Shell .ai Hackathon for Sustainable and Affordable Energy 2025

Fuel Blend Properties Prediction Challenge:

Team Siddharthians

Team Members

1. Eshwar Vijay Dheeraj Kollati
2. Pothumanchi Bhargav Narendra Raju

Affiliation: Siddhartha Academy of Higher Education

Problem Statement

The challenge is to develop models capable of accurately predicting the properties of fuel blends based on their constituent components and their proportions. These predictions must be precise enough to guide real-world blending decisions where safety, performance, and sustainability are paramount.

Data

We were provided a train.csv which has 65 features which include 10 target properties and test.csv which has 55 features excluding the 10 target features. The 10 target features were regarding the Blend Properties of the aviation fuel.

Evaluation

The evaluation metric which was given is Mean Absolute percentage Error (MAPE) which is guided by the formula:

$$\text{MAPE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} \frac{|y_i - \hat{y}_i|}{\max(\varepsilon, |y_i|)}$$

Where:

y_i = ground-truth predictions

$\hat{y}_i$ = model predictions

$n_{samples}$ = number of data samples

The scores on the leaderboard were calculated using :

$$LeaderboardScore = max\left[10, \left(100 - \frac{90 \times cost}{Reference\ Cost}\right)\right]$$

Proposed Approach

After many iterations of ML and DL architectures, we decided upon a weighed ensemble of deep learning-based architecture (Primarily Real MLP).

We will briefly explain our previous iterations in concise in this page and will detailly explain about our approach in the detailed explanation section.

Since it is a regression problem we first tinkered with gradient boosting techniques such as XGBoost, CatBoost, Light GBM etc.. After some iterations with gradient boosting techniques combined with reasonable feature selection, we hit a plateau at a score of 83 with these boosting techniques. We also tried our luck with tree based methods but we didn't find any much noticeable change in the score.

We then carried out some research on various transformer based models that are hot in the market. We came across many transformer based models that were specifically developed for tabular datasets such as Tab-M, Tab-R, Tab-ICL, TabNet etc. A major drawback with all these transformer based models is that they are resource intensive , require GPU and take up a lot of time for training and inference. We nevertheless tinkered with some of these models as we first wanted to increase our score. TabNet gave us a score of 87 and the remaining models performed similarly. Out of these, one architecture named Tabpfn gave us the best score of 89. We also tried some feature selection for Tabpfn and were able to breach the 90 score level. Even then, the model was incredibly slow and heavy-weight. Also, TabPFN supports a max of 1000 tuples. So, we carried out further more research on various new machine and deep learning architectures that provided high accuracy and were of less weight and low inference time.

We then came across this paper entitled "*Better by default: Strong pre-tuned mlps and boosted trees on tabular data*" [1] which was presented at NeurIPS, 2024.

This paper introduced some lightweight models that provided high accuracy on tabular data. They also provided a scikit-learn based wrapper to implement this pre-weighed models at ease. They provided many models some novel, some already famous which pre-tuned weights (TD). One model that worked really well for us was Real-MLP which is a variation of MLP as seen in Figure 1 below . It gave us a score of 91. On performing further ensembling using method prescribed in Caruana .et [2] and some smart feature selection, we arrived at the final score of 95.2 .

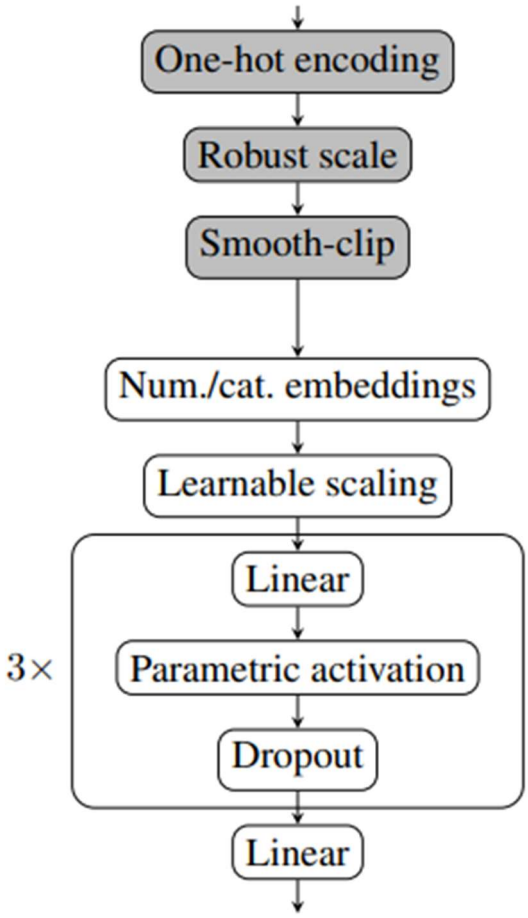

Figure 1 : Real-MLP Architecture

Detailed Explanation :

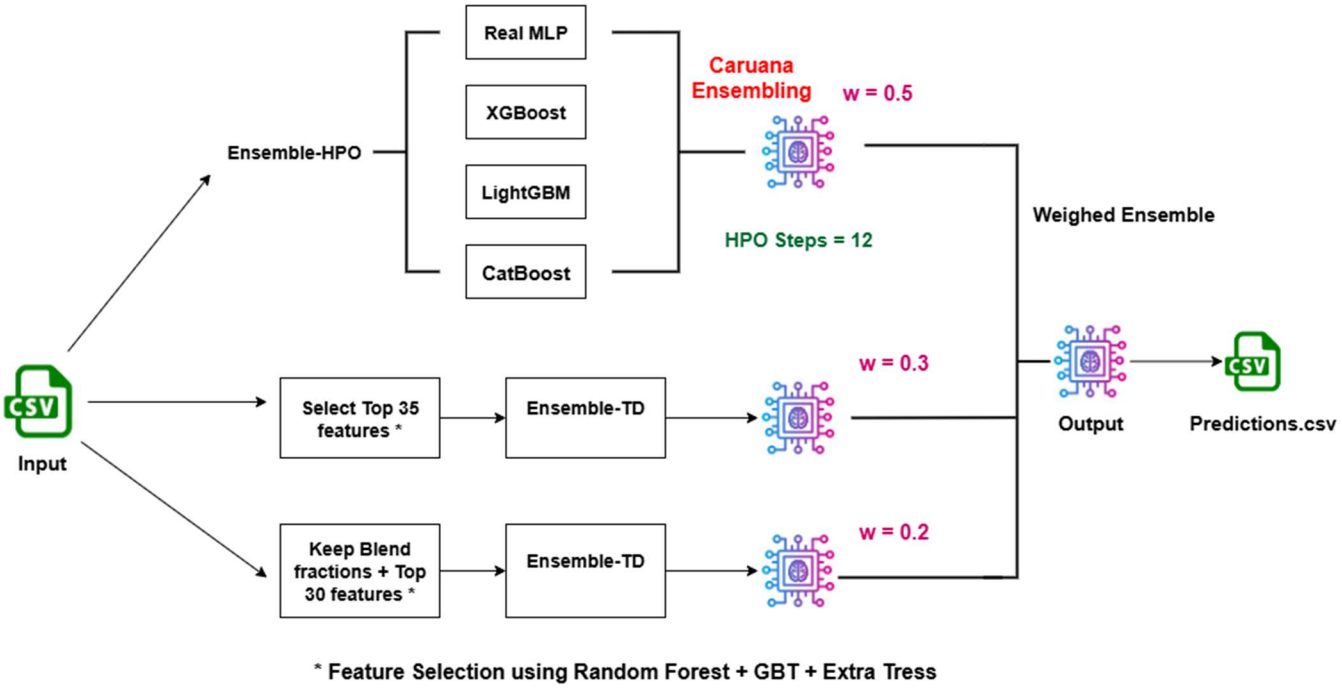

Figure 2 : Our Approach

Figure 2 clearly describes our approach which gave us the highest score in this hackathon. As said before, Real-MLP was our strongest working model. We ensembled this with various models to boost our score.

Real-MLP model was ensembled with XGBoost , LightGBM, CatBoost models using technique described in Caruana et. [2] known as Caruana ensembling. Caruana ensembling is a greedy approach where we iteratively select and weight models from a pool (our chosen model list) based on how much each improves performance on a validation set. The process works like this: start with an empty ensemble, then repeatedly add the model that maximizes validation performance when combined with the current ensemble. Models can be selected multiple times, which implicitly increases their weight. This was done in 30 steps to make our final ensemble very robust. The output of this ensemble is Ensemble-TD.

As our hero base model, we took this Ensemble-TD and performed hyper-parameter optimization using 12 steps of random search in feature space. (12 was chosen as we got a maxima of results at 12. We observed worse results at 15,20 and 10). This output model gave a score of around 94. The output model say Model-1 was given a weight of 0.5 in final weighed ensemble.

For our second model, we did a feature selection of top 35 features. Random Forest + Gradient Boosting Trees + Extra Trees were chosen as feature selectors. These features were then inputted to Ensemble-TD to predict the 10 predictand features. We assure that ensembling a non-feature selected and feature selected model is of no worry as they ultimately predict the same target variables. They were not exposed to test data so there is no issue of data leakage. This output model say Model-2 was given a weight of 0.3.

For our third model. We intuitively thought that blend fraction had to be an important feature that cant be excluded out during feature selection. So, we took the first 5 input features i.e. blend fractions and then did feature selection of top 30 features from the rest features. Random Forest + Gradient Boosting Trees + Extra Trees were chosen as feature selectors. The output model say Model-3 was given a weight of 0.2 in final weighed ensemble.

The model weights of each of these 3 models was chosen by performing a grid search on all combinations of weights. We choose the combination of [0.5, 0.3, 0.2] as it gave us the best result. This overall ensemble gave us the final score of 95.2.

Major reasons for choosing Real-MLP based models was because it's models were lightweight, didn't have input size constraints like TabPFN and gave us reasonably fast training and inference time.

Model Inference

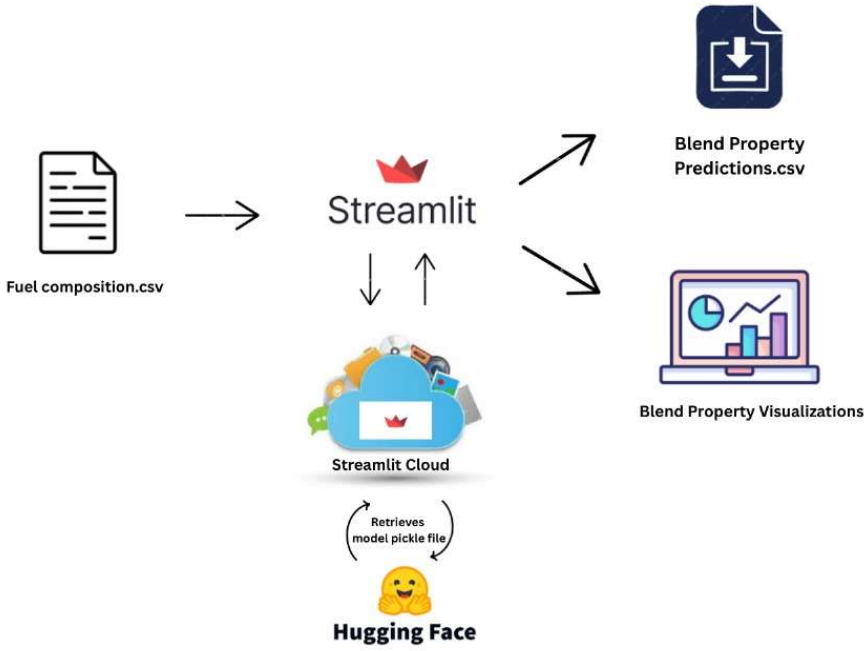

Figure 3 : Model Inference Architecture

The demonstration interface was created using streamlit cloud and the models hosted on Hugging Face space. We didn't use any cloud based services like AWS or Azure as the models weren't that heavy (our models were of max weight 40-50 MB). Hence, Hugging Face space was more than sufficient for our use case.

We created some useful and robust visualizations using Streamlit. Since , it is tiresome to manually type in 55 feature values, we instead provide the option to input csv file to get output predictions. If you need only one single prediction, give the csv file with a single tuple. If you have multiple blends, give the whole csv file.

We provide following visualizations in our interface :

1. Bar gauge (to see the the prediction values). (change ID no. to see for each tuple)
2. Radar chart (for comparisons)
3. 3D scatter plot (to see predictions across 3 axes space components).
4. Predictions chart.

Demonstration Link : https://shellaihacksiddharthians.streamlit.app/

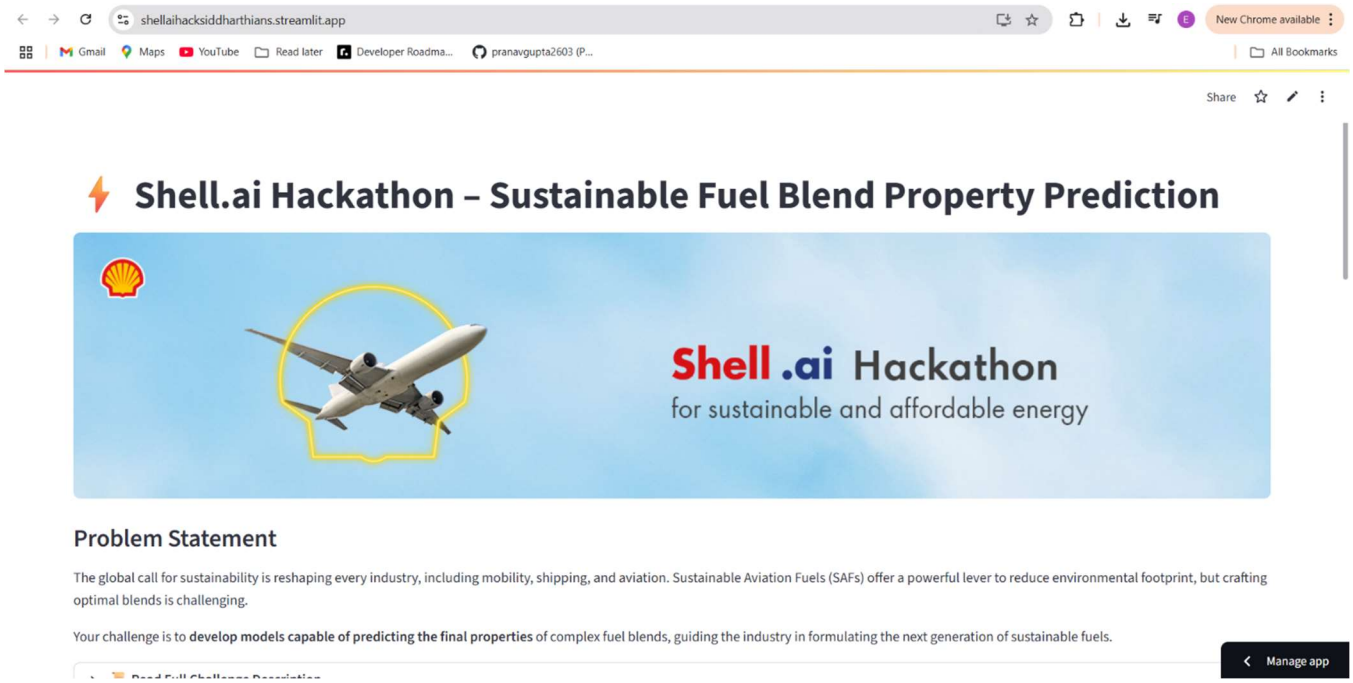

Figure 4 : Interface

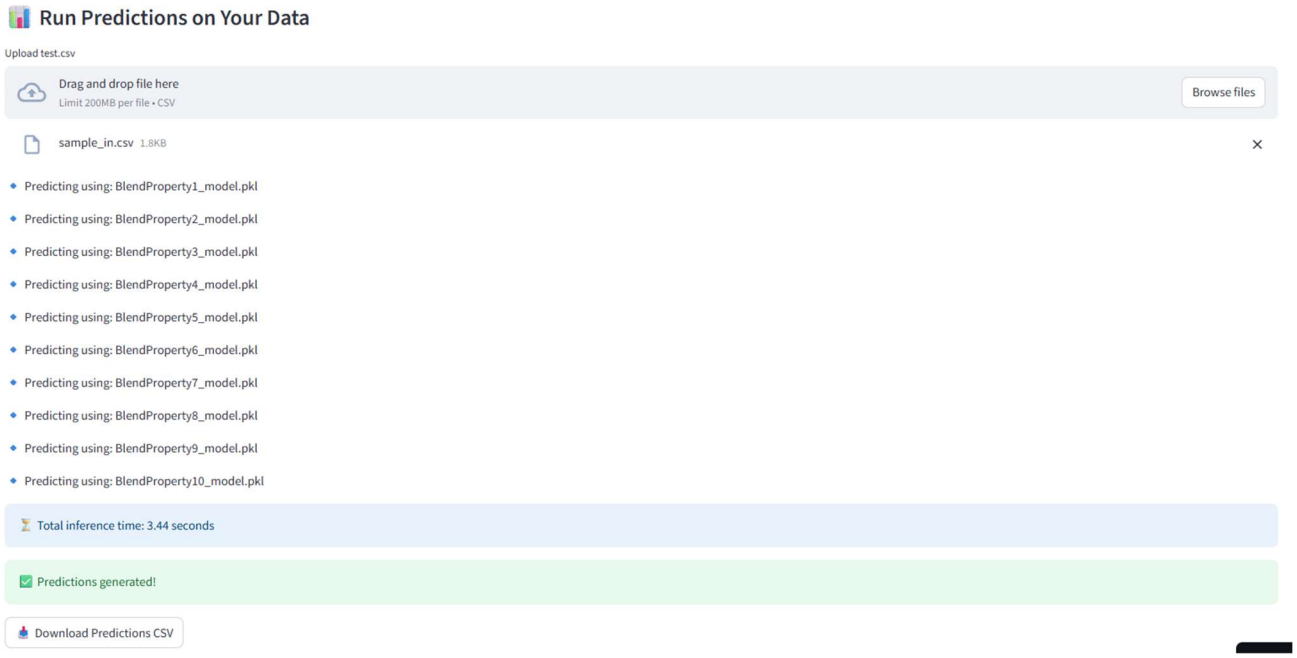

Figure 5 : Input / Output

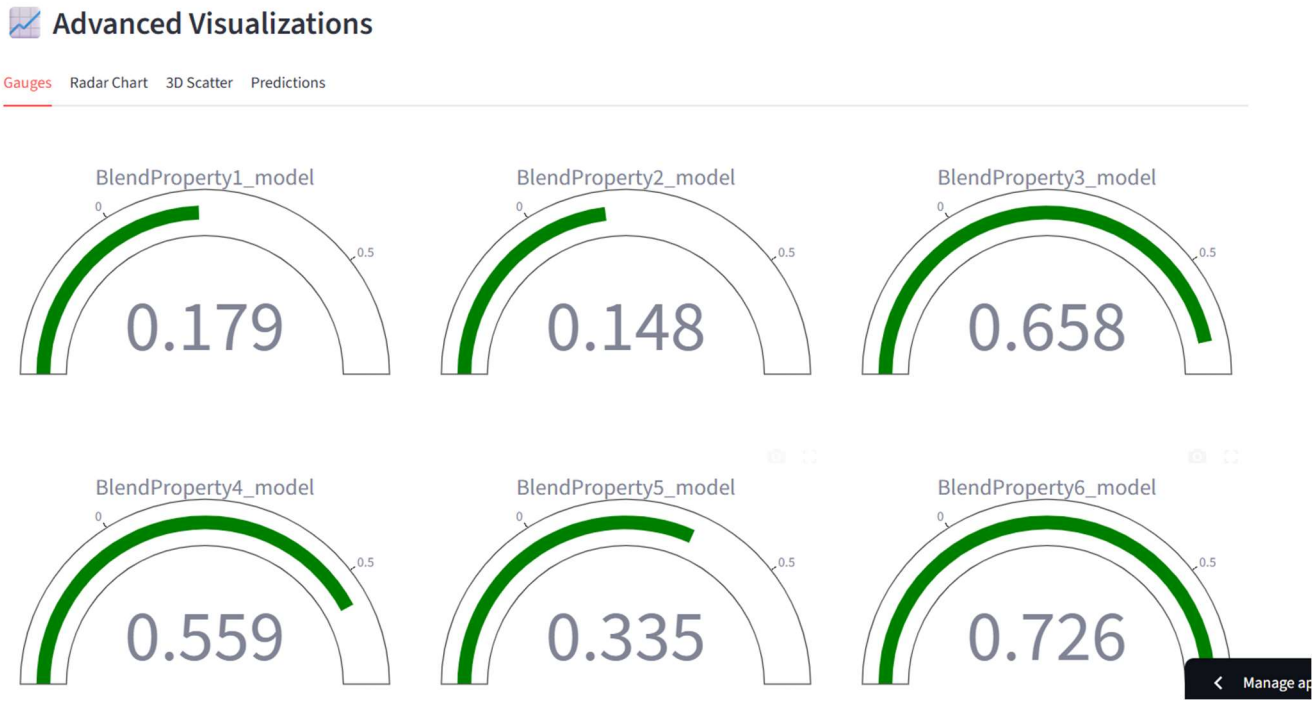

Figure 6 : Gauge Charts

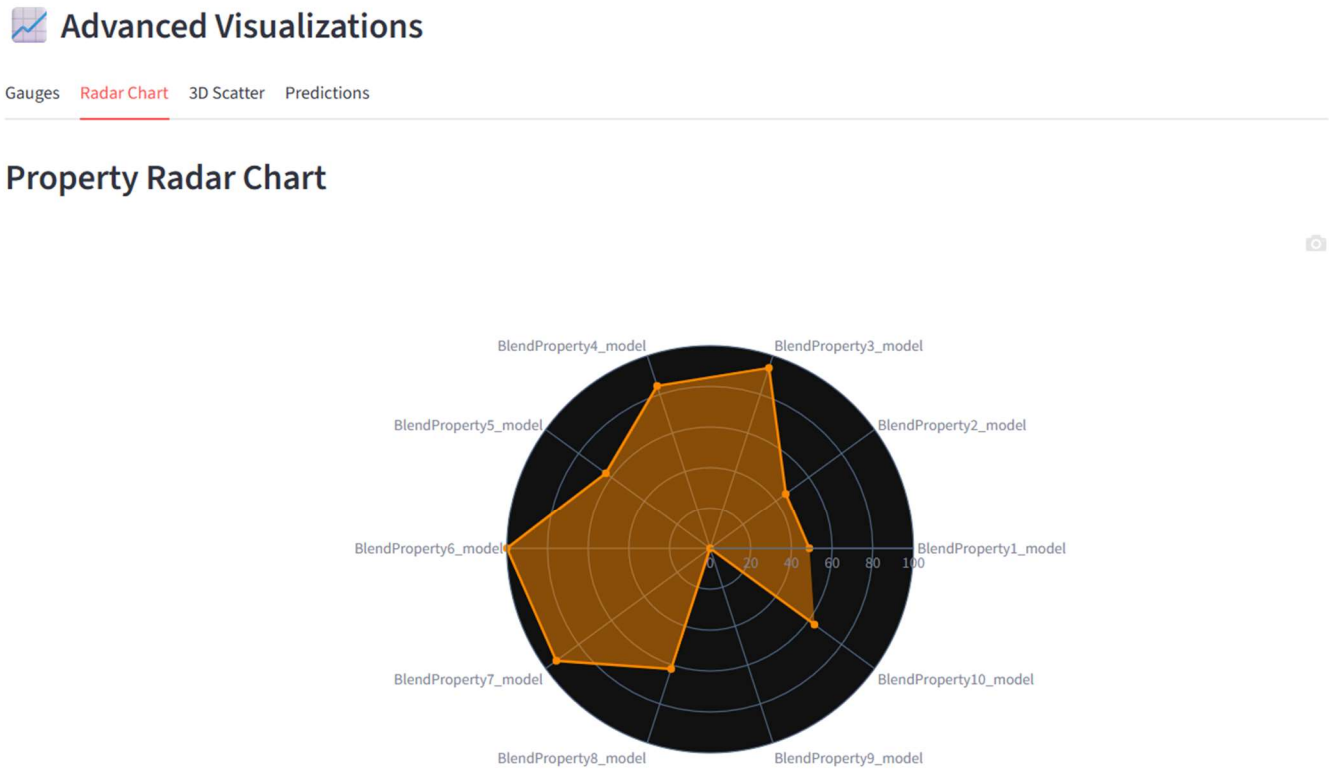

Figure 7 : Radar Chart

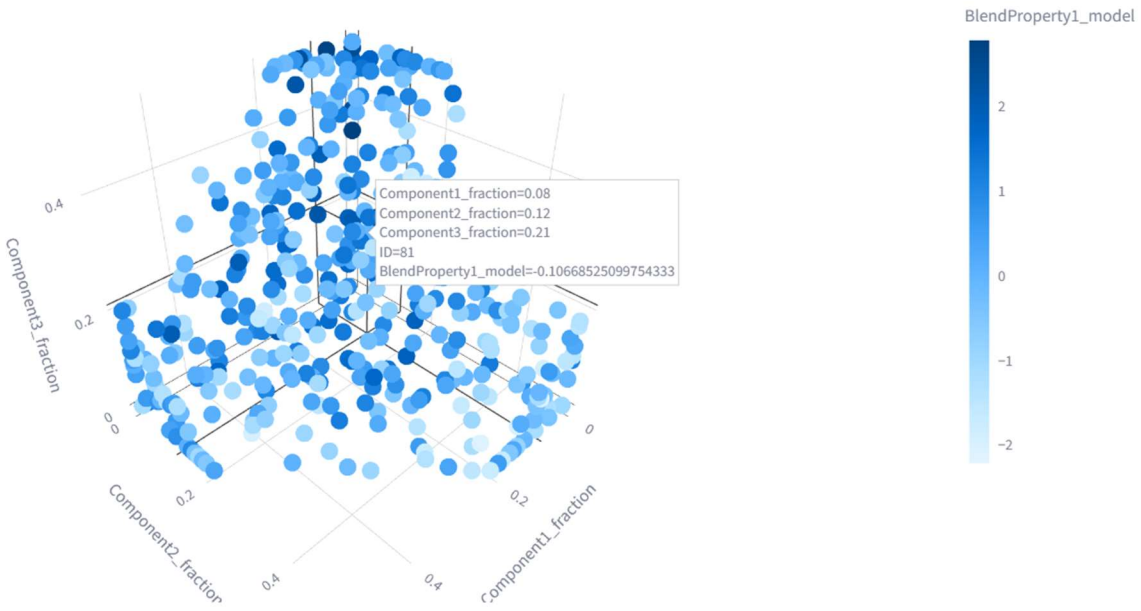

Figure 8 : 3D Scatter Plot

Advanced Visualizations

Gauges | Radar Chart | 3D Scatter | Predictions

	Predicted Value
BlendProperty1_model	0.1789
BlendProperty2_model	0.1484
BlendProperty3_model	0.6579
BlendProperty4_model	0.5587
BlendProperty5_model	0.3346
BlendProperty6_model	0.7259
BlendProperty7_model	0.6566
BlendProperty8_model	0.3196
BlendProperty9_model	-0.3430
BlendProperty10_model	0.3352

Figure 9 : Predictions Chart

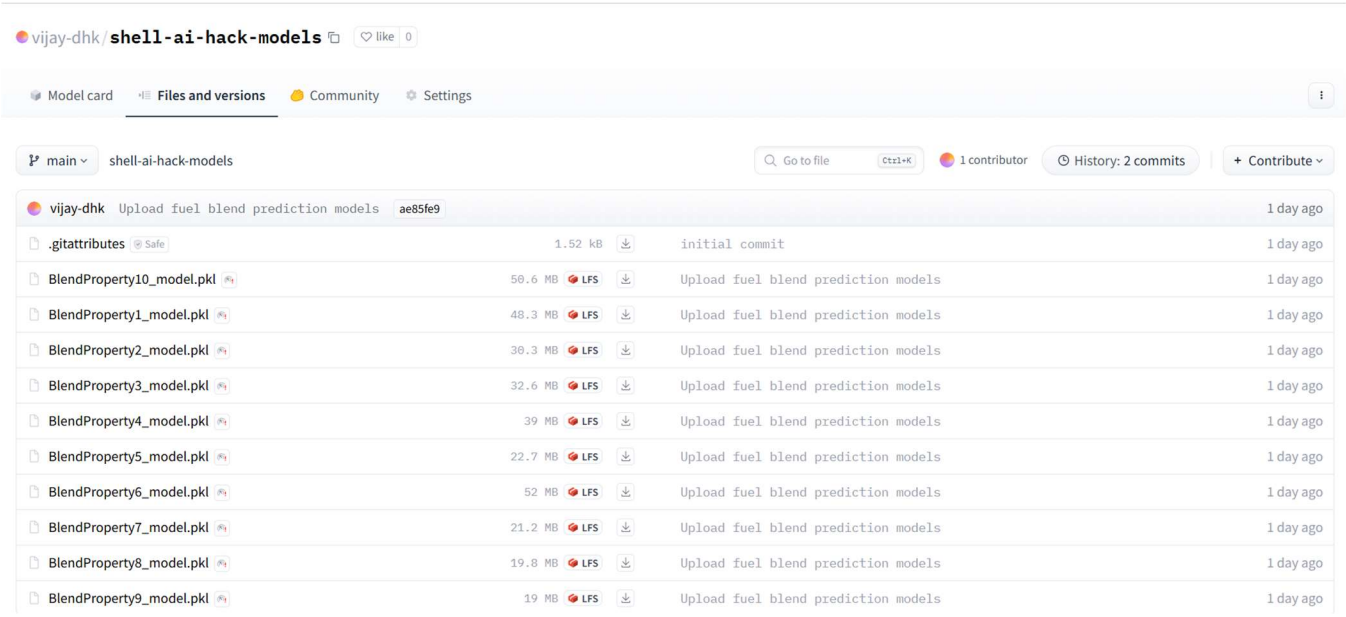

Figure 10 : Model card on Hugging Face

Link : https://huggingface.co/vijay-dhk/shell-ai-hack-models/tree/main

Git-Repo : https://github.com/nabro356/Shell_AI_Hack_Siddharthians

References

[1] D. Holzmüller, L. Grinsztajn, and I. Steinwart, "Better by default: Strong pre-tuned MLPs and boosted trees on tabular data," in *Neural Information Processing Systems*, 2024.

[2] Caruana, R., Niculescu-Mizil, A., Crew, G., & Ksikes, A. (2004). Ensemble selection from libraries of models. *Proceedings of the Twenty-First International Conference on Machine Learning (ICML '04)*, 18.