AAanalysis Decision Map

v1.1 dev-preview · stable on PyPI: 1.0.3

decision / question · option · AAanalysis box: class / method / function · **data (df_*, X)** · explanation · planned (post v1.1)

START
What is your goal?

explore

① Explore
Sequences & groups

What do you want to explore?

mutations
AAMut
explore substitutions

sequences / datasets
One or two groups to analyze?
one
AAlogoPlot.single_logo
one group · conservation
two
AAlogoPlot.multi_logo
CPP.run
compare groups

representative proteins
How to find representatives?
sequence
filter_seq
cluster by similarity
numerical
AAclust.select_proteins
numerical representatives

predict

② Build
Engineer features · predict & explain

What is your prediction task?

classification
binary classification (1/0)
labels

regression
regression → threshold
SequenceFeature.get_labels_quantile
SequenceFeature.get_labels_tiered
labels

multi-class
multi-class
SequenceFeature.get_labels_ovr
SequenceFeature.get_labels_ovo
labels

What is your reference group? (label 0)

have negatives

positives + unlabeled
dPULearn
mine reliable negatives → 0

residue · need decoys
AAWindowSampler
build reference windows

df_seq + labels

CPP
feature engineering → df_feat

predict
SequenceFeature.feature_matrix
→ **X**
fit model
TreeModel
scikit-learn · PyTorch

explain
What kind of explanation?

feature attribution
Which level of explanation?
global
TreeModel
global feature importance
local
ShapModel
local feature impact
CPPPlot
feature_map · profile · ranking

others (later)
rule extraction · concept-based · example-based · causal modeling · uncertainty
(planned · post v1.1)

optimize

③ Optimize
Protein engineering

have a fitted predictor?

yes
SeqOpt
directed evolution

no
model-guided or model-free?

model-guided
build a model first (→ ②)
then

model-free
SeqMut
ΔCPP mutation scan
optional model → Δprediction

CPP Feature engineering (*Parts* × *Splits* × *Scales*): How AAanalysis identifies interpretable physicochemical features

df_seq + labels

What are your input data / settings?

AA scales
load_scales
Which ***Scale***? (physicochemical representation)
all
load_scales()
All AAontology scales
curated
load_scales(top_explain_n=10)
Interpretable scale sets
custom redundancy reduced
AAclust
reduce redundancy
Scale

sequences
SequenceFeature.get_df_parts
Which ***Part***? (biological sequence region)
residue
TMD window
(fixed TMD length)
domain
TMD + JMD-N / JMD-C
(flexible TMD length)
protein
the **whole chain**
Part

splitting
SequenceFeature.get_split_kws
Which ***Split***? (segments & periodic patterns)
composition-like
Segment (n_split_max=1)
position-resolved
Segment (n_split_max>1) · Pattern · PeriodicPattern
Split

numerical residue values
EmbeddingPreprocessor
StructurePreprocessor
AnnotationPreprocessor
→ **dict_num**
NumericalFeature.get_parts
→ **dict_num_parts**
Part

df_scales
df_parts
split_kws

CPP.run
Part* × *Split* × *Scale
→ **df_feat**

CPP.run_num
Part* × *Split
→ **df_feat**

CPP needs two inputs: features (Parts × Splits × Scales, built in the CPP block below) and **labels** (test group = 1 vs reference group = 0). **Sampling** builds the label-0 reference group: use your own negatives, `AAWindowSampler` decoys (residue level), or `dPULearn` — a deterministic PU-learning (ML) step that mines reliable negatives from positives + an unlabeled pool by PCA / distance on a feature matrix X (embeddings or features). It produces *labels* and is independent of CPP. **Splits apply to both** `df_parts` and `dict_num` parts — only **Scales** are sequence-only: numerical inputs already carry their values, so they **skip Scales** and go through `CPP.run_num`. **Two axes of a prediction task:** *kind of task* (classification / regression / multi-class) × *kind of data / unit* (residue-window / domain-TMD / whole protein). **Explain:** `CPPPlot` visualizes the CPP group signature straight from `df_feat` (no model); `TreeModel` (global importance) and `ShapModel` (local SHAP) need a fitted model on **X**. **Colors:** blue = class / method / function (cheat-sheet style); **dashed** = dev-preview / planned (post v1.1). **Plot pairs:** every analysis class has a mirror `*Plot`.