Higgs Boson CP State Analysis

DOCUMENTATION AND GUIDES

JULY 2024

Table of Contents

1. Configuring The Environment
2. ML Flow
3. Model Architecture
4. Command Line Options
5. Examples
6. Evaluation Plots
7. Other Plots

CONFIGURING THE ENVIRONMENT

This section shows how to create a virtual environment on **Linux** (though also tested on **MacOS**) and then install all the necessary dependencies. You are expected to have the **Python 3.11.3** interpreter (Python Software Foundation official page) and **Git** (Project Git official page) installed on your computer.

If you are going to configure your working environment on Ares clusters (PLGrid, Cyfronet infrastructure), you need to allocate resources and then load the appropriate Python module. The following command will allocate one CPU for 1 hour and then load the interpreter we need:

```
$ srun --time 00-01:00:00 --pty /bin/bash
$ module load python/3.11.3-gcccore-12.3.0
```

Let's now clone the repository, create a new virtual environment in it, activate the environment, update `pip` (package-management system) and install all the libraries stated in `requirements.txt` (available in the repository):

```
$ git clone https://github.com/klasocha/HiggsCP.git
$ cd HiggsCP
$ git checkout tyerniyazov-rhorho_refactored
$ python3.11 -m venv .venv
$ source .venv/bin/activate
(.venv) $ python -m pip install --no-cache-dir --upgrade pip
(.venv) $ pip install --no-cache-dir -r requirements.txt
```

The process of installing the libraries may take a while to finish. When it completes, your working environment is ready to run the code from the cloned repository. For Ares users, the first two steps in the violet box are regular. For all users, the step involving activating the virtual environment is regular. Don't forget to deactivate the environment if you finish experimenting with the project:

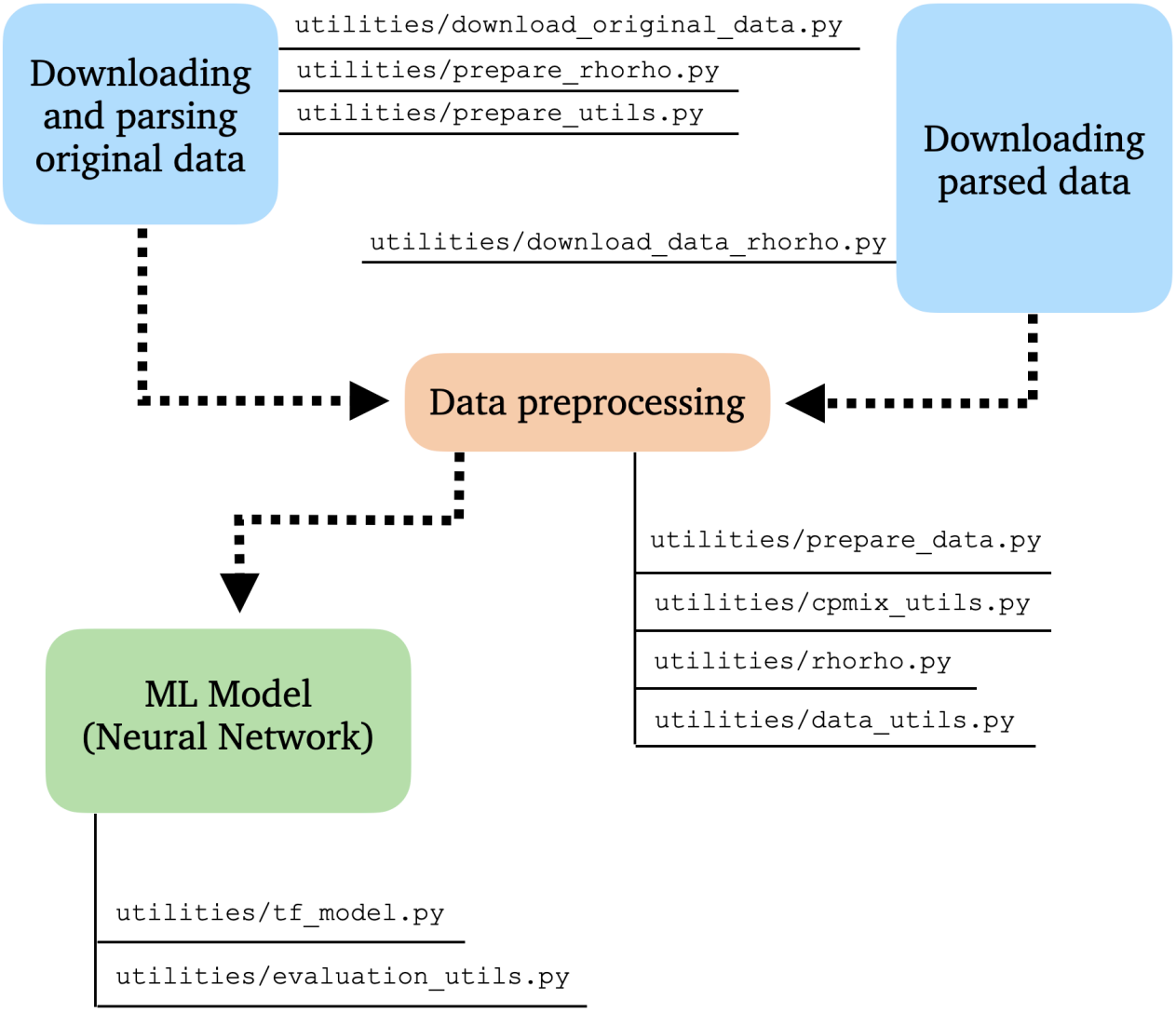
```
# Activating
$ source .venv/bin/activate

# Deactivating
(.venv) $ deactivate
```

ML FLOW

The whole project contains a number of packages responsible for implementing different parts of the machine learning flow. Each package contain several modules which are used by the main program `main.py` invoking particular actions, such as downloading data, training the model or even preparing some plots.

The below diagram depicts the main parts of the ML pipeline and the names of some of the modules involved:

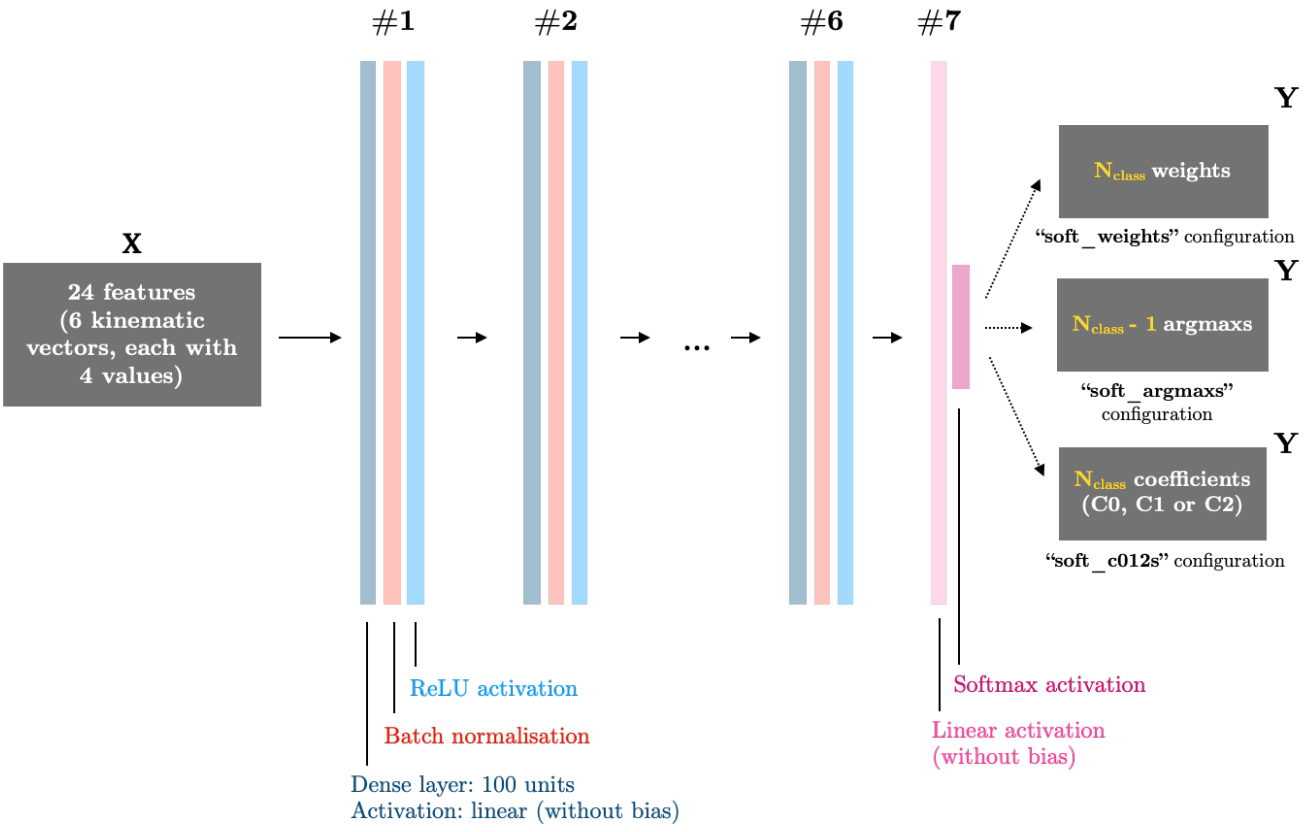


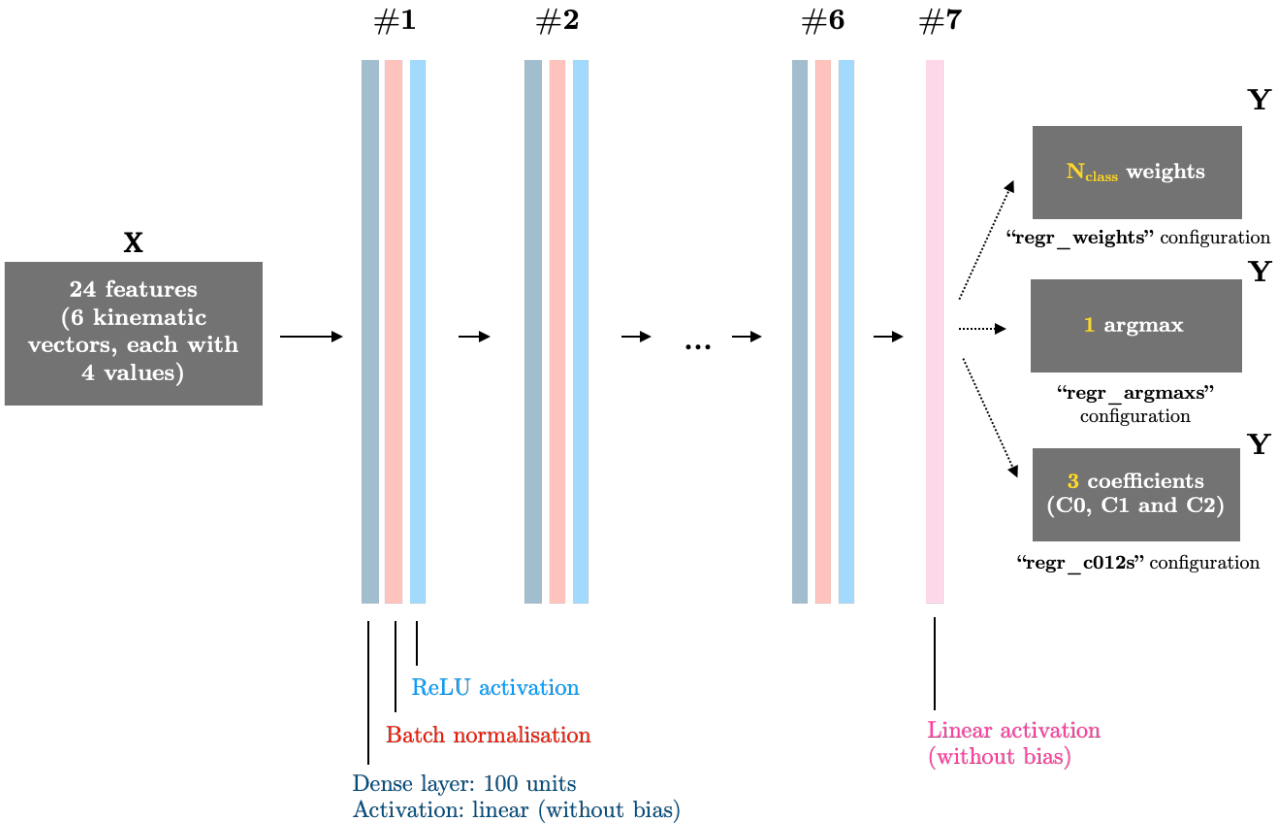
MODEL ARCHITECTURE

The ML model is represented by an artificial neural network which takes 6 different configurations. Depending on the type of configuration the architecture changes. Additionally, it is possible to specify some of the hyperparameters affecting the model structure and the training process (optimiser, loss function, learning rate etc.)

The two diagrams below show the whole architecture for those configurations:

- soft_weights: classification, Wt
- soft_argmaxs: classification, α^{CP}_{max}
- soft_c012s: classification, C_0, C_1, C_2 (only one type of coefficients is trained at once)
- regr_weights: regression, Wt
- regr_argmaxs: regression, α^{CP}_{max}
- regr_c012s : regression, C_0, C_1, C_2 (all three types of coefficients are trained simultaneously)




#1
#2
#6
#7
X
24 features
(6 kinematic
vectors, each with
4 values)
Y
N_{class} weights
"regr_weights" configuration
Y
1 argmax
"regr_argmaxs"
configuration
Y
3 coefficients
(C0, C1 and C2)
"regr_c012s" configuration
ReLU activation
Batch normalisation
Dense layer: 100 units
Activation: linear (without bias)
Linear activation
(without bias)

COMMAND LINE OPTIONS

Whilst there are a number of options available, we will consider the most important ones in this section. To see the full list of the command line arguments, you can use the `--help` option while calling `main.py`:

```
$ python main.py --help
```

`--input [data path]` – the path to the directory containing the data sets. If you download the data sets for the first time, this directory will be created automatically.

`--num_classes [number of classes]` – how many classes are going to be used for discretisation. This also impacts the model architecture as the output shape may change (see section „Model Architecture").

`--hits_c012s [hits_c0s / hits_c1s / hits_c2s]` – defines which coefficients (C0, C1 or C2) to choose as labels during the training process.

`--features [input features choice]` – choosing the set of features for training. You may choose one of the following values: `Variant-All, Variant-1.0, Variant-1.1, Variant-2.0, Variant-2.1, Variant-4.1.`

--**`training_method [configuration]`** – choosing the model type. It defines the architecture of the model (see section *„Model Architecture"*). You may choose one of the following types: `soft_weights, soft_c012s, soft_argmaxs, regr_c012s, regr_weights, regr_argmaxs.`

The following options allow you to control some of the hyperparameters applied during the training process:

`--dropout [rate]` – dropout probability (applied during the training process).
`--optimizer` – the type of the optimizer. You may choose one of the following (the default and recommended value for this option is AdamOptimizer):

- `GradientDescentOptimizer`
- `AdadeltaOptimizer`
- `AdagradOptimizer`
- `ProximalAdagradOptimizer`
- `AdamOptimizer`
- `FtrlOptimizer`
- `ProximalGradientDescentOptimizer`
- `RMSPropOptimizer`

`--epochs [epochs]` – the number of epochs used during the training process.

The next option is important for specifying the path to the model weights, training history, configuration file, as well as the directory containing predictions:

`--model_location [directory name]` - specifying the name of the directory in `results/` containing the model state (weights, metadata).

The following group of arguments impacts the way in which different plots are prepared (more details are available in the source code of the modules responsible for drawing the plots. You can find them all located in `plots/`):

`--output [path]` – output path for plots
`--format [format]` – the format of the output plots (`png, pdf, eps`)
`--option` – specifying the plotting module to call. You may choose one of the following:

- `PHISTAR-DISTRIBUTION`
- `C012S-WEIGHT`
- `C012S-DISTRIBUTION`
- `WEIGHTS-FOR-EVENT-VIA-C012`
- `UNWEIGHTED-EVENTS-WEIGHTS`
- `RESULTS_ANALYSIS_1`
- `RESULTS_ANALYSIS_2`
- `RESULTS_ANALYSIS_3`
- `RESULTS_ANALYSIS_4`
- `RESULTS_ANALYSIS_5`
- `WEIGHTS-FOR-PREDICTED`
- `C012S-FOR-PREDICTED`

`--hypothesis [hypothesis class indices]` – the alphaCP classes (e.g. `0-4-46` for #0, #4 and #46, respectively).

You can find out more information about the plots in sections „*Evaluation Plots*” and „*Other Plots*”.

The last option we should take a look at before running the main program is about choosing the action controlling the ML flow behaviour (see section *„Examples”*):

`--action [action]` - choosing the action. Available values:

- `download_and_prepare_original`
- `download_prepared_and_preprocess`
- `preprocess`
- `train`
- `continue_training`

- predict_train_and_valid
- plot
- test_parsed_data
- test_model_on_unwt_events
- predict_test
- test_labels
- experimental
- test_model_on_all_events

EXAMPLES

Take a look at some of the examples to see how `--action` controls what modules are called by `main.py`:

```
# Downloading and preparing the original (raw) data sets
python main.py --action "download_and_prepare_original" --input "data"

# Preprocessing:
python main.py --action "preprocess" --input "data" --features
Variant-All --num_classes "51"

# Training the model on 25 epochs for the „soft_weights" configuration:
python main.py --action "train" --input "data" --num_classes "51"
--epochs 25 --training_method "soft_weights" --model_location
"51_classes_variant_all" --features Variant-All

# Using the pre-trained model weights to make predictions on the validation
and training sets:
python main.py --action "predict_train_and_valid" --input "data"
--num_classes "51" --model_location "51_classes_variant_all" --features
 Variant-All --use_filtered_data
```

EVALUATION PLOTS

In this section you can find a list of commands needed to obtain all the plots related to the model evaluation. We consider the `Variant-All` feature set (`51` classes) as an example:

```
# soft_weights
python main.py --action "plot" --input
"results/soft_weights/51_classes_variant_all/predictions" --output "plots/
figures" --option "RESULTS_ANALYSIS_1" --num_classes "51" --training_method
"soft_weights" --features Variant-All --dataset "test" --use_filtered_data

# soft_c012s
python main.py  --action "plot" --input
"results/soft_c012s/variant_all/51_classes_c" --output "plots/figures" --
option "RESULTS_ANALYSIS_2" --num_classes "51" --training_method "soft_c012s"
--features Variant-All --dataset "test" --use_filtered_data

# soft_argmaxs
python main.py --action "plot" --input
"results/soft_argmaxs/51_classes_variant_all/predictions" --output
"plots/figures" --format "png" --option "RESULTS_ANALYSIS_3" --features
Variant-All --training_method "soft_argmaxs" --dataset "test" --num_classes
"51" --use_filtered_data

# regr_weights
python main.py --action "plot" --input
„results/regr_weights/51_classes_variant_all/predictions" --output "plots/
figures" --format "png" --option "RESULTS_ANALYSIS_1" --num_classes "51"
--training_method "regr_weights" --features Variant-All --dataset "test"
--use_filtered_data

# regr_c012s
python main.py --action "plot" --input
"results/regr_c012s/51_classes_variant_all/predictions" --output "plots/
figures" --option "RESULTS_ANALYSIS_4" --num_classes "51" --training_method
"regr_c012s" --features Variant-All --dataset "test" --use_filtered_data

# regr_argmaxs
python main.py --action "plot" --input
"results/regr_argmaxs/51_classes_variant_all/predictions" --output
"plots/figures" --format "png" --option "RESULTS_ANALYSIS_5" --num_classes
"51" --training_method "regr_argmaxs" --features Variant-All --dataset "test"
--use_filtered_data
```

OTHER PLOTS

In this section you can find the commands responsible for other plots (the unweighted events experiment, ϕ^* distribution, $C_0/C_1/C_2$ distribution etc.). You may change most of the command line arguments:

```
# Unweighted events test

python main.py --action "test_model_on_unwt_events" --input "data"
--output "plots/figures/test_model_on_unwt_events/51_classes_variant_all/
soft_weights" --num_classes "51" --hypothesis "0-4-46" --training_method
"soft_weights" --model_location "51_classes_variant_all" --features
"Variant-All"

python main.py --action "test_model_on_unwt_events" --input "data" --output
"plots/figures/test_model_on_unwt_events/51_classes_variant_all/
regr_weights" --num_classes "51" --hypothesis "0-4-46" --training_method
"regr_weights" --model_location "51_classes_variant_all" --features
"Variant-All"

python main.py --action "test_model_on_unwt_events" --input "data" --output
"plots/figures/test_model_on_unwt_events/51_classes_variant_all/soft_c012s"
--num_classes "51" --hypothesis "0-4-46" --training_method "soft_c012s" --
model_location "variant_all/51_classes" --features "Variant-All"

python main.py --action "test_model_on_unwt_events" --input "data" --output
"plots/figures/test_model_on_unwt_events/51_classes_variant_all/regr_c012s"
--num_classes "51" --hypothesis "0-4-46" --training_method "regr_c012s" --
model_location "51_classes_variant_all" --features "Variant-All"

python main.py --action "test_model_on_unwt_events" --input "data" --output
"plots/figures/test_model_on_unwt_events/51_classes_variant_all/
soft_argmaxs" --num_classes "51" --hypothesis "0-4-46" --training_method
"soft_argmaxs" --model_location "51_classes_variant_all" --features
"Variant-All"

python main.py --action "test_model_on_unwt_events" --input "data" --output
"plots/figures/test_model_on_unwt_events/51_classes_variant_all/
regr_argmaxs" --num_classes "51" --hypothesis "0-4-46" --training_method
"regr_argmaxs" --model_location "51_classes_variant_all" --features
"Variant-All"


# All events test

python main.py --action "test_model_on_all_events" --input "data"
--output "plots/figures/test_model_on_all_events/51_classes_variant_all/
soft_weights" --num_classes "51" --training_method "soft_weights" --
model_location "51_classes_variant_all" --features "Variant-All"
```

```
python main.py --action "test_model_on_all_events" --input "data" --output
"plots/figures/test_model_on_all_events/51_classes_variant_all/
regr_weights" --num_classes "51" --training_method "regr_weights" --
model_location "51_classes_variant_all" --features "Variant-All"

python main.py --action "test_model_on_all_events" --input "data" --output
"plots/figures/test_model_on_all_events/51_classes_variant_all/soft_c012s"
--num_classes "51" --training_method "soft_c012s" --model_location
"variant_all/51_classes" --features "Variant-All"

python main.py --action "test_model_on_all_events" --input "data" --output
"plots/figures/test_model_on_all_events/51_classes_variant_all/regr_c012s"
--num_classes "51" --training_method "regr_c012s" --model_location
"51_classes_variant_all" --features "Variant-All"

python main.py --action "test_model_on_all_events" --input "data" --output
"plots/figures/test_model_on_all_events/51_classes_variant_all/
soft_argmaxs" --num_classes "51" --training_method "soft_argmaxs" --
model_location "51_classes_variant_all" --features "Variant-All"

python main.py --action "test_model_on_all_events" --input "data" --output
"plots/figures/test_model_on_all_events/51_classes_variant_all/
regr_argmaxs" --num_classes "51" --training_method "regr_argmaxs" --
model_location "51_classes_variant_all" --features "Variant-All"


# Calculated C0/C1/C2 distribution

python main.py --action "plot" --option "C012S-DISTRIBUTION" --input "data"
--output "plots/figures" --format "png" --show


# Predicted C0/C1/C2 distribution

python main.py --action "plot" --option "C012S-FOR-PREDICTED" --input
"results/soft_c012s/variant_all/51_classes_c" --output "plots/figures" --
use_filtered_data --features "Variant-All" --training_method "soft_c012s"
--dataset "test" --num_classes 51


# Wt calculated via C0/C1/C2

python main.py --action "plot" --option "WEIGHTS-FOR-EVENT-VIA-C012" --
input "data" --output "plots/figures" --show


# Predicted Wt

python main.py --action "plot" --option "WEIGHTS-FOR-PREDICTED" --input
"results/soft_weights/51_classes_variant_all/predictions/" --output "plots/
figures" --use_filtered_data --features "Variant-All" --training_method
"soft_weights" --dataset "test" --num_classes 51
```

```
# Calculated Wt test

python main.py --action "plot" --option "C012S-WEIGHT" --input "data" --
output "plots/figures" --format "png" --show


# Phistar distribution

python main.py --action "plot" --option "PHISTAR-DISTRIBUTION" --input
"data" --output "plots/figures" --format "png" --show --num_classes "51" --
feature "Variant-1.1"

python main.py --action "plot" --option PHISTAR-DISTRIBUTION --input "data"
--output "plots/figures" --format "png" --show --num_classes 51 --feature
"Variant-1.1" --use_unweighted_events

python main.py --action "plot" --option PHISTAR-DISTRIBUTION --input "data"
--output "plots/figures" --format "png" --show --num_classes 51 --feature
"Variant-1.1" --hypothesis „0-4-46"


# Unweighted events

python main.py --action "plot" --input "data" --output "plots/figures" --
format "png" --option "UNWEIGHTED-EVENTS-WEIGHTS" --num_classes "51"
```

Please, notice that many command line options shown in the present document require you to have the appropriate data, models, weights or model predictions prepared in advance.