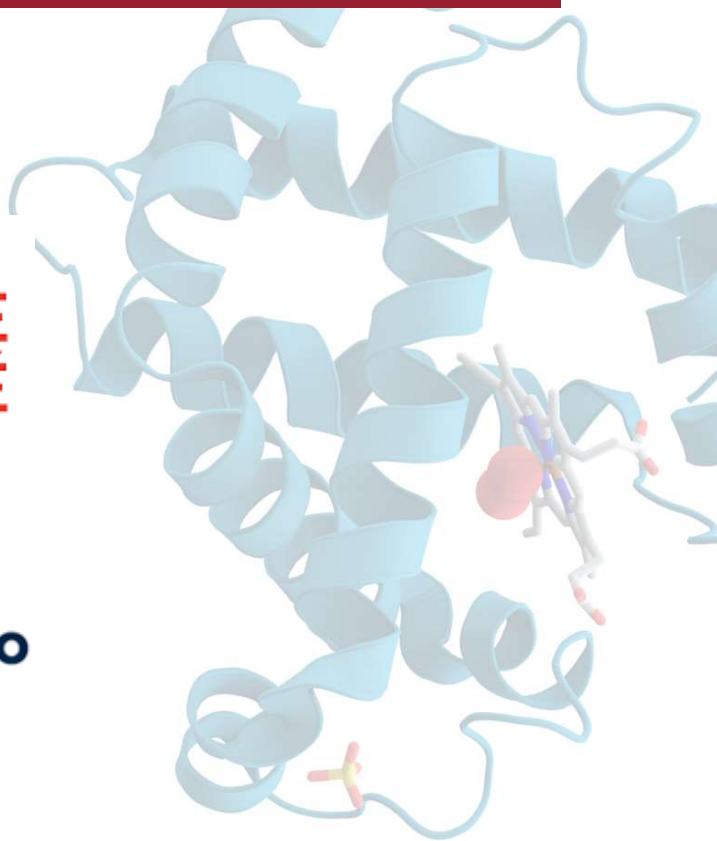
A machine learning playground

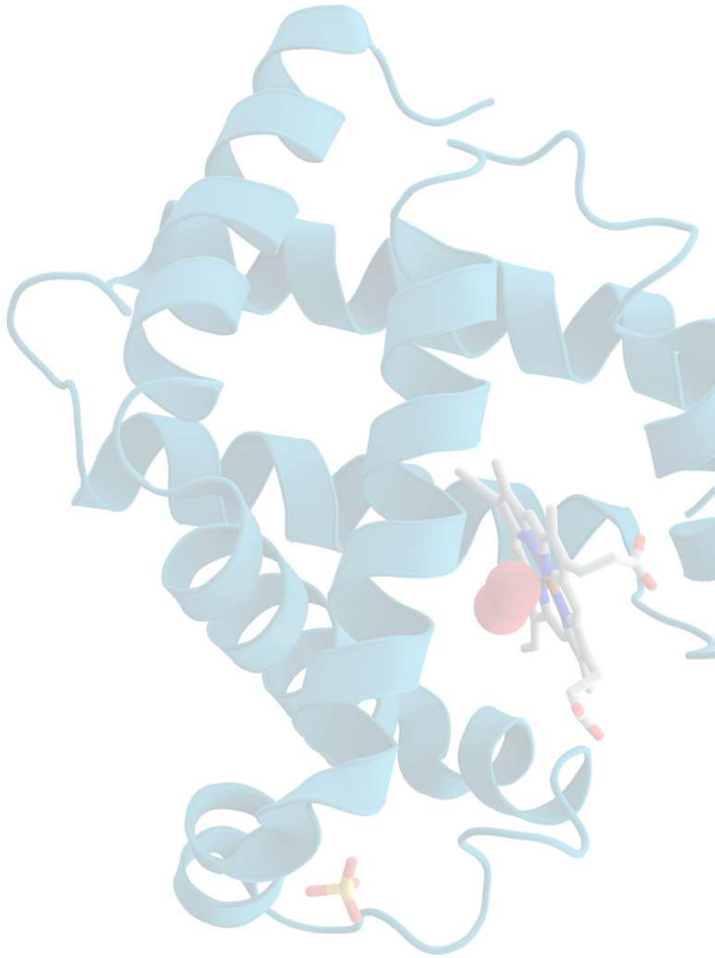
Unsupervised and supervised analysis of protein sequences

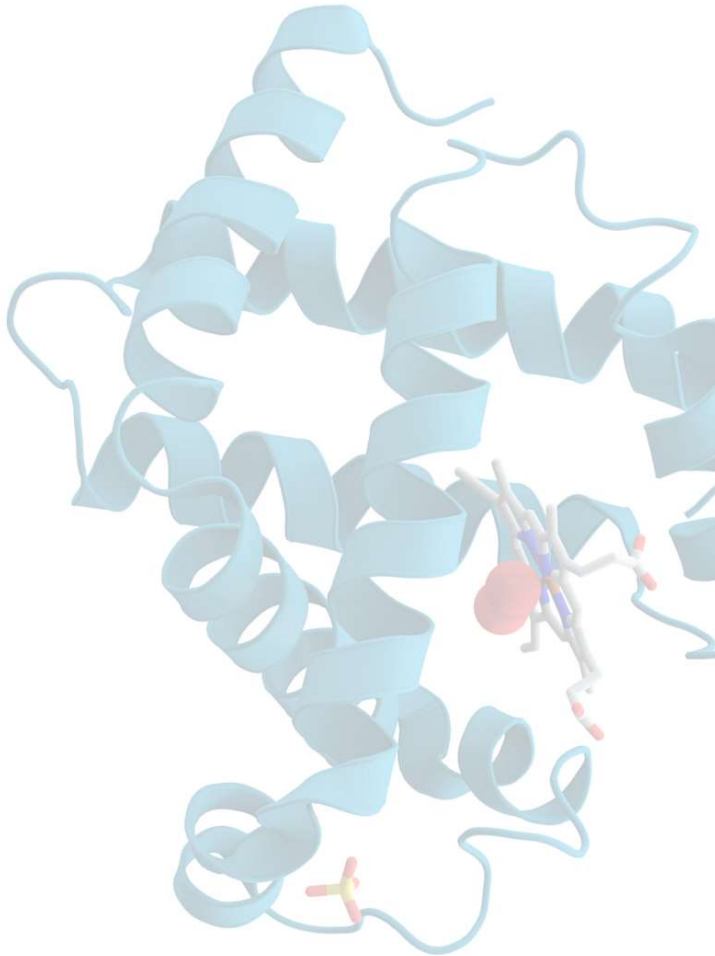
Matteo Allione

SORBONNE UNIVERSITÉ

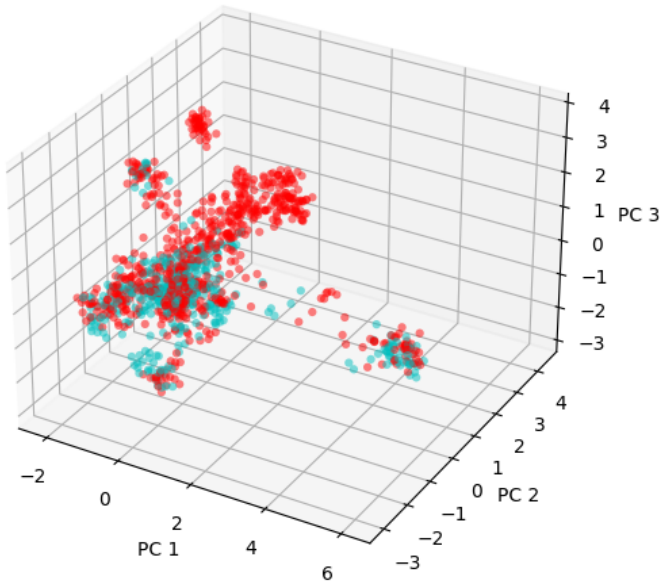
Politecnico di Torino

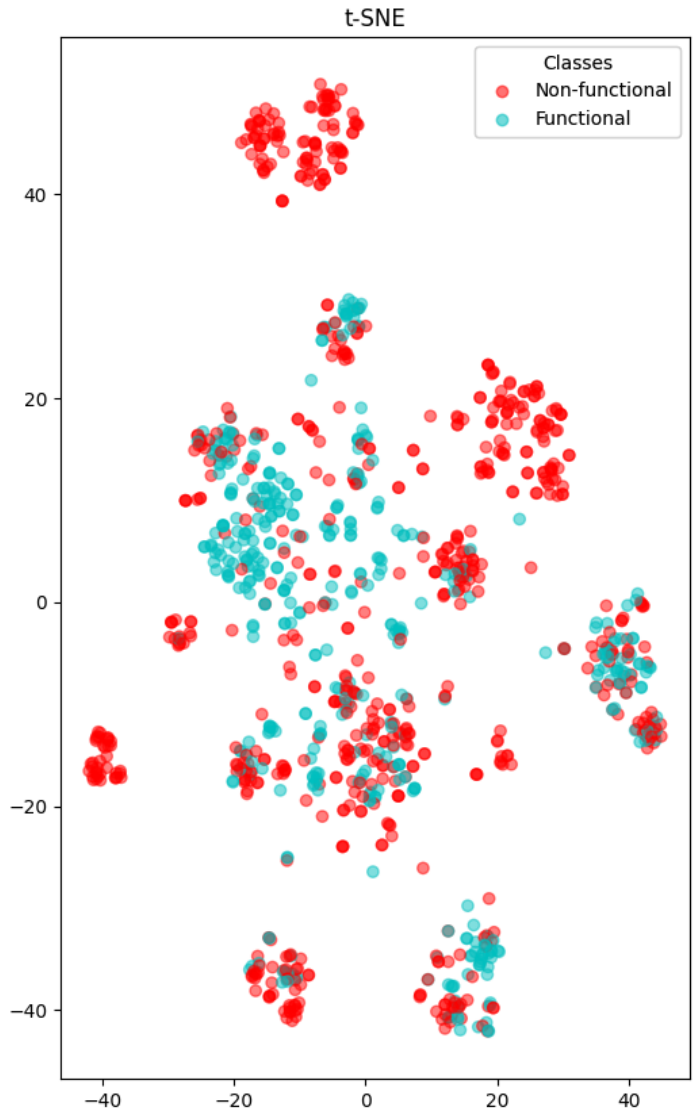






PCA and t-SNE





PCA with $\sim 10\%$ of the variance (left); t-SNE with perplexity 40 (right)

Edit distance and ISOMAP

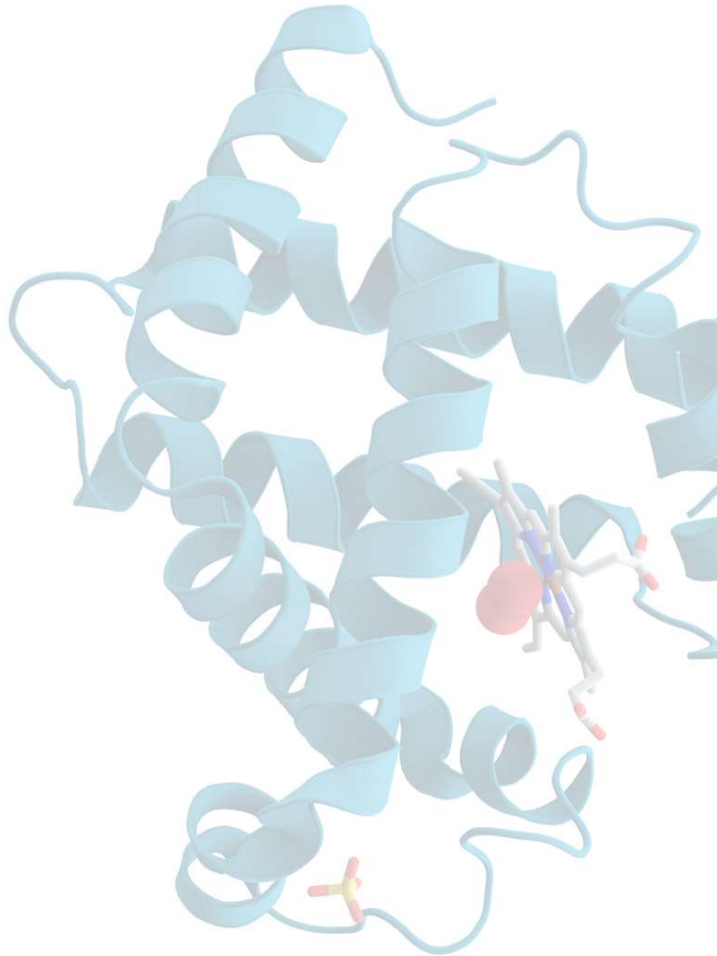
One-hot-encoding paradigm

···	I	L	A	D	L	-	···
···	0	0	1	0	0	0	···
···	0	0	0	0	0	0	···
···	0	0	0	0	0	0	···
···	0	0	0	1	0	0	···
···	···	···	···	···	···	···	···

$$E : \vec{p} \to \vec{s} \in \{0,1\}^{20 \cdot L}$$

$$\downarrow$$

- preserves equality between letters
- allows efficient computation of Hamming-like distances



Edit distance and ISOMAP

One-hot-encoding paradigm

···	I	L	A	D	L	-	···
···	0	0	1	0	0	0	···
···	0	0	0	0	0	0	···
···	0	0	0	0	0	0	···
···	0	0	0	1	0	0	···
···	···	···	···	···	···	···	···

$$E : \vec{p} \to \vec{s} \in \{0,1\}^{20 \cdot L}$$

$$\downarrow$$

- preserves equality between letters
- allows efficient computation of Hamming-like distances

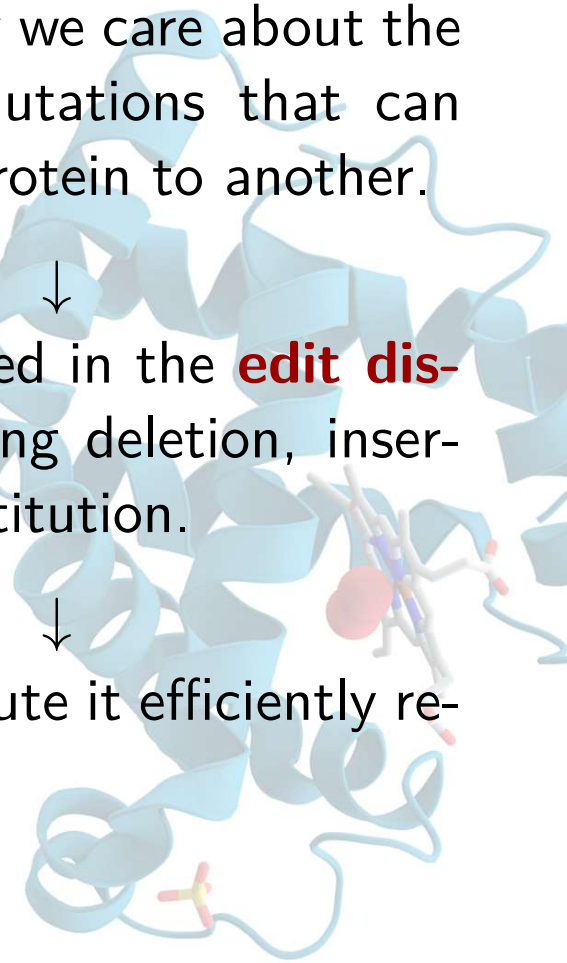
But in biology we care about the number of mutations that can lead from a protein to another.

$$\downarrow$$

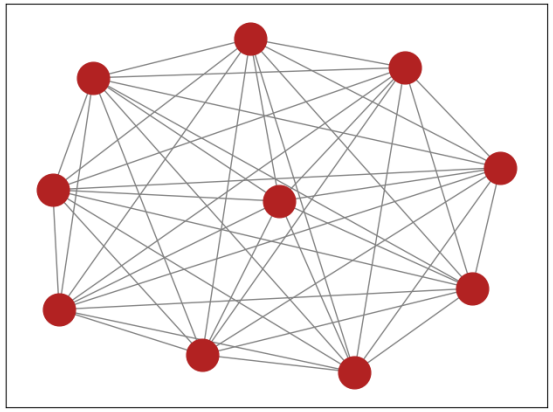
This is encoded in the **edit distance**: allowing deletion, insertion and substitution.

$$\downarrow$$

We can compute it efficiently recursively.

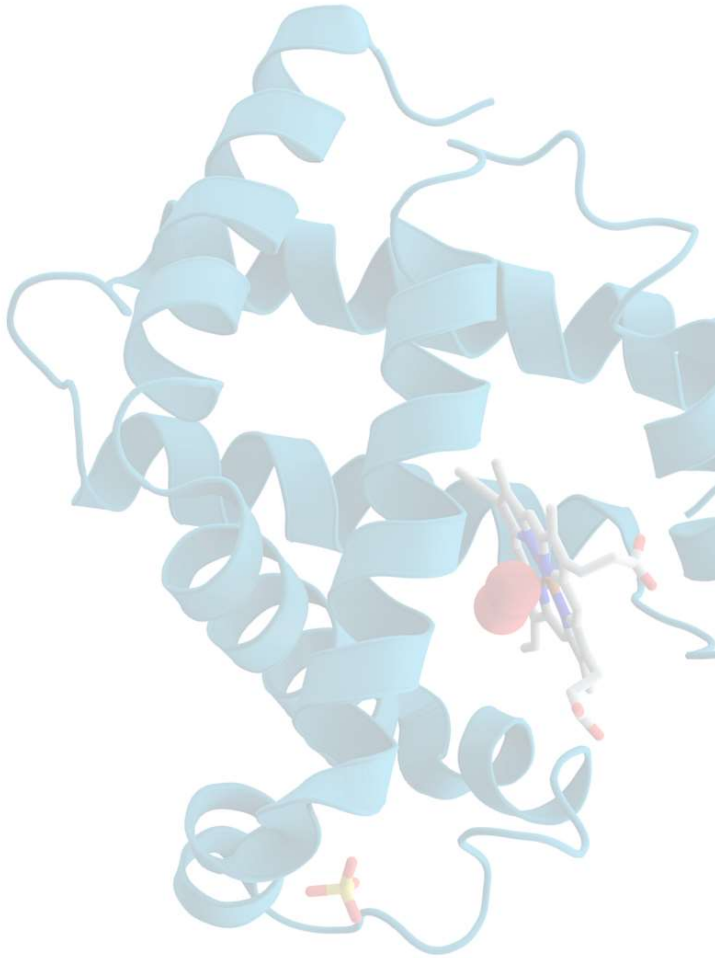


Edit distance and ISOMAP

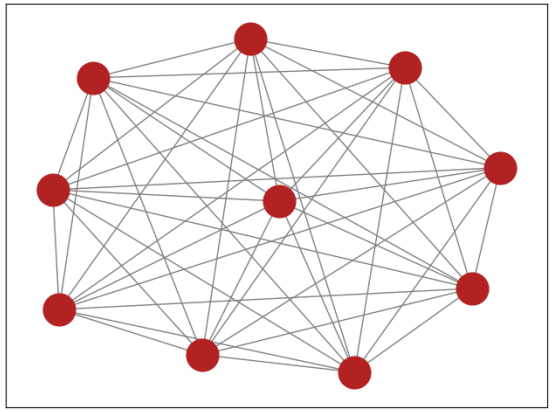


$$\Delta^{ij} = ||z_i - z_j||^2$$

$$\hat{G} = -\frac{1}{2}(\mathbb{I} - \frac{1}{N}\mathbf{U})\Delta(\mathbb{I} - \frac{1}{N}\mathbf{U})$$

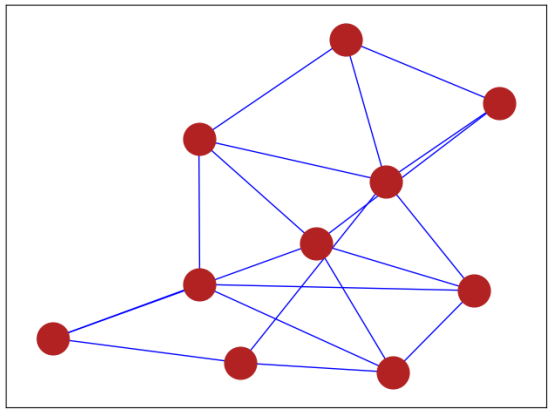


Edit distance and ISOMAP



$$\Delta^{ij} = ||z_i - z_j||^2$$

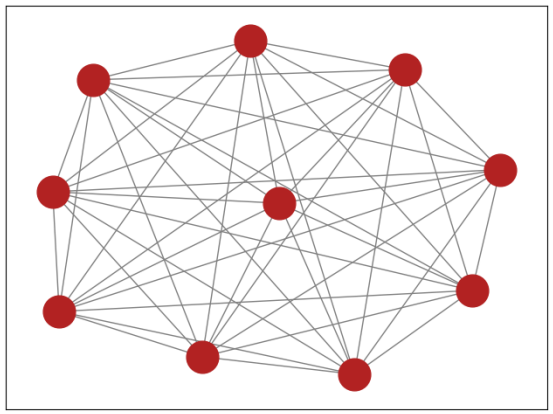
$$\hat{G} = -\frac{1}{2}(\mathbb{I} - \frac{1}{N}\mathbf{U})\Delta(\mathbb{I} - \frac{1}{N}\mathbf{U})$$

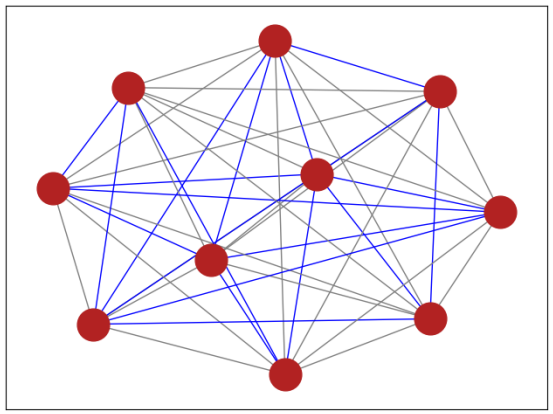


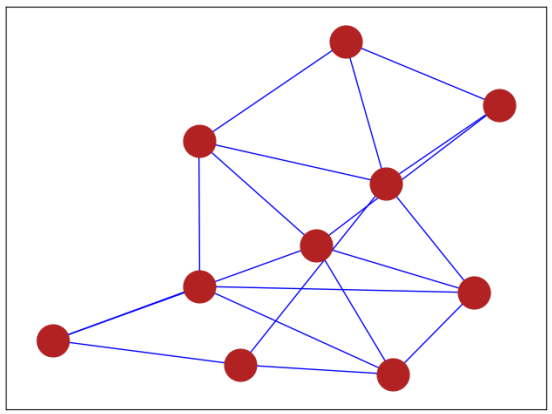
Edit distance and ISOMAP

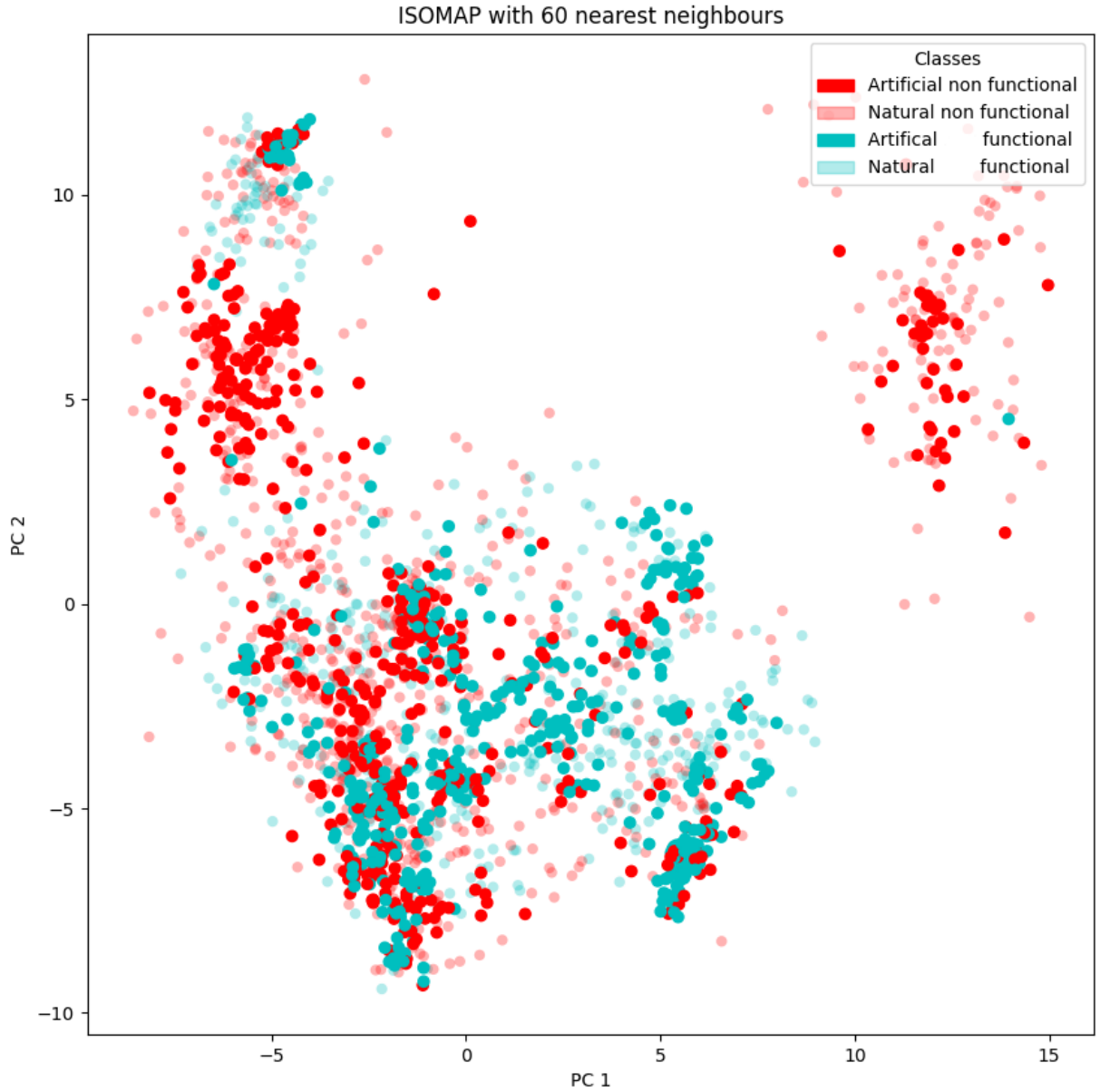
$$\Delta^{ij} = ||z_i - z_j||^2$$

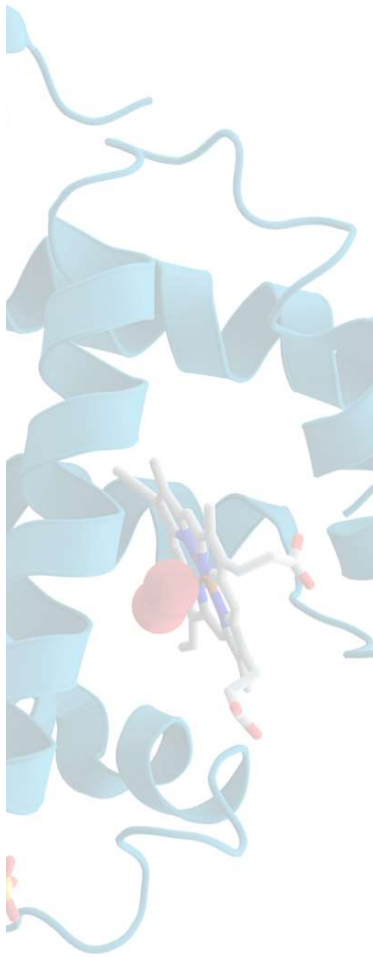
$$\hat{G} = -\frac{1}{2}(\mathbb{I} - \frac{1}{N}\mathbf{U})\Delta(\mathbb{I} - \frac{1}{N}\mathbf{U})$$

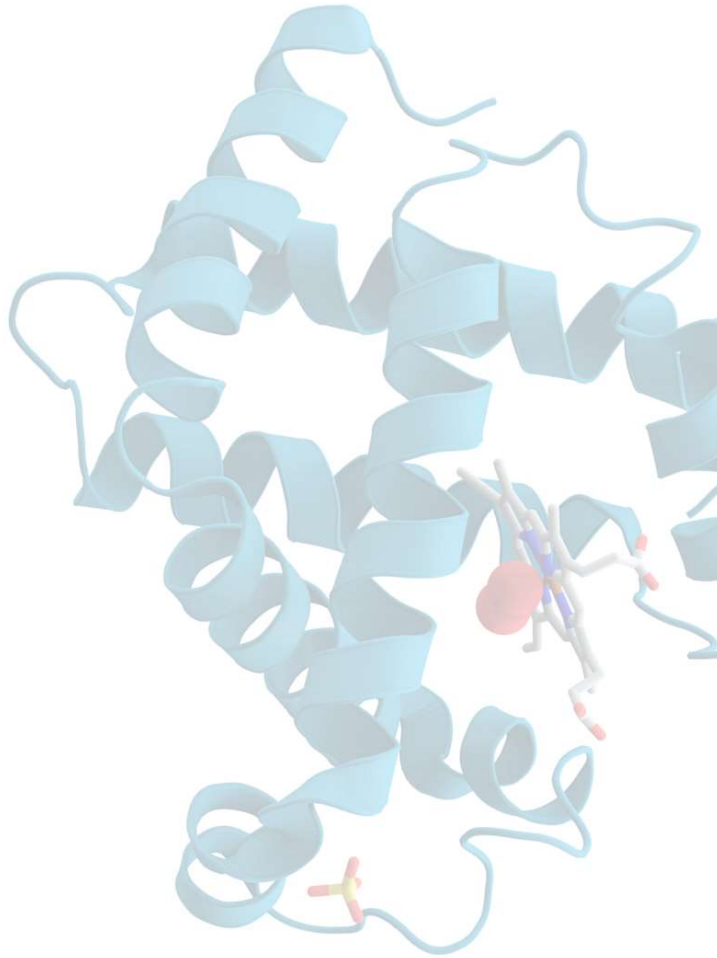




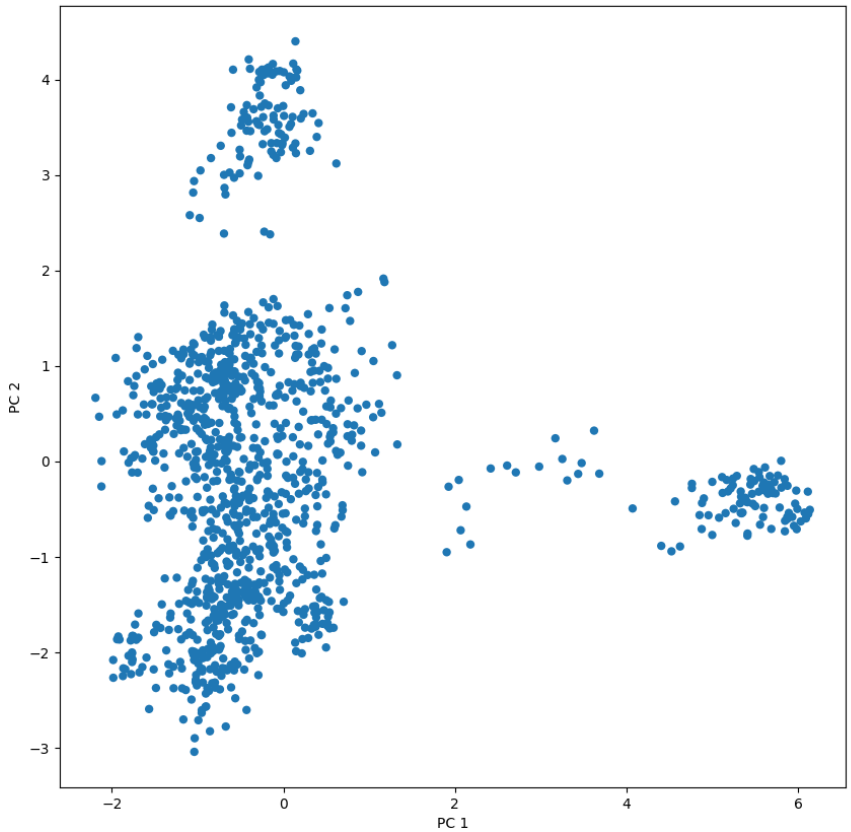


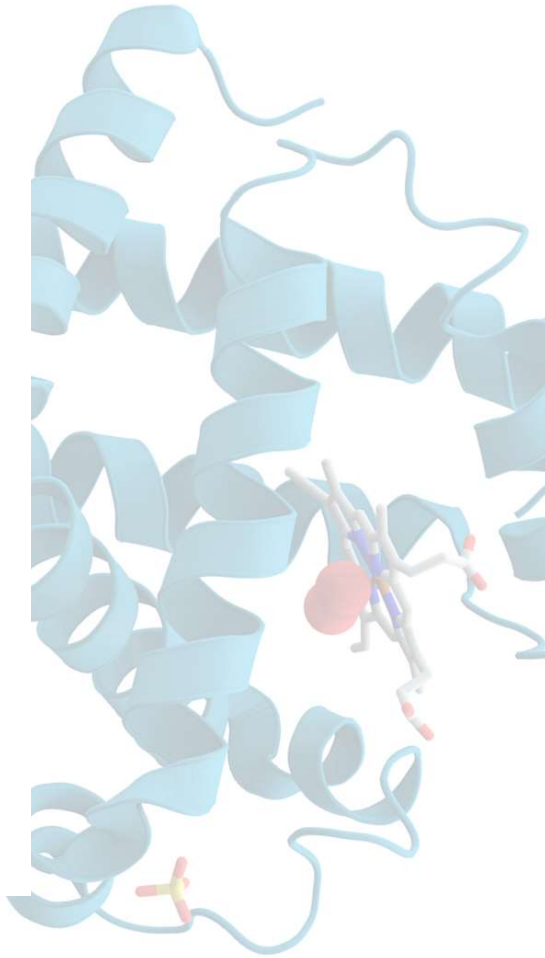

ISOMAP with 60 nearest neighbours
Classes
Artificial non functional
Natural non functional
Artifical functional
Natural functional
PC 1
PC 2
10
5
0
−5
−10
−5
0
5
10
15



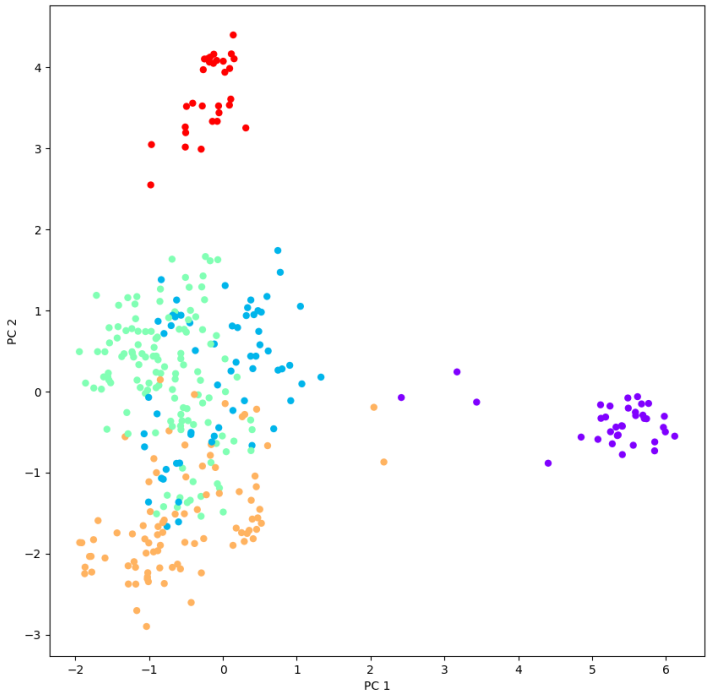


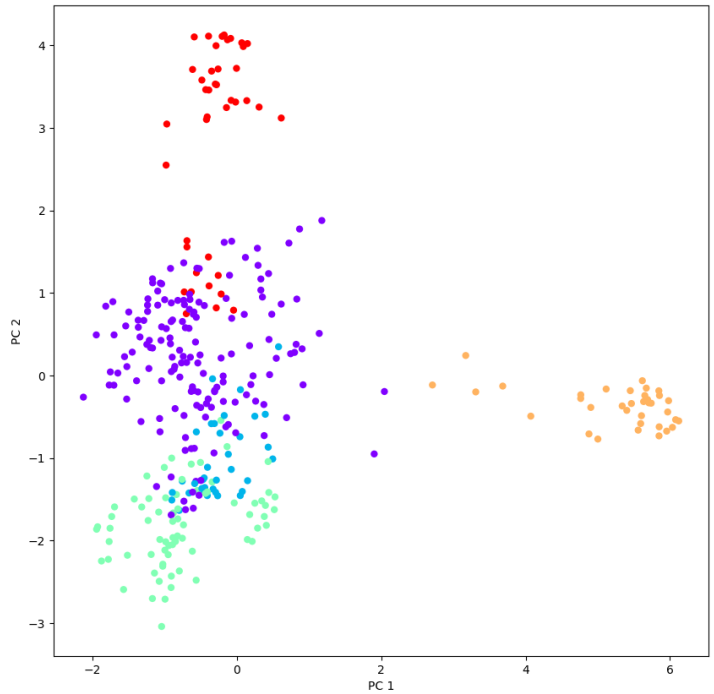
k-means and its consistency



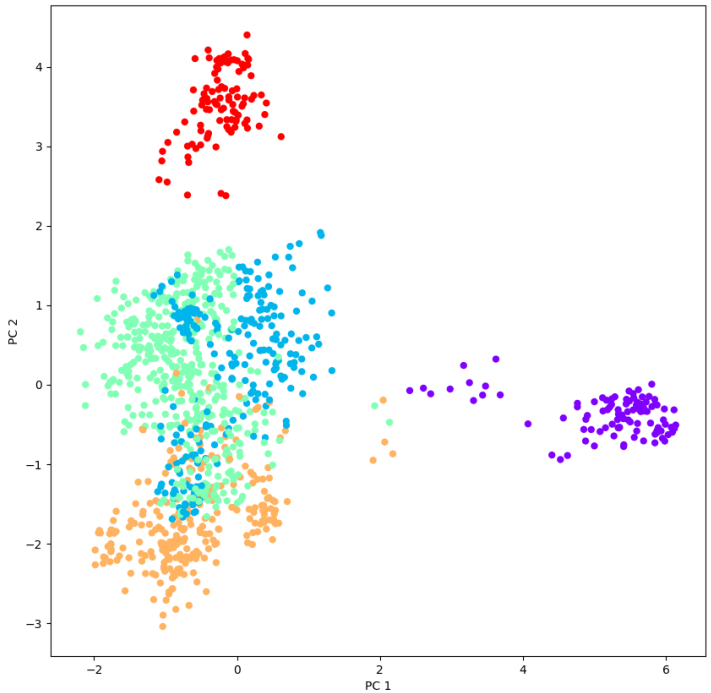


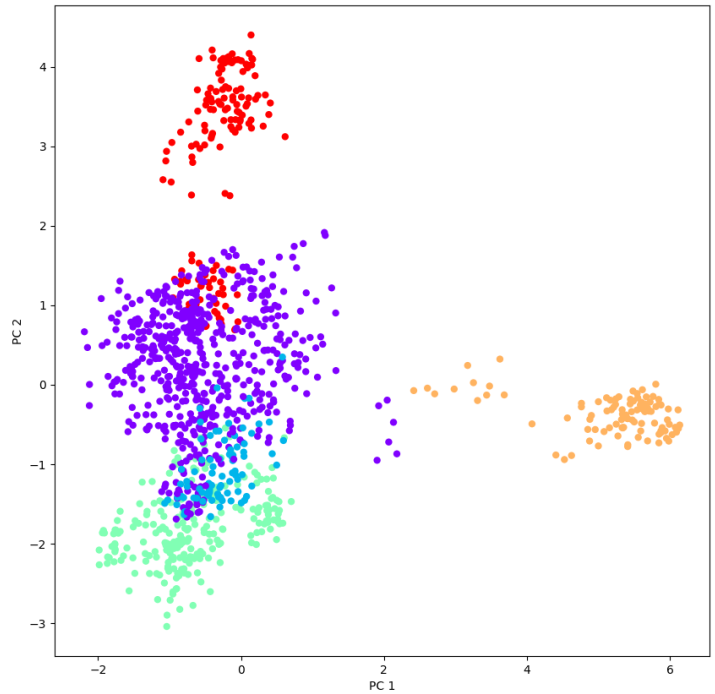
k-means and its consistency



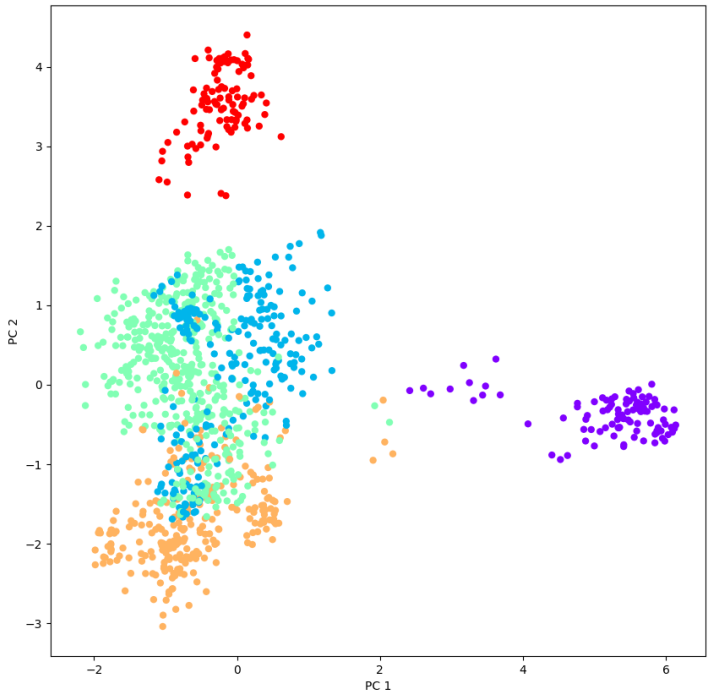


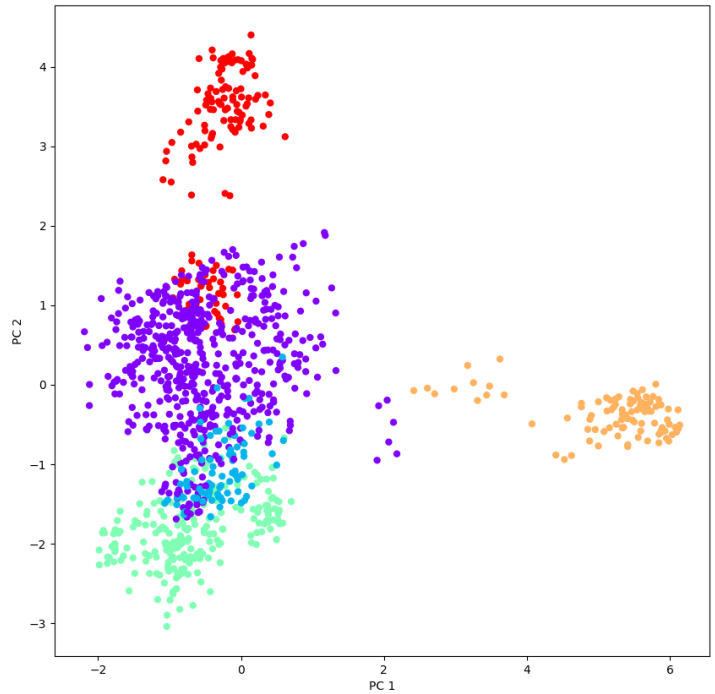
k-means and its consistency



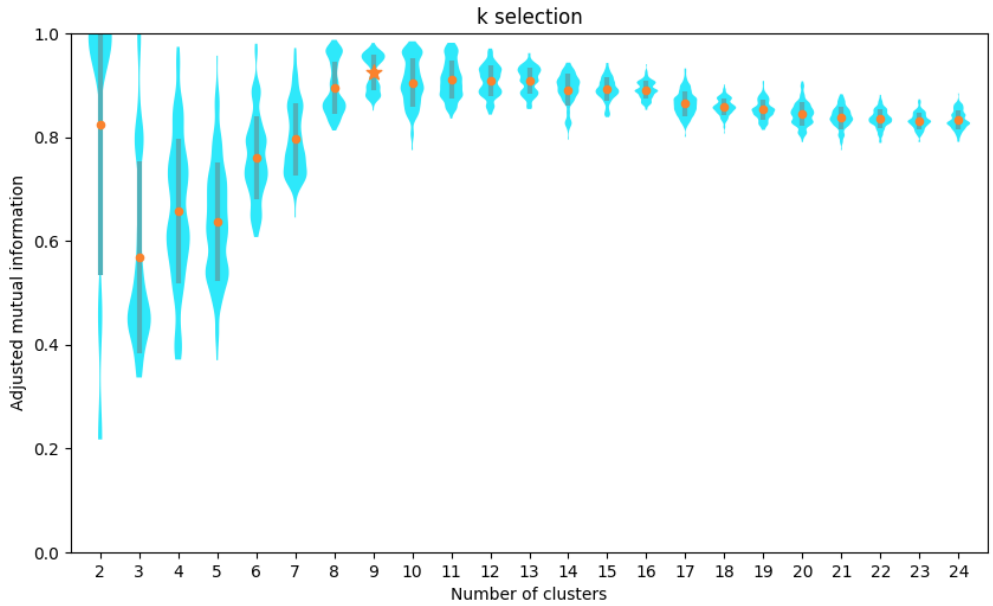


k-means and its consistency



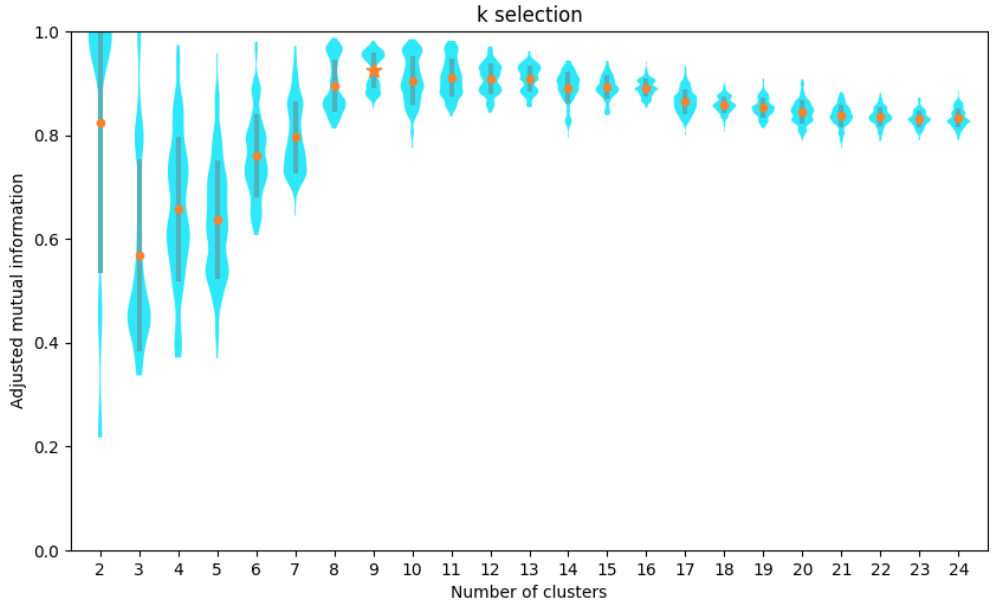


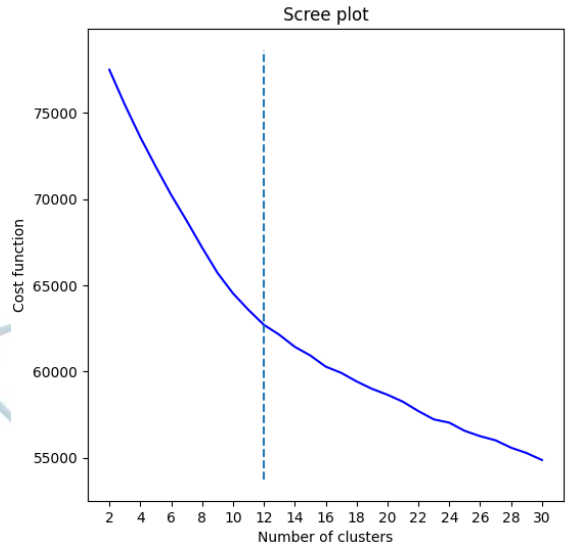
$$I(C_1, C_2) = \sum_{C_1, C_2} p(C_1, C_2) \log \frac{p(C_1, C_2)}{p(C_1)p(C_2)}$$



Caveat

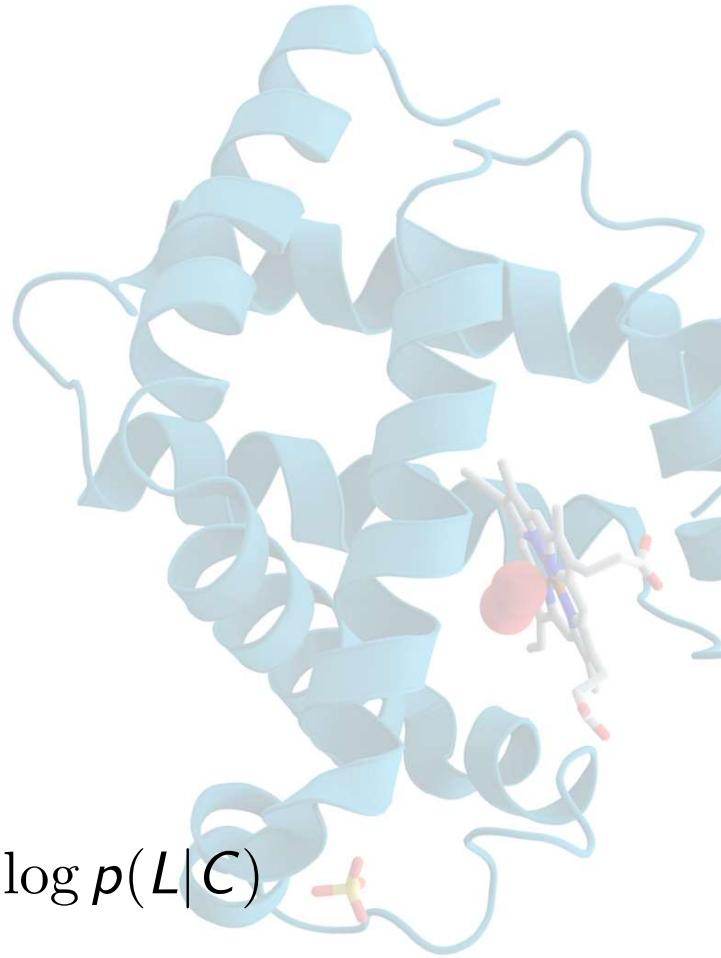
To get meaningful information many k-means++ initializations are run for every crop. The mutual information must be 'adjusted'.



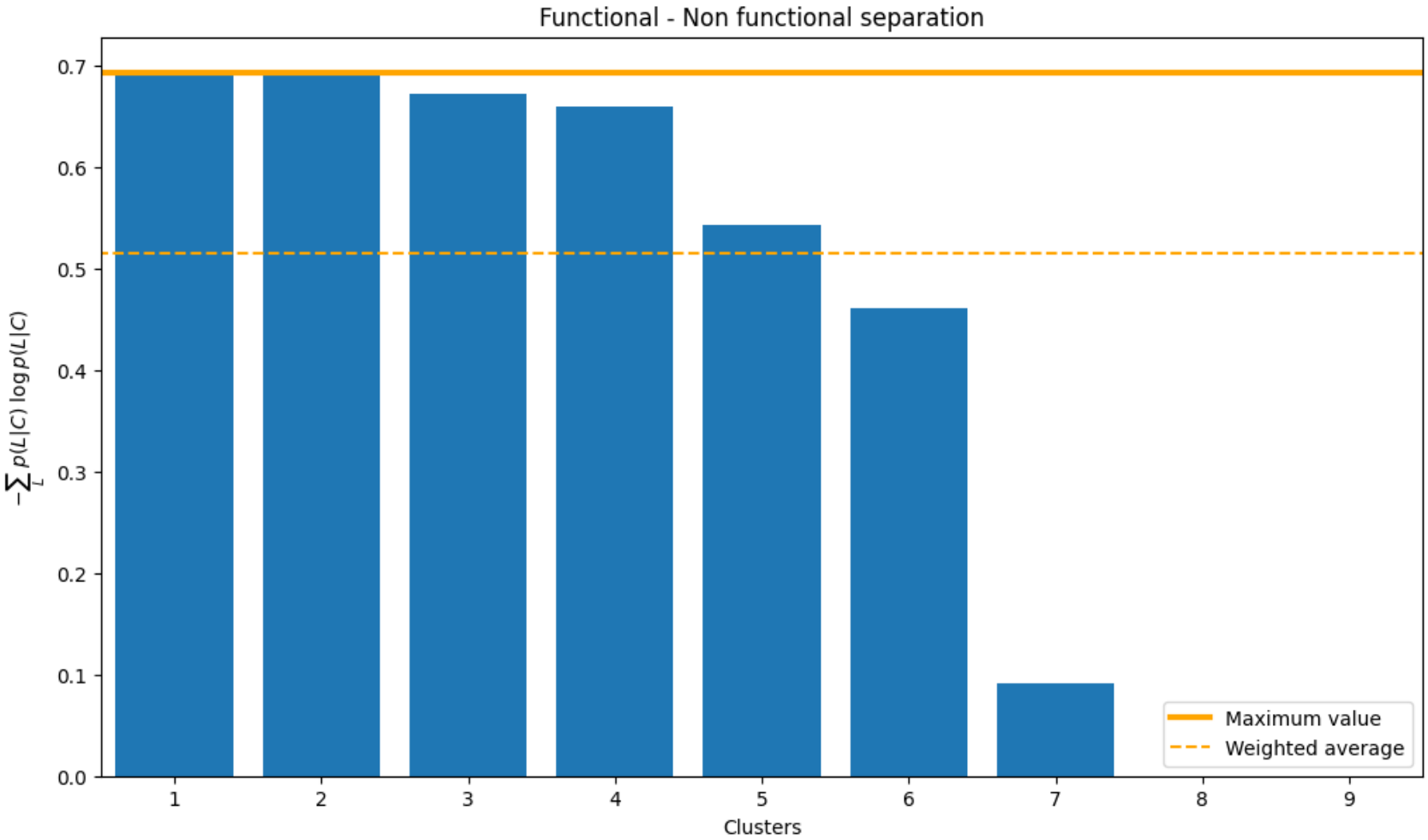


Caveat

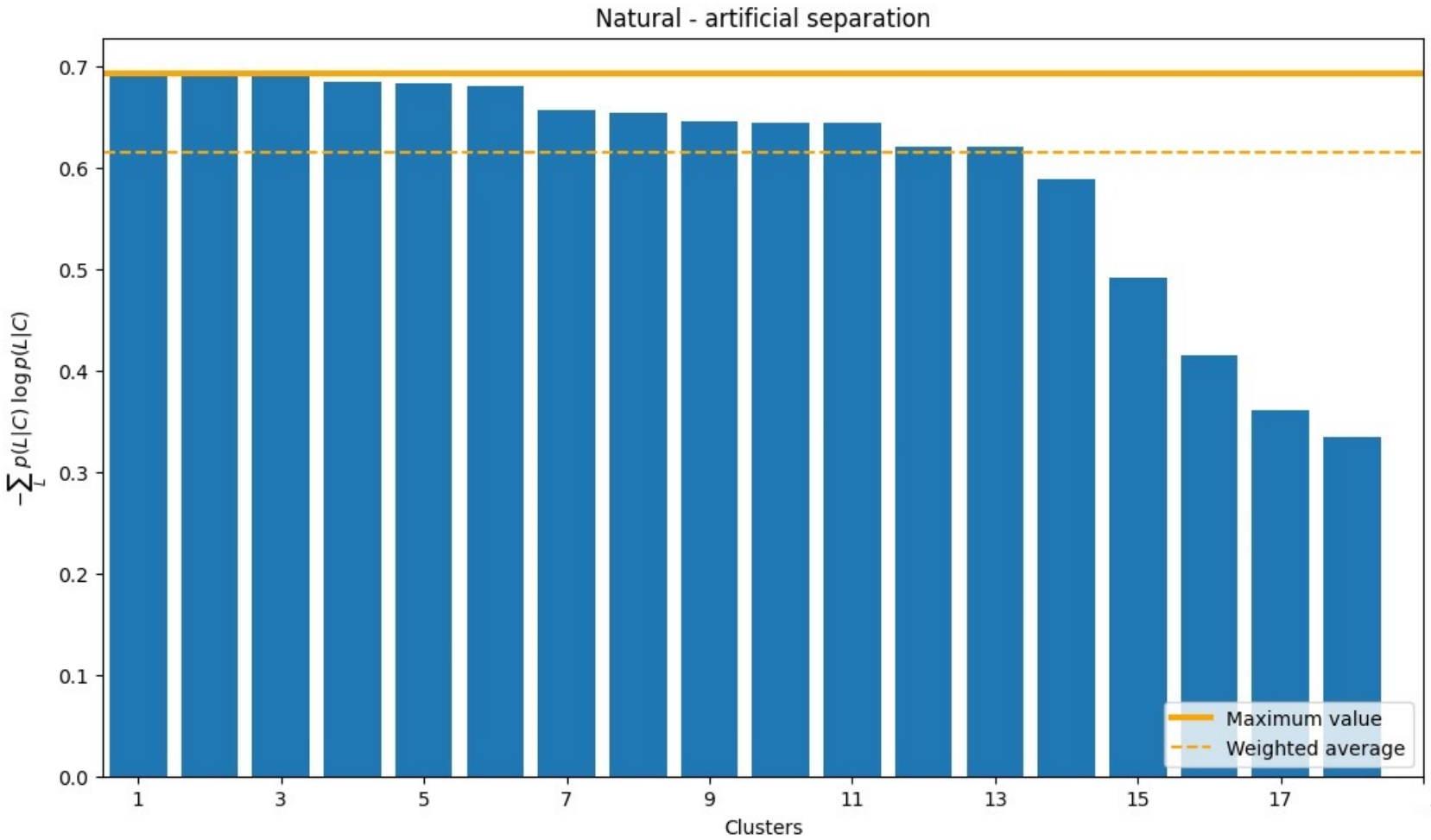
To get meaningful information many k-means++ initializations are run for every crop. The mutual information must be 'adjusted'.



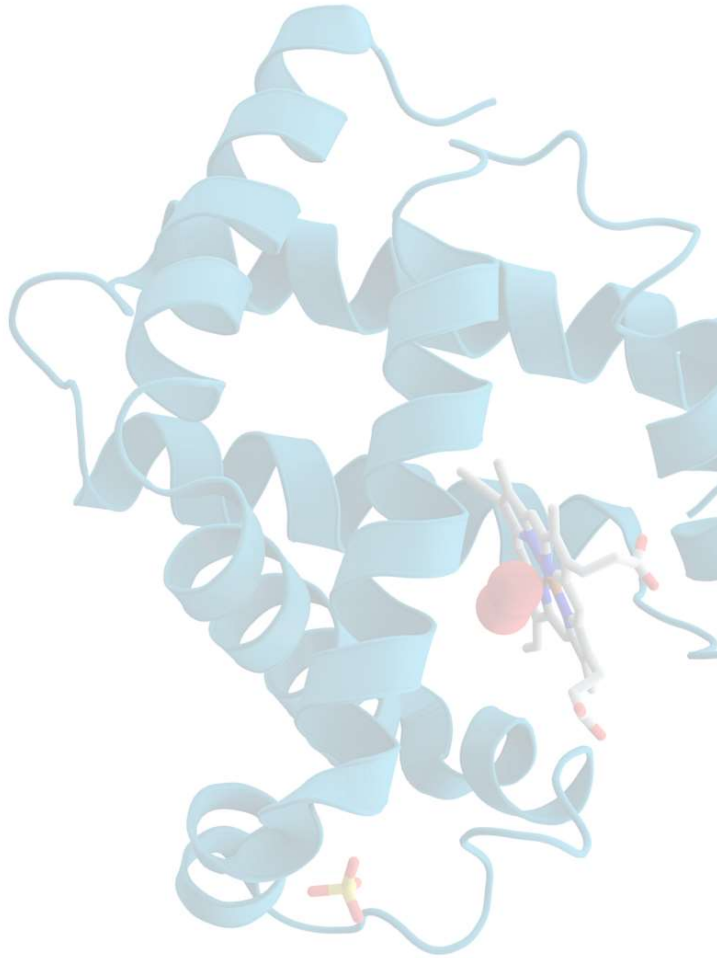
$$\mathbf{H}[L|C] = -\sum_{C} p(C) \sum_{L} p(L|C) \log p(L|C)$$



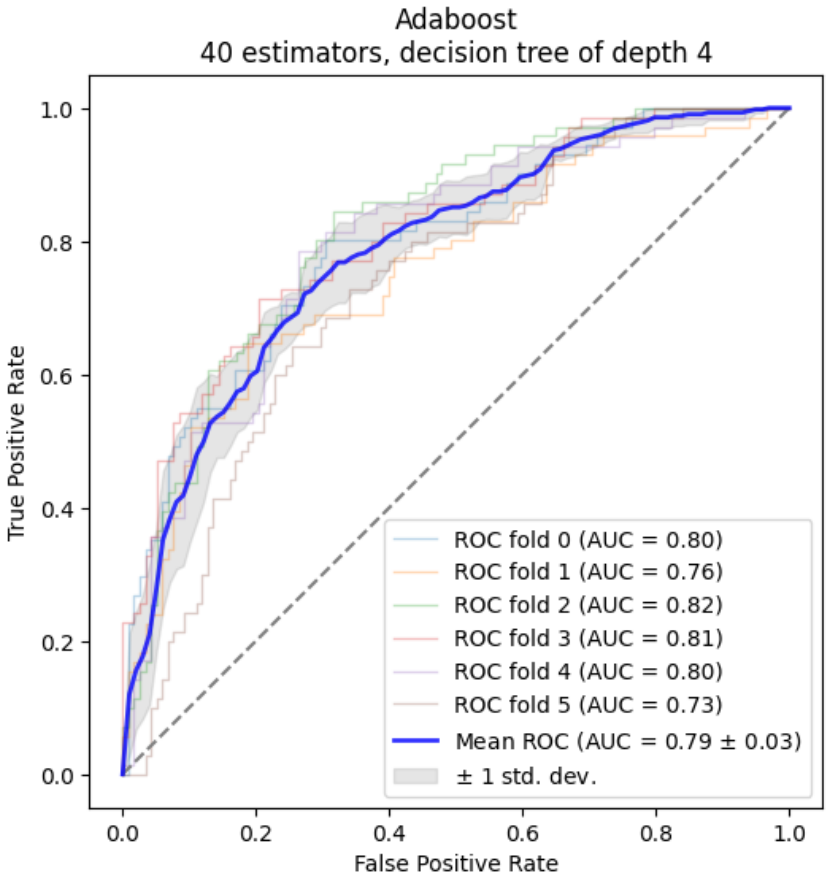
$$\mathbf{H}[L|C] = -\sum_C p(C) \sum_L p(L|C) \log p(L|C)$$

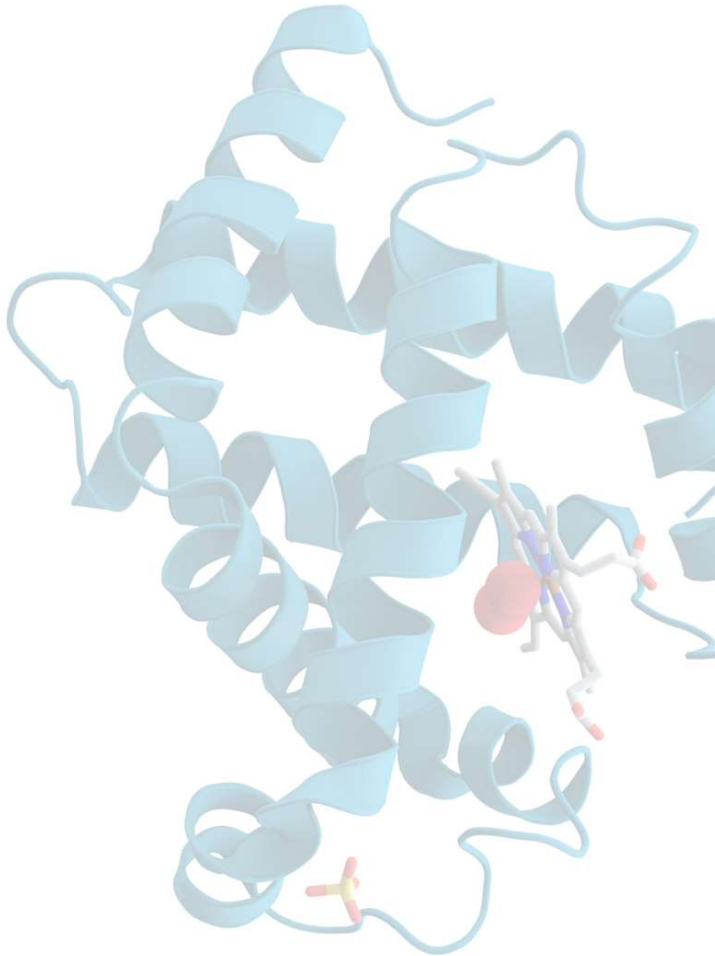


$$\mathbf{H}[L|C] = -\sum_C p(C) \sum_L p(L|C) \log p(L|C)$$

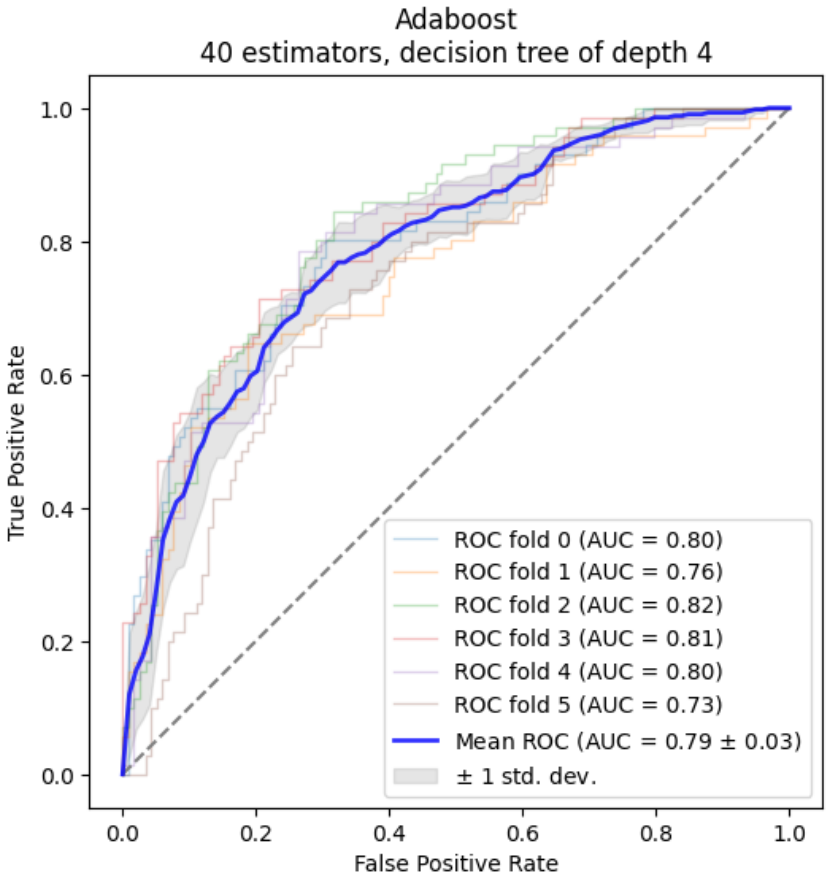


Selecting the best model





Selecting the best model



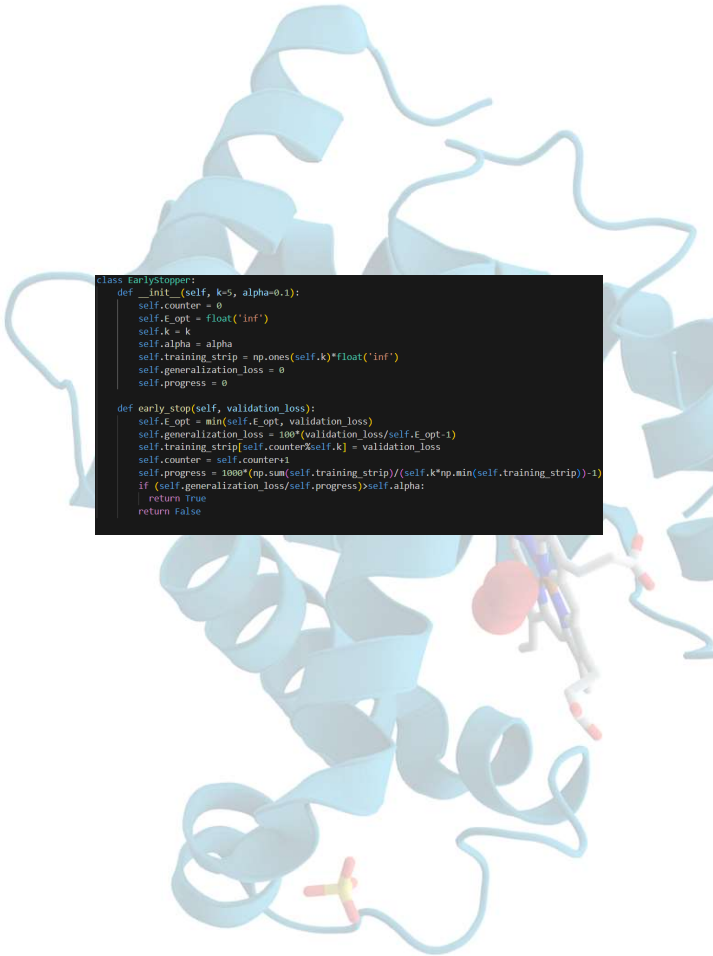
	ROC auc
DT (4)	$(\mathbf{0.75 \pm 0.03})$
DT (7)	$(\mathbf{0.76 \pm 0.02})$
RF (40, 4)	$(\mathbf{0.84 \pm 0.02})$
RF (40, 15)	$(\mathbf{0.87 \pm 0.02})$
RF (300, 15)	$(\mathbf{0.88 \pm 0.01})$
AL (50)	$(\mathbf{0.88 \pm 0.01})$

DT: Decision Tree (depth)
RF: Random Forest
(# of estimators, depth)
AL: Adaboost with logistic regression (# of estimators)

Networks

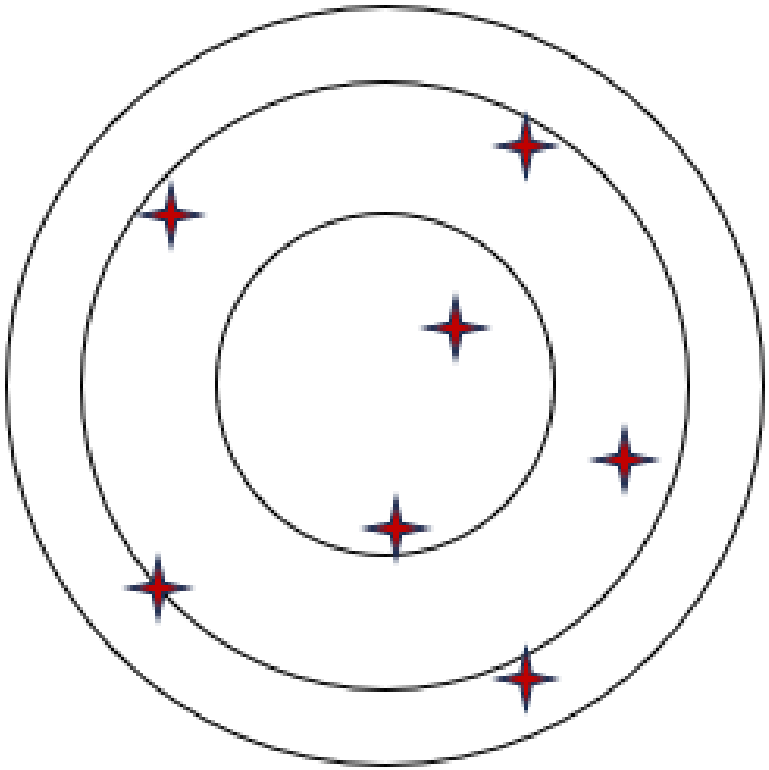
We tried many different architectures and used several tricks to solve some of the problems encountered:

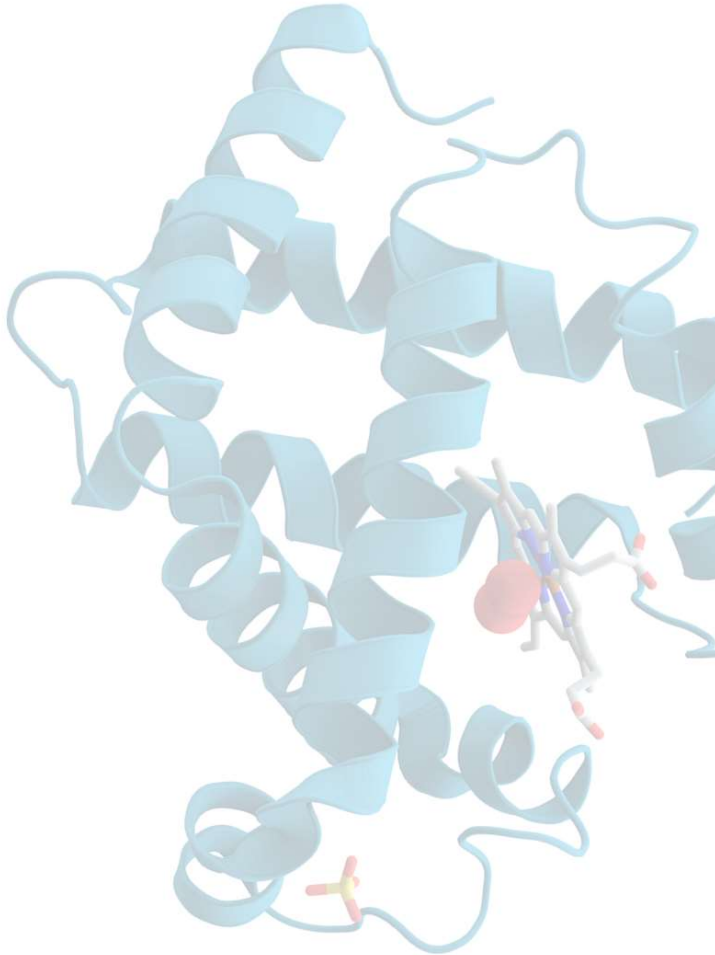
- easily get 100% accuracy on the training set and **overfitting**:
  → inserting **dropout**
  → performing **early stopping**
- problems dealing with an **unbalanced** dataset:
  → using **weighted cost** functions
- **vanishing gradients** with CNN and some sigmoid functions:
  → **batch normalisation**



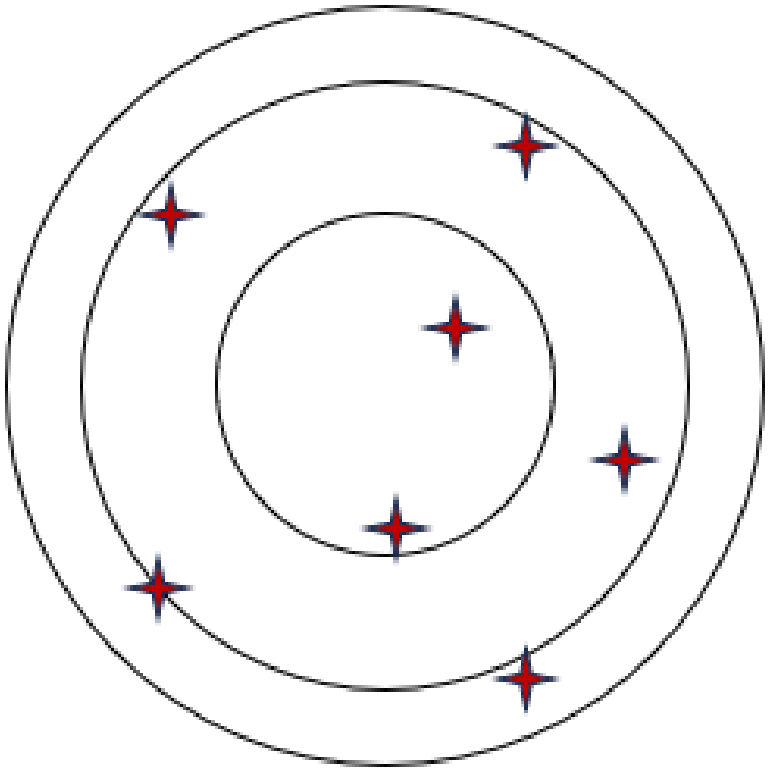
```
class EarlyStopper:
    def __init__(self, k=5, alpha=0.1):
        self.counter = 0
        self.E_opt = float('inf')
        self.k = k
        self.alpha = alpha
        self.training_strip = np.ones(self.k)*float('inf')
        self.generalization_loss = 0
        self.progress = 0

    def early_stop(self, validation_loss):
        self.E_opt = min(self.E_opt, validation_loss)
        self.generalization_loss = 100*(validation_loss/self.E_opt-1)
        self.training_strip[self.counter%self.k] = validation_loss
        self.counter = self.counter+1
        self.progress = 1000*(np.sum(self.training_strip)/(self.k*np.min(self.training_strip))-1)
        if (self.generalization_loss/self.progress)>self.alpha:
            return True
        return False
```

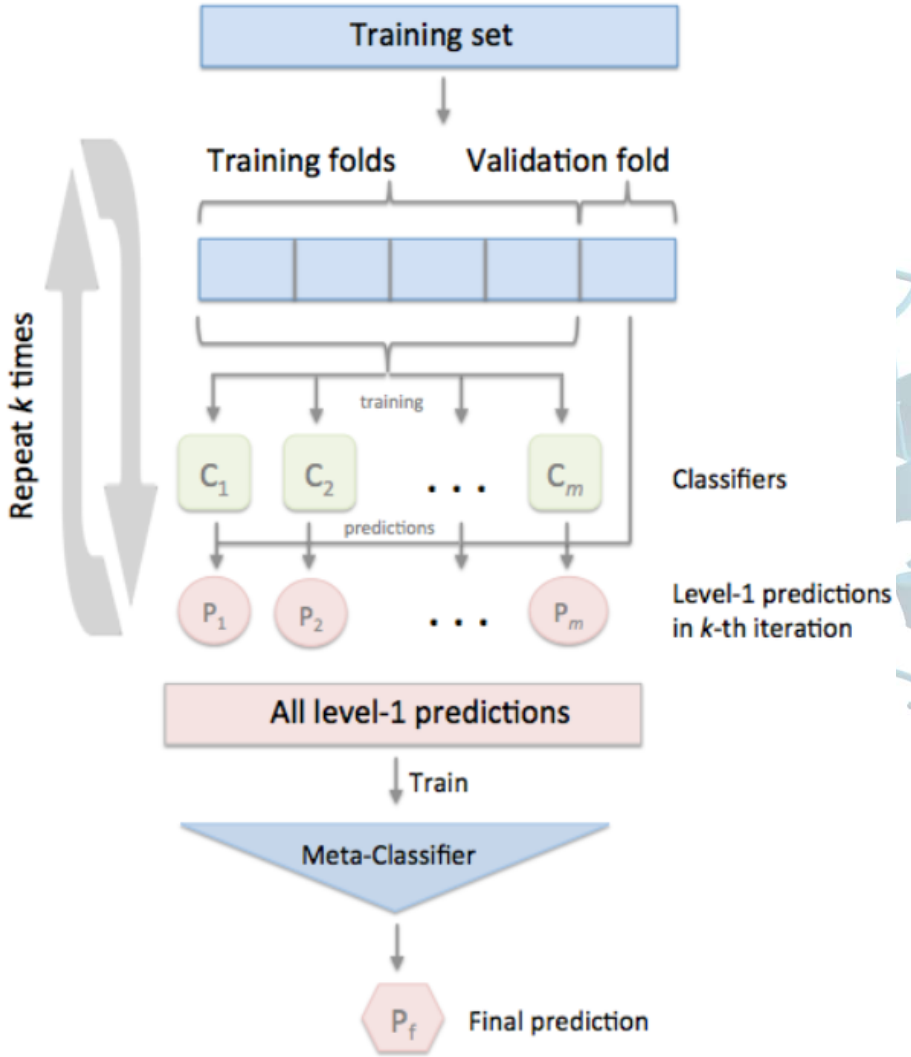
Metaclassifier





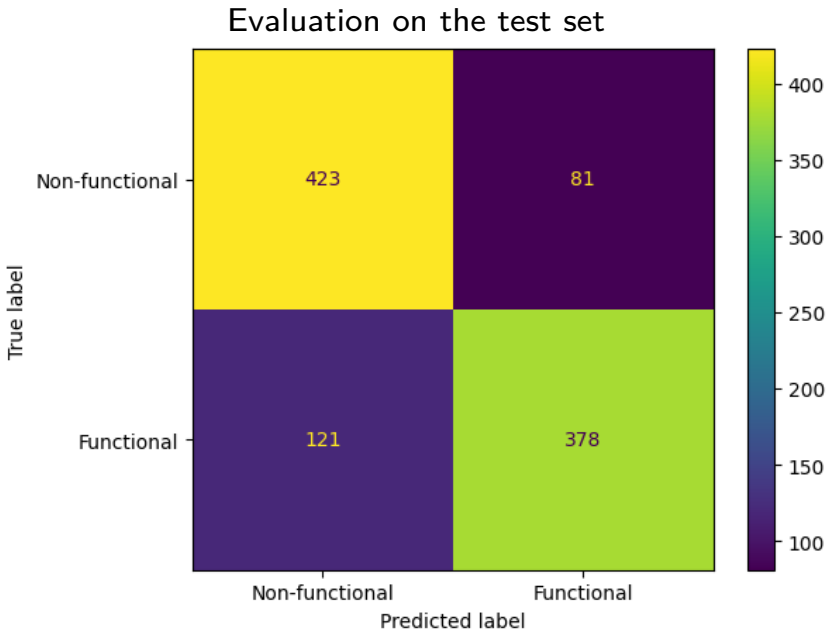
Metaclassifier

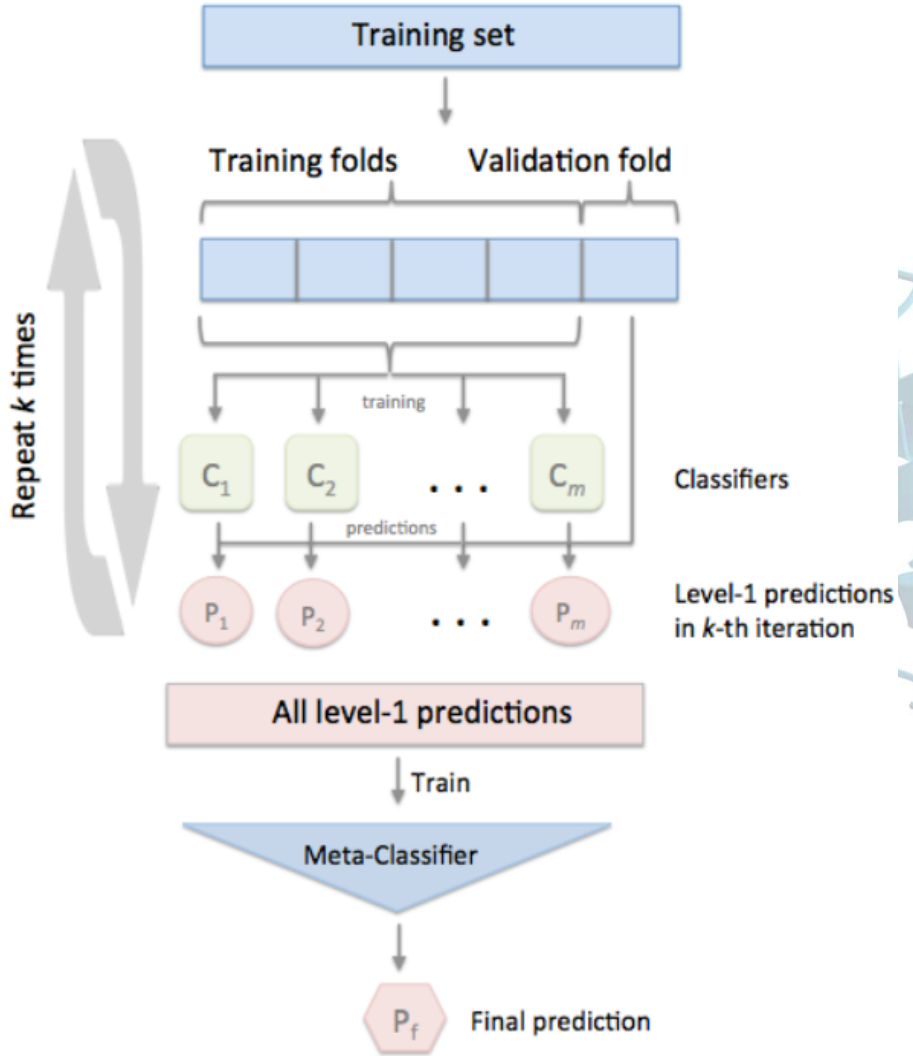




Right image from https://rasbt.github.io/mlxtend/user_guide/classifier/StackingCVClassifier/

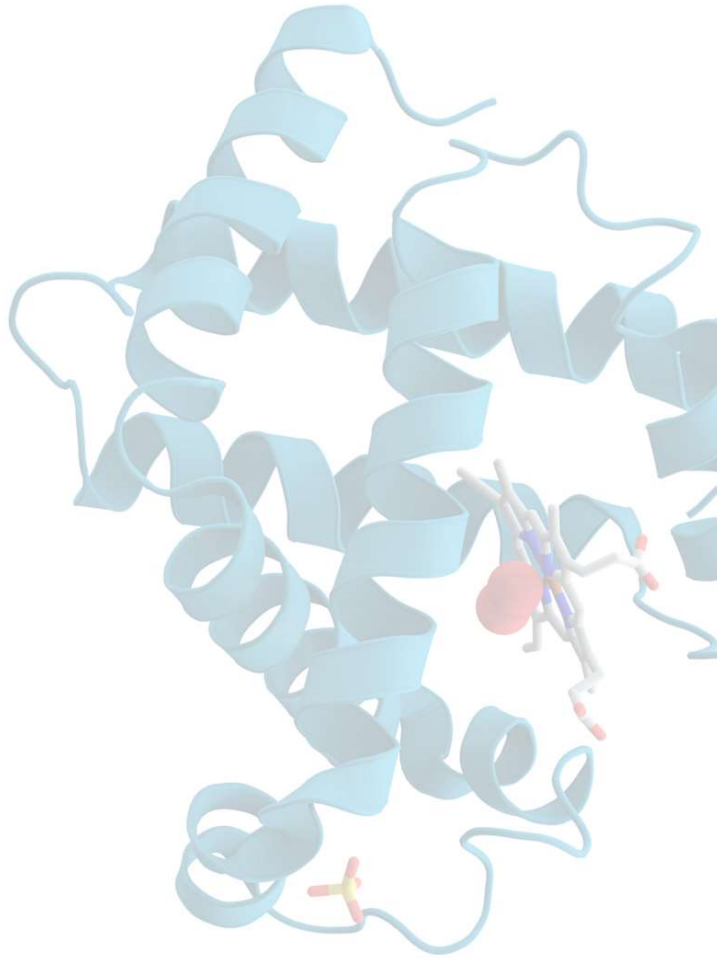
Metaclassifier



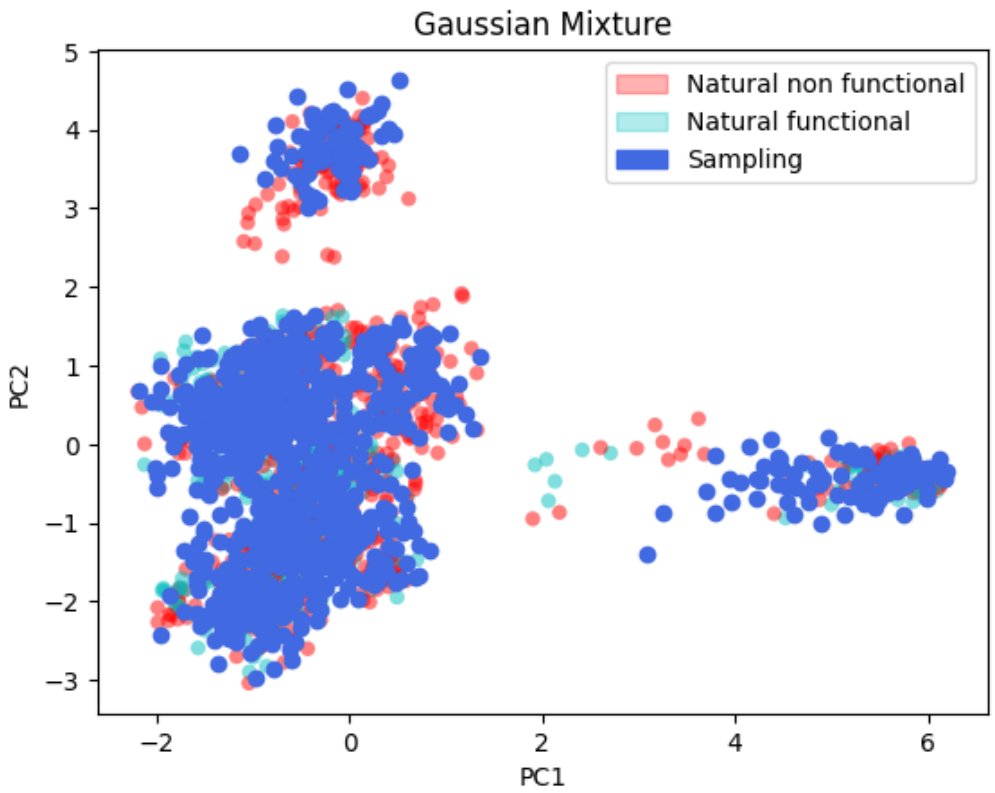


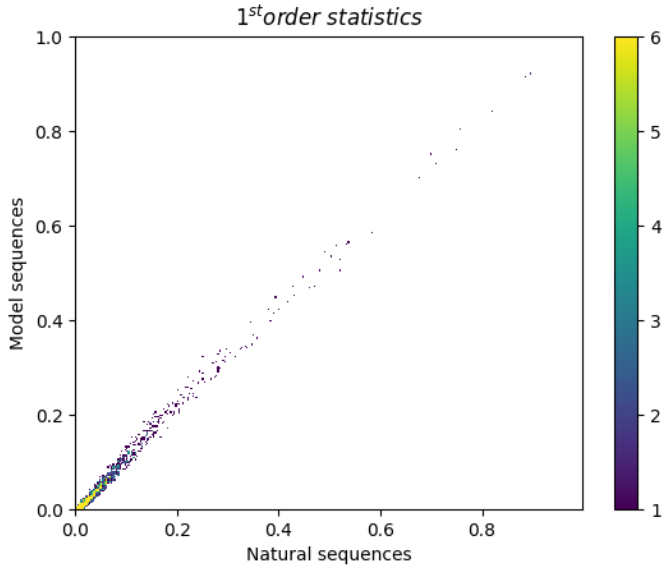
Notice

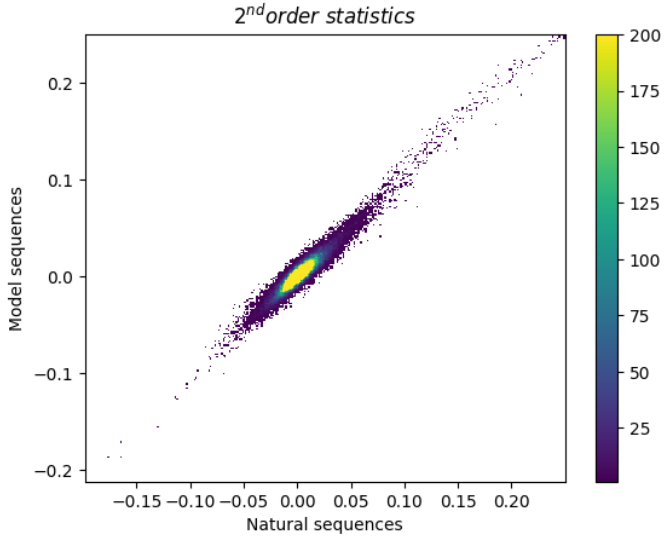
The results shown are obtained through the use of an Adaboost classifier as metaclassifier. With other classifiers it is possible to rank the feature importance and it can be seen that features from classifier that use Isomap embedding can be relevant. In the following parts, for fastness of the computations, we will remove them.



GMM







Using the previously defined classifier you get that 42.6% of the sampled sequences are functional. (Natural functional frequency $\sim 37.4\%$)

Towards an energy model

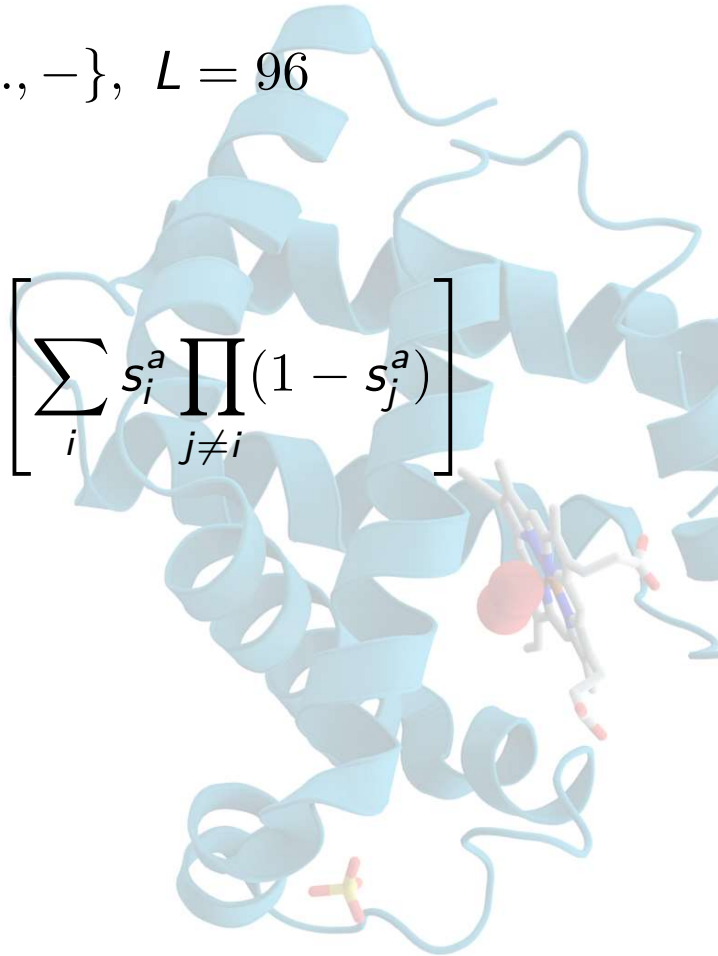
Field only potts model:

$$H(\underline{s}) = -\sum_{i=a}^{L} h_a(s_a) \quad , \; s_a \in \{A, B, ..., -\}, \; L = 96$$

Ising-like model for one-hot-encoded proteins:

$$H(\underline{s}) = -\sum_{i,j,a,b} J_{ij}^{ab} \; s_i^a s_j^b \; - \sum_{i,a} h_i^a \; s_i^a + \gamma \sum_a \left[\sum_i s_i^a \prod_{j \neq i} (1 - s_j^a) \right]$$

$$s_i^a \in \{0, 1\}$$



Towards an energy model

Field only potts model:

$$H(\underline{s}) = -\sum_{i=a}^{L} h_a(s_a) \quad , \; s_a \in \{A, B, ..., -\}, \; L = 96$$

Ising-like model for one-hot-encoded proteins:

$$H(\underline{s}) = -\sum_{i,j,a,b} J_{ij}^{ab} \; s_i^a s_j^b \; - \sum_{i,a} h_i^a \; s_i^a + \gamma \sum_a \left[\sum_i s_i^a \prod_{j \neq i} (1 - s_j^a) \right]$$

So far: pairwise interactions are sufficient for creating an effective generative model

↓

Our question: is a minimally connected model enough to explain different features of the dataset?

W.P. Russ et al., 'An evolution-based model for designing chorismate mutase enzymes,' Nature, Jul 2020)

Towards an energy model

Field only potts model:

$$H(\underline{s}) = -\sum_{i=a}^{L} h_a(s_a) \quad , \; s_a \in \{A, B, ..., -\}, \; L = 96$$

Ising-like model for one-hot-encoded proteins:

$$H(\underline{s}) = -\sum_{i,j,a,b} J_{ij}^{ab} \; s_i^a s_j^b \; - \sum_{i,a} h_i^a \; s_i^a + \gamma \sum_a \left[\sum_i s_i^a \prod_{j \neq i} (1 - s_j^a) \right]$$

So far: pairwise interactions are sufficient for creating an effective generative model

↓

Our question: is a minimally connected model enough to explain different features of the dataset?

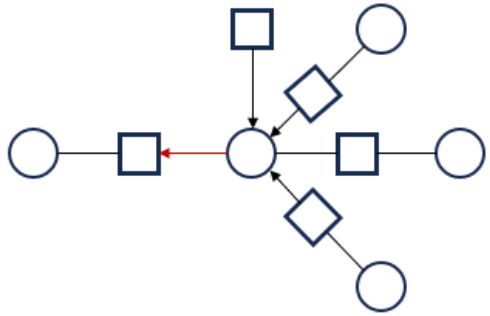
W.P. Russ et al., 'An evolution-based model for designing chorismate mutase enzymes,' Nature, Jul 2020)

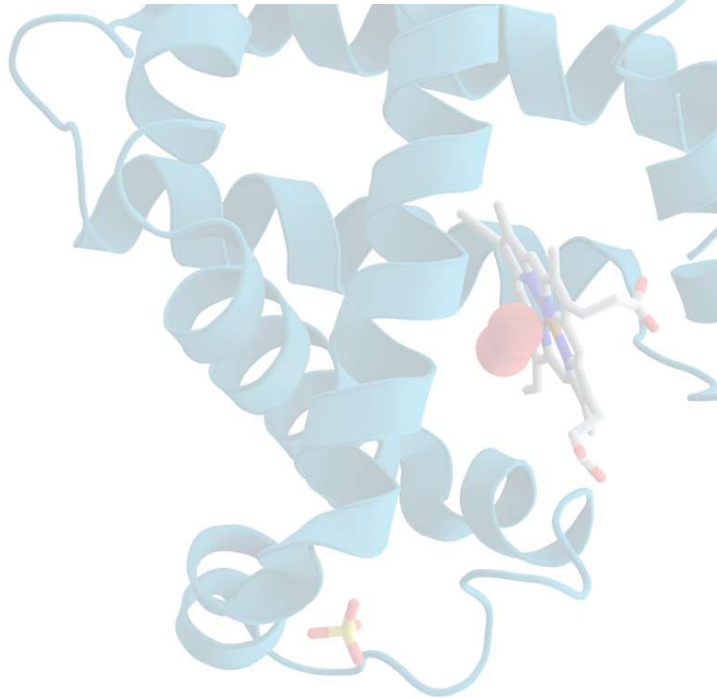
Objective:

$$\{J_{ij}^*, h_i^*\} = \underset{\{J_{ij}, h_i\}}{\operatorname{argmax}} \mathcal{L}$$

$$\frac{\partial \mathcal{L}}{\partial h_i} = \beta(\hat{m}_i - < s_i >_p) = 0$$

$$\frac{\partial \mathcal{L}}{\partial J_{ij}} = \beta(\hat{c}_{ij} - < s_i s_j >_p) = 0$$



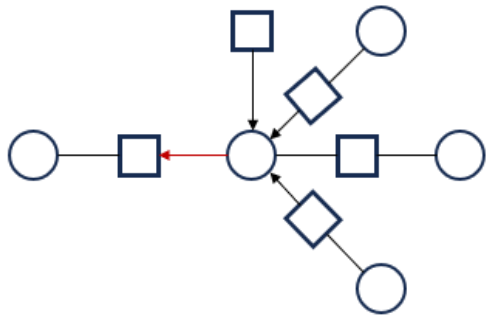


Objective:

$$\{J_{ij}^*, h_i^*\} = \underset{\{J_{ij}, h_i\}}{\operatorname{argmax}} \mathcal{L}$$

$$\frac{\partial \mathcal{L}}{\partial h_i} = \beta(\hat{m}_i - < s_i >_p) = 0$$

$$\frac{\partial \mathcal{L}}{\partial J_{ij}} = \beta(\hat{c}_{ij} - < s_i s_j >_p) = 0$$

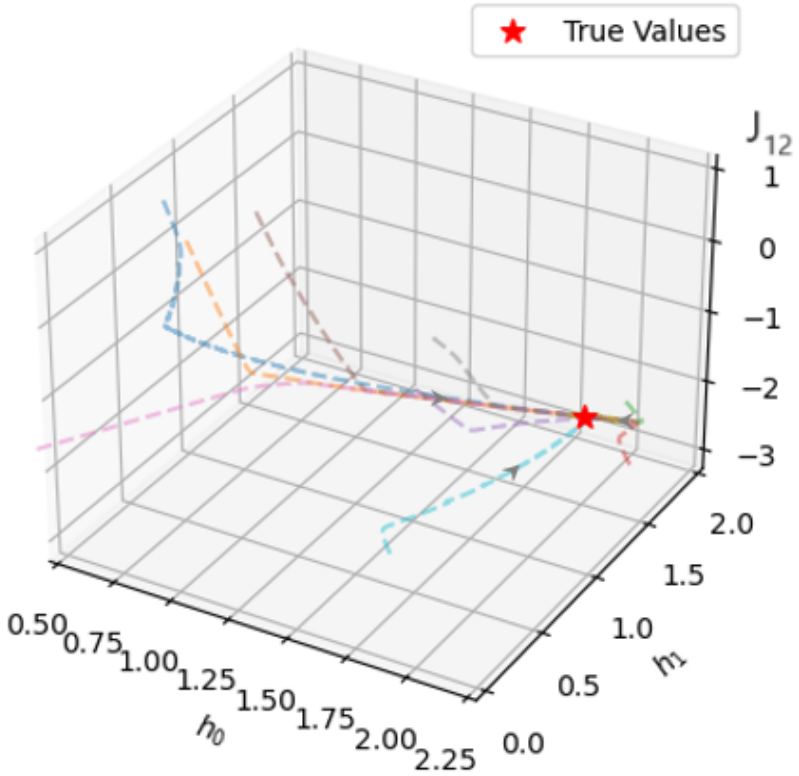


Chow Liu Theorem

The best topology is the Maximum Spanning Tree of the complete graph with $w_{ij} = M_{ij}$

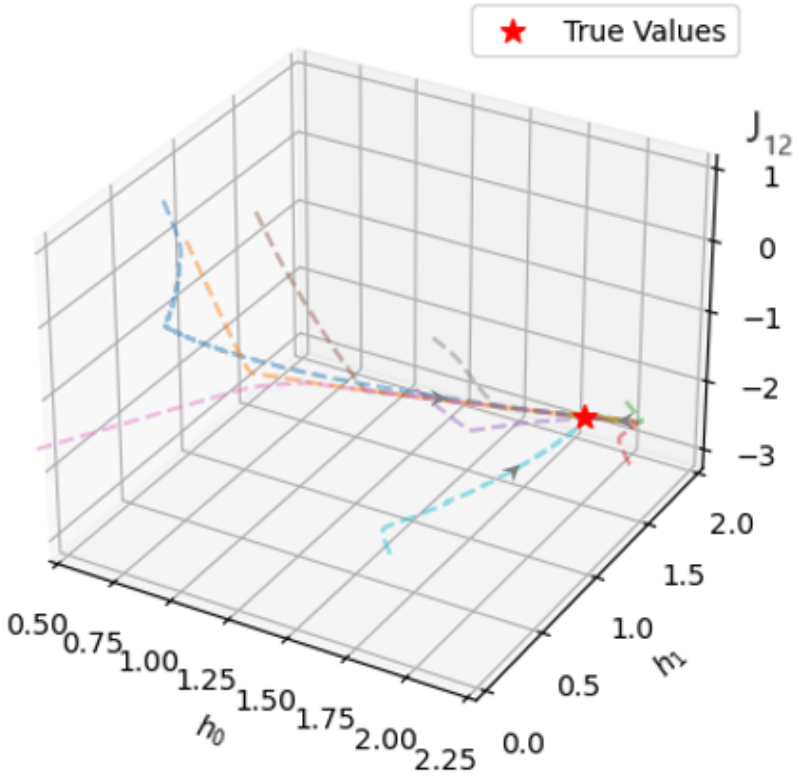
Belief propagation equations

$$b_i(s_i) = \frac{1}{z_i} \prod_{b \in \partial i} m_{bi}(s_i) \qquad b_a(\underline{s}_a) = \frac{1}{z_a} \prod_{i \in \partial a} m_{ia}(x_i)$$



Dynamic of the procedure and convergence for a 2 spin example.

For the protein dataset:



Dynamic of the procedure and convergence for a 2 spin example.

For the protein dataset:

WORK IN PROGRESS...