areas of success as well as potential improvements.

Successes:
• While assessing the noisy columns we did excessive examination on the statistical analysis of the dataset which then revealed an anomaly within column 2. We found that column 2 showed noisy characteristic for the feature negpmax whereas, for all the other four features it did not exhibit any different behaviour than the other columns, thus in order to verify whether column 2 was just a coincidental anomaly in the dataset or a noisy column, we further re-trained our model by replacing column 2 with the noisy columns that showed the least correlation with the target coordinates. According to our analysis, columns 0, 7, 12 and 15 showed similar noisy behaviour for all of the 5 features, however column 16 and 17 depicted a peculiar deviation for negpmax and tmax. Thus, the ability to identify and correct anomalies had further enhanced the model's predictive performance.
• The project's success is also highlighted by the model's capability to predict with a degree of precision that surpassed our initial experiments. The implementation of RF algorithm proved effective, owing to its robust nature in handling complex datasets. Using optimized techniques and exploring the hyperparameters space enabled us to choose the ideal parameters to achieve accuracy. Through a rigorous process of data preprocessing, including the experimentation with the anomalies of column 2, testing and experimenting multiple combinations of features, we had observed a notable improvement in our model's performance. This is quantitatively proven by the improved leaderboard score, which initially gave us a 7.9 and with multiple fluctuations finally landed us with our best improved score of 4.736, indicating a closer proximity to the accurate target coordinates.

Improvements & Limitations:
Despite these achievements, the project also presented us with the opportunities for improvement. The time taken to train the model iteratively by tuning hyperparameters and experimenting with different feature combinations, resulted in a significant amount of time, which depicts that efficiency could be further enhanced, possibly by opting for more advanced computational strategies or optimization of algorithms. We had also tried different coding platforms to improve computational efficiency, however various platforms demanded changes in the script compatible to their libraries and thus increased the Leaderboard score to 200+. Since, GridSearchCV and RandomSearchCV were both time and resource-intensive in model training, it limited us to explore a wider range of hyperparameters.

Alternate Solutions:
While the Random Forest algorithm proved to be suitable for our project, exploring additional machine learning models like Neural Networks, might lead to improved results. Although Neural Networks require a larger dataset combined with the complexity of tuning hyperparameters due to the numerous parameters involved, demanding considerably more computational power for fine-tuning and training, their integration could enhance overall performance.

Considerations:
In contemplating the problem, it became clear that the accuracy in results depends on the quality of the dataset the model is trained on. Thus, continuous enhancement of data preprocessing steps is mandatory. Moreover, thorough consideration should be given in balancing the model's complexity and its computational efficiency, to ensure that the model not only makes accurate predictions but is also scalable and practically viable for real-world applications.

References