SHAP Ranking

Principle

$$\phi_i = \sum_{S\subseteq F\setminus\{i\}} \frac{|S|!\,(|F|-|S|-1)!}{|F|!}\left[f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)\right]$$

- F is the set of all features.
- S is any subset of features that does not include feature i.
- | F | and | S | are the number of features in the respective sets.

Meaning

Select a feature subset S that does not contain feature i.

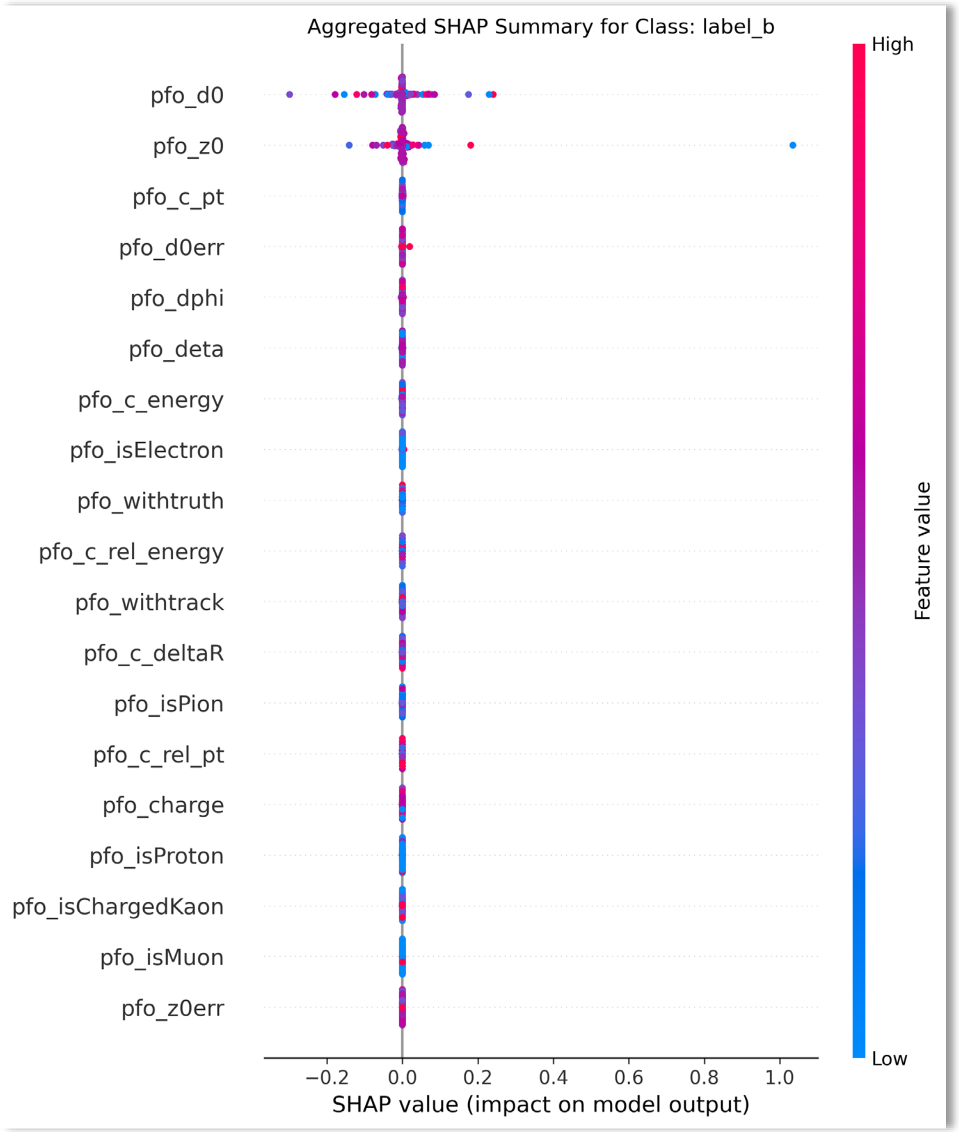
$$\begin{cases} \text{with i} \rightarrow f_S(x_S) \\ \text{without i} \rightarrow f_{S\cup\{i\}}\,(x_{S\cup\{i\}}) \end{cases}$$

The **marginal contribution** of i to S is:

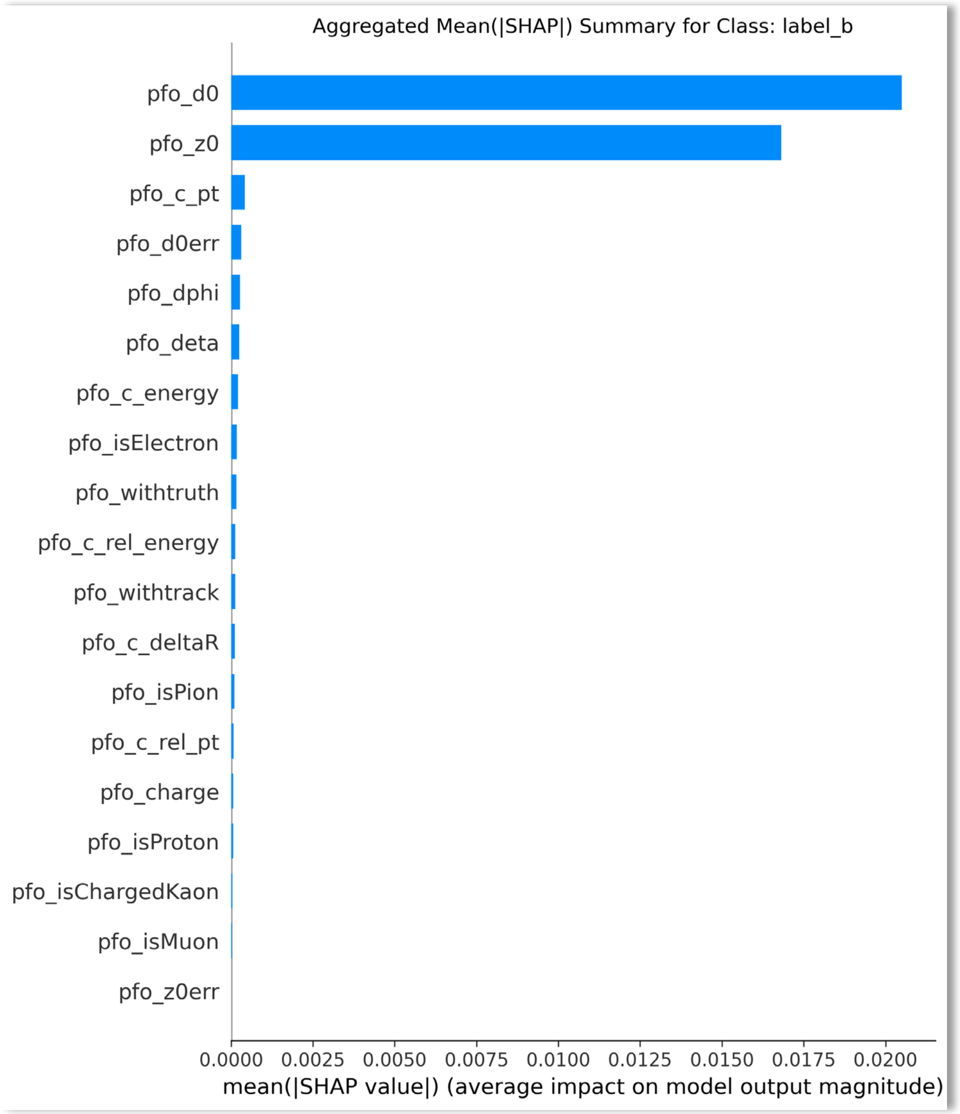
$$f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)$$

By considering all possible feature subsets S, the SHAP value for feature i is the weighted average of its marginal contributions.

Description

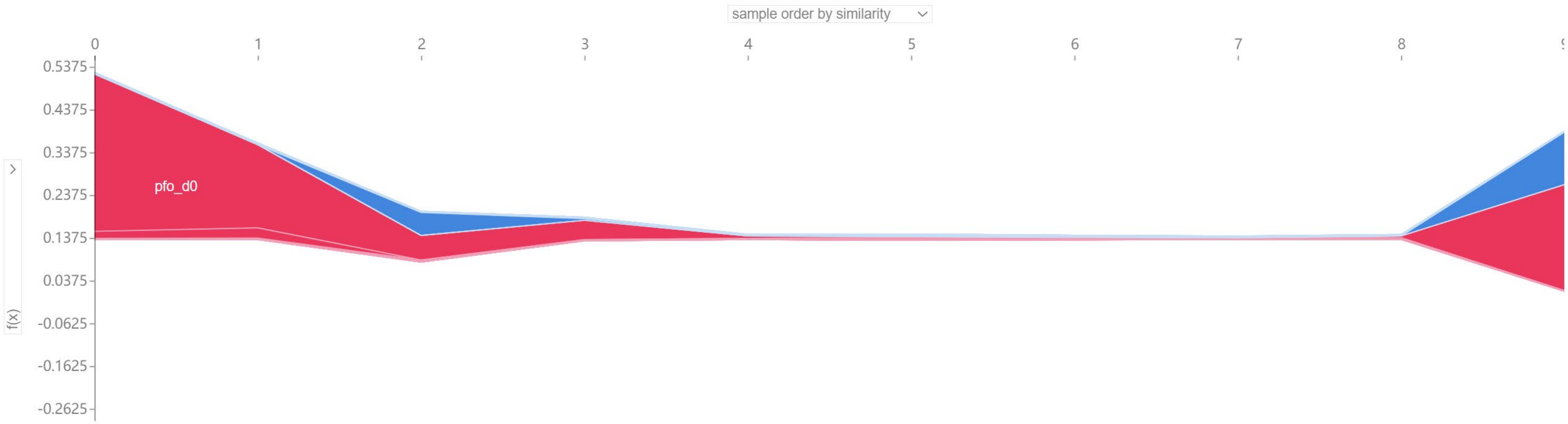


Summary Plot - Beeswarm



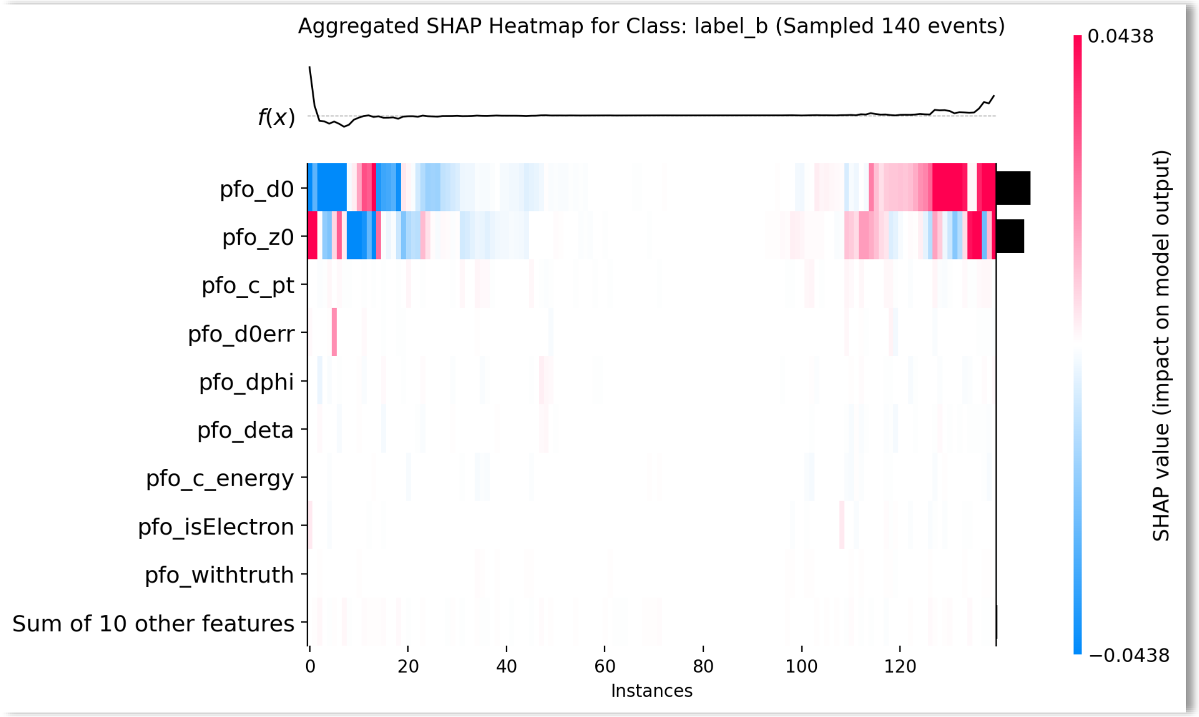
Summary Plot - Bar

Description

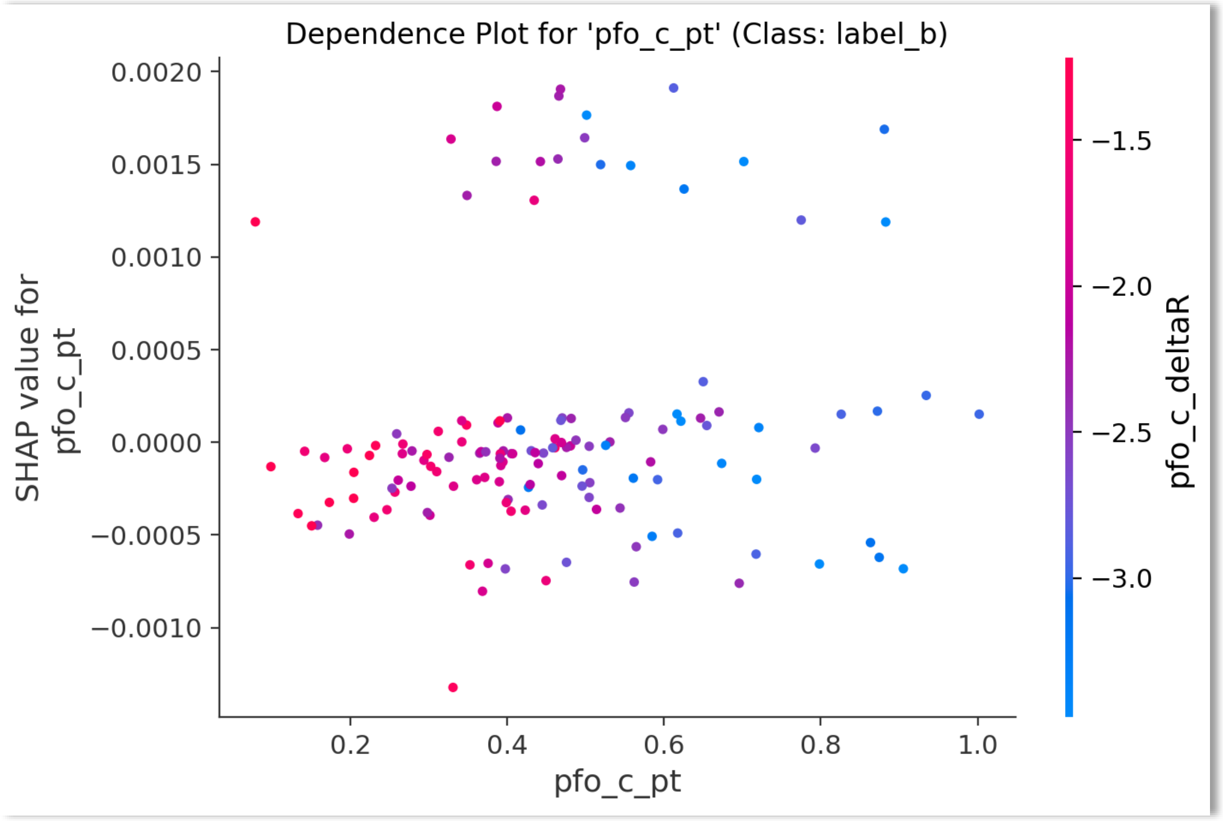


Force Plot

Description

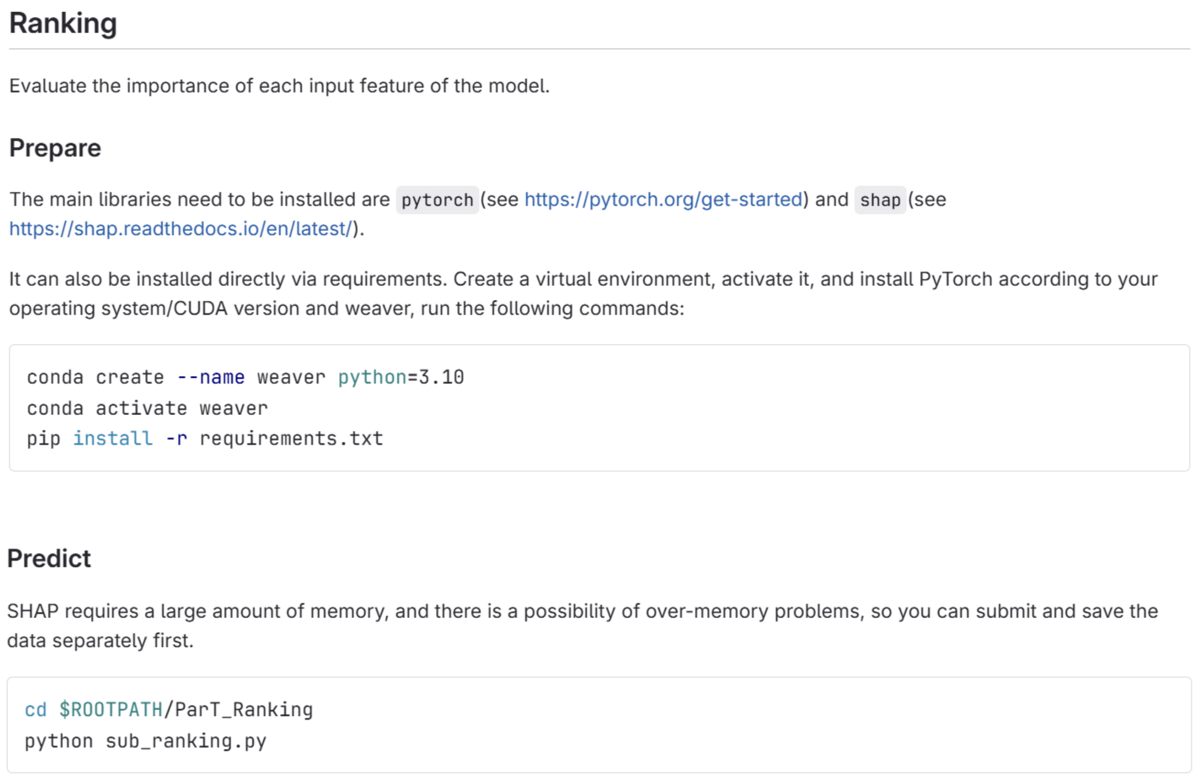


Heatmap



Dependence Plot

To use



Ranking

Evaluate the importance of each input feature of the model.

Prepare

The main libraries need to be installed are `pytorch` (see https://pytorch.org/get-started) and `shap` (see https://shap.readthedocs.io/en/latest/).

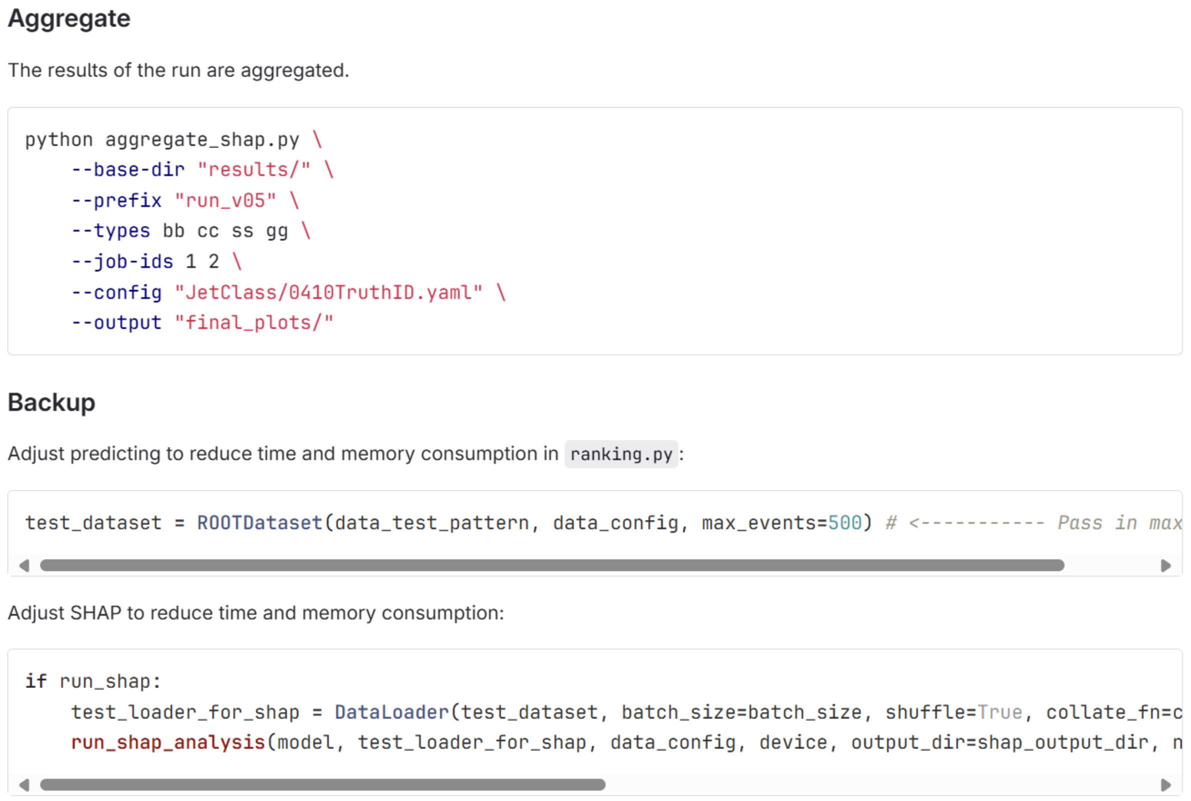
It can also be installed directly via requirements. Create a virtual environment, activate it, and install PyTorch according to your operating system/CUDA version and weaver, run the following commands:

```
conda create --name weaver python=3.10
conda activate weaver
pip install -r requirements.txt
```

Predict

SHAP requires a large amount of memory, and there is a possibility of over-memory problems, so you can submit and save the data separately first.

```
cd $ROOTPATH/ParT_Ranking
python sub_ranking.py
```



Aggregate

The results of the run are aggregated.

```
python aggregate_shap.py \
    --base-dir "results/" \
    --prefix "run_v05" \
    --types bb cc ss gg \
    --job-ids 1 2 \
    --config "JetClass/0410TruthID.yaml" \
    --output "final_plots/"
```

Backup

Adjust predicting to reduce time and memory consumption in `ranking.py`:

```
test_dataset = ROOTDataset(data_test_pattern, data_config, max_events=500) # <----------- Pass in max
```

Adjust SHAP to reduce time and memory consumption:

```
if run_shap:
    test_loader_for_shap = DataLoader(test_dataset, batch_size=batch_size, shuffle=True, collate_fn=c
    run_shap_analysis(model, test_loader_for_shap, data_config, device, output_dir=shap_output_dir, n
```

[pimohan@ihep.ac.cn / ParT_Ranking · GitLab]()